FIND: Human-in-the-Loop Debugging Deep Text Classifiers

Piyawat Lertvittayakumjorn, Lucia Specia, Francesca Toni
Department of Computing, Imperial College London, UK
{pl1515, l.specia, ft}@imperial.ac.uk

Abstract

Since obtaining a perfect training dataset (i.e., a dataset which is considerably large, unbiased, and well-representative of unseen cases) is hardly possible, many real-world text classifiers are trained on the available, yet imperfect, datasets. These classifiers are thus likely to have undesirable properties. For instance, they may have biases against some sub-populations or may not work effectively in the wild due to overfitting. In this paper, we propose **FIND** – a framework which enables humans to debug deep learning text classifiers by disabling irrelevant hidden features. Experiments show that by using FIND, humans can improve CNN text classifiers which were trained under different types of imperfect datasets (including datasets with biases and datasets with dissimilar train-test distributions).

1 Introduction

Deep learning has become the dominant approach to address most Natural Language Processing (NLP) tasks, including text classification. With sufficient and high-quality training data, deep learning models can perform incredibly well (Zhang et al., 2015; Wang et al., 2019). However, in real-world cases, such ideal datasets are scarce. Often times, the available datasets are small, full of regular but irrelevant words, and contain unintended biases (Wiegand et al., 2019; Gururangan et al., 2018). These can lead to suboptimal models with undesirable properties. For example, the models may have biases against some sub-populations or may not work effectively in the wild as they overfit the imperfect training data.

To improve the models, previous work has looked into different techniques beyond standard model fitting. If the weaknesses of the training datasets or the models are anticipated, strategies can be tailored to mitigate such weaknesses. For example, augmenting the training data with gender-swapped input texts helps reduce gender bias in the models (Park et al., 2018; Zhao et al., 2018). Adversarial training can prevent the models from exploiting irrelevant and/or protected features (Jaiswal et al., 2019; Zhang et al., 2018). With a limited number of training examples, using human rationales or prior knowledge together with training labels can help the models perform better (Zaidan et al., 2007; Bao et al., 2018; Liu and Avci, 2019).

Nonetheless, there are side-effects of sub-optimal datasets that cannot be predicted and are only found after training thanks to post-hoc error analysis. To rectify such problems, there have been attempts to enable humans to fix the trained models (i.e., to perform *model debugging*) (Stumpf et al., 2009; Teso and Kersting, 2019). Since the models are usually too complex to understand, manually modifying the model parameters is not possible. Existing techniques, therefore, allow humans to provide feedback on individual predictions instead. Then, additional training examples are created based on the feedback to retrain the models. However, such local improvements for individual predictions could add up to inferior overall performance (Wu et al., 2019). Furthermore, these existing techniques allow us to rectify only errors related to examples at hand but provide no way to fix problems kept hidden in the model parameters.

In this paper, we propose a framework which allows humans to debug and improve deep text classifiers by disabling hidden features which are irrelevant to the classification task. We name this framework **FIND** (**F**eature **I**nvestigation a**N**d **D**isabling). FIND exploits an explanation method, namely layer-wise relevance propagation (LRP) (Arras et al., 2016), to understand the behavior of a classifier when it predicts each training instance. Then it aggregates all the information using word clouds to create a global visual picture of the model.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 332–348,
November 16–20, 2020.

This enables humans to comprehend the features automatically learned by the deep classifier and then decide to disable some features that could undermine the prediction accuracy during testing. The main differences between our work and existing work are: (i) first, FIND leverages human feedback on the model components, not the individual predictions, to perform debugging; (ii) second, FIND targets deep text classifiers which are more convoluted than traditional classifiers used in existing work (such as Naive Bayes classifiers and Support Vector Machines).

We conducted three human experiments (one feasibility study and two debugging experiments) to demonstrate the usefulness of FIND. For all the experiments, we used as classifiers convolutional neural networks (CNNs) (Kim, 2014), which are a popular, well-performing architecture for many text classification tasks including the tasks we experimented with (Gambäck and Sikdar, 2017; Johnson and Zhang, 2015; Zhang et al., 2019). The overall results show that FIND with human-in-the-loop can improve the text classifiers and mitigate the said problems in the datasets. After the experiments, we discuss the generalization of the proposed framework to other tasks and models. Overall, the **main contributions** of this paper are:

- We propose using word clouds as visual explanations of the features learned.
- We propose a technique to disable the learned features which are irrelevant or harmful to the classification task so as to improve the classifier. This technique and the word clouds form the human-debugging framework – FIND.
- We conduct three human experiments that demonstrate the effectiveness of FIND in different scenarios. The results not only highlight the usefulness of our approach but also reveal interesting behaviors of CNNs for text classification.

The rest of this paper is organized as follows. Section 2 explains related work about analyzing, explaining, and human-debugging text classifiers. Section 3 proposes FIND, our debugging framework. Section 4 explains the experimental setup followed by the three human experiments in Section 5 to 7. Finally, Section 8 discusses generalization of the framework and concludes the paper. Code and datasets of this paper are available at https://github.com/plkumjorn/FIND.

2 Related Work

Analyzing deep NLP models – There has been substantial work in gaining better understanding of complex, deep neural NLP models. By visualizing dense hidden vectors, Li et al. (2016) found that some dimensions of the final representation learned by recurrent neural networks capture the effect of intensification and negation in the input text. Karpathy et al. (2015) revealed the existence of interpretable cells in a character-level LSTM model for language modelling. For example, they found a cell acting as a line length counter and cells checking if the current letter is inside a parenthesis or a quote. Jacovi et al. (2018) presented interesting findings about CNNs for text classification including the fact that one convolutional filter may detect more than one n-gram pattern and may also suppress negative n-grams. Many recent papers studied several types of knowledge in BERT (Devlin et al., 2019), a deep transformer-based model for language understanding, and found that syntactic information is mostly captured in the middle BERT layers while the final BERT layers are the most task-specific (Rogers et al., 2020). Inspired by many findings, we make the assumption that each dimension of the final representation (i.e., the vector before the output layer) captures patterns or qualities in the input which are useful for classification. Therefore, understanding the roles of these dimensions (we refer to them as *features*) is a prerequisite for effective human-in-the-loop model debugging, and we exploit an explanation method to gain such an understanding.

Explaining predictions from text classifiers – Several methods have been devised to generate explanations supporting classifications in many forms, such as natural language texts (Liu et al., 2019), rules (Ribeiro et al., 2018), extracted rationales (Lei et al., 2016), and attribution scores (Lertvittayakumjorn and Toni, 2019). Some explanation methods, such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017), are model-agnostic and do not require access to model parameters. Other methods access the model architectures and parameters to generate the explanations, such as DeepLIFT (Shrikumar et al., 2017) and LRP (layer-wise relevance propagation) (Bach et al., 2015; Arras et al., 2016). In this work, we use LRP to explain not the predictions but the learned features so as to expose the model behavior to humans and enable informed model debugging.

Debugging text classifiers using human feedback – Early work in this area comes from the human-computer interaction community. Stumpf et al. (2009) studied the types of feedback humans usually give in response to machine-generated predictions and explanations. Also, some of the feedback collected (i.e., important words of each category) was used to improve the classifier via a user co-training approach. Kulesza et al. (2015) presented an explanatory debugging approach in which the system explains to users how it made each prediction, and the users then rectify the model by adding/removing words from the explanation and adjusting important weights. Even without explanations shown, an active learning framework proposed by Settles (2011) asks humans to iteratively label some chosen features (i.e., words) and adjusts the model parameters that correspond to the features. However, these early works target simpler machine learning classifiers (e.g., Naive Bayes classifiers with bag-of-words) and it is not clear how to apply the proposed approaches to deep text classifiers.

Recently, there have been new attempts to use explanations and human feedback to debug classifiers in general. Some of them were tested on traditional text classifiers. For instance, Ribeiro et al. (2016) showed a set of LIME explanations for individual SVM predictions to humans and asked them to remove irrelevant words from the training data in subsequent training. The process was run for three rounds to iteratively improve the classifiers. Teso and Kersting (2019) proposed CAIPI, which is an explanatory interactive learning framework. At each iteration, it selects an unlabelled example to predict and explain to users using LIME, and the users respond by removing irrelevant features from the explanation. CAIPI then uses this feedback to generate augmented data and retrain the model.

While these recent works use feedback on low-level features (input words) and individual predictions, our framework (FIND) uses feedback on the learned features with respect to the big picture of the model. This helps us avoid local decision pitfalls which usually occur in interactive machine learning (Wu et al., 2019). Overall, what makes our contribution different from existing work is that *(i)* we collect the feedback on the model, not the individual predictions, and *(ii)* we target deep text classifiers which are more complex than the models used in previous work.

3 FIND: Debugging Text Classifiers

3.1 Motivation

Generally, deep text classifiers can be divided into two parts. The first part performs *feature extraction*, transforming an input text into a dense vector (i.e., a *feature vector*) which represents the input. There are several alternatives to implement this part such as using convolutional layers, recurrent layers, and transformer layers. The second part performs *classification* passing the feature vector through a dense layer with softmax activation to get predicted probability of the classes. These deep classifiers are not transparent, as humans cannot interpret the meaning of either the intermediate vectors or the model parameters used for feature extraction. This prevents humans from applying their knowledge to modify or debug the classifiers.

In contrast, if we understand which patterns or qualities of the input are captured in each feature, we can comprehend the overall reasoning mechanism of the model as the dense layer in the classification part then becomes interpretable. In this paper, we make this possible using LRP. By understanding the model, humans can check whether the input patterns detected by each feature are relevant for classification. Also, the features should be used by the subsequent dense layer to support the right classes. If these are not the case, debugging can be done by disabling the features which may be harmful if they exist in the model. Figure 1 shows the overview of our debugging framework, FIND.

3.2 Notation

Let us consider a text classification task with $|\mathcal{C}|$ classes where $\mathcal{C}$ is the set of all classes and let $\mathcal{V}$ be a set of unique words in the corpus (the vocabulary). A training dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is given, where x_i is the i-th document containing a sequence of L words, $[x_{i1}, x_{i2}, ..., x_{iL}]$, and $y_i \in \mathcal{C}$ is the class label of x_i. A deep text classifier M trained on dataset $\mathcal{D}$ classifies a new input document x into one of the classes (i.e., $M(x) \in \mathcal{C}$). In addition, M can be divided into two parts – a feature extraction part M_f and a classification part M_c. Formally, $M(x) = (M_c \circ M_f)(x)$; $M_f(x) = \mathbf{f}$; $M(x) = M_c(\mathbf{f}) = \text{softmax}(\mathbf{W}\mathbf{f} + \mathbf{b}) = \mathbf{p}$ where $\mathbf{f} = [f_1, f_2, \ldots, f_d] \in \mathbb{R}^d$ is the feature vector of x, while $\mathbf{W} \in \mathbb{R}^{|\mathcal{C}| \times d}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{C}|}$ are parameters of the dense layer of M_c. The final output is the predicted probability vector $\mathbf{p} \in [0, 1]^{|\mathcal{C}|}$.

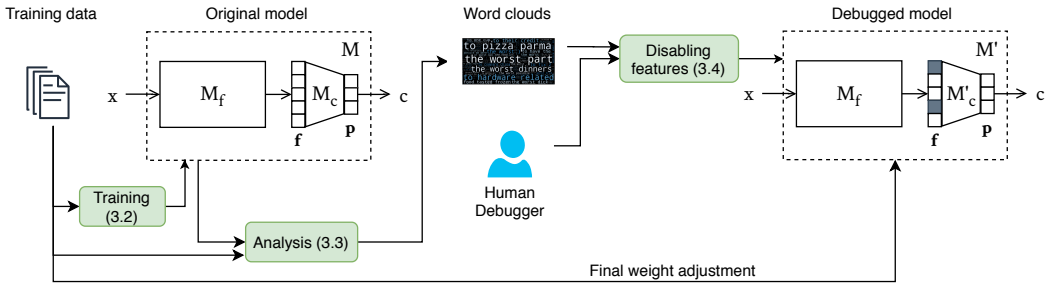

Figure 1: Overview of the proposed debugging framework, FIND. The numbers in the green boxes refer to the corresponding Sections in this paper.

3.3 Understanding the Model

To understand how the model M works, we analyze the patterns or characteristics of the input that activate each feature f_i. Specifically, using LRP[1], for each f_i of an example x_j in the training dataset, we calculate a relevance vector $\mathbf{r}_{ij} \in \mathbb{R}^L$ showing the relevance scores (the contributions) of each word in x_j for the value of f_i. After doing this for all d features of all training examples, we can produce word clouds to help the users better understand the model M.

Word clouds – For each feature f_i, we create (one or more) word clouds to visualize the patterns in the input texts which highly activate f_i. This can be done by analyzing $\mathbf{r}_{ij}$ for all x_j in the training data and displaying, in the word clouds, words or n-grams which get high relevance scores. Note that different model architectures may have different ways to generate the word clouds so as to effectively reveal the behavior of the features.

For CNNs, the classifiers we experiment with in this paper, each feature has one word cloud containing the n-grams, from the training examples, which were selected by the max-pooling of the CNNs. For instance, Figure 2, corresponding to a feature of filter size 2, shows bi-grams (e.g., "love love", "love my", "loves his", etc.) whose font size corresponds to the feature values of the bi-grams. This is similar to how previous works analyze CNN features (Jacovi et al., 2018; Lertvittayakumjorn and Toni, 2019), and it is equivalent to back-propagating the feature values to the input using LRP and cropping the consecutive input words with non-zero LRP scores to show in the word clouds.[2]

[1]See Appendix A for more details on how LRP works.

[2]We also propose how to create word clouds and perform debugging for bidirectional LSTM networks (Hochreiter and Schmidhuber, 1997) in Appendix C.

Figure 2: A word cloud (or, literally, an n-gram cloud) of a feature from a CNN.

3.4 Disabling Features

As explained earlier, we want to know whether the learned features are valid and relevant to the classification task and whether or not they get appropriate weights from the next layer. This is possible by letting humans consider the word cloud(s) of each feature and tell us which class the feature is relevant to. A word cloud receiving human answers that are different from the class it should support (as indicated by $\mathbf{W}$) exhibits a flaw in the model. For example, if the word cloud in Figure 2 represents the feature f_i in a sentiment analysis task but the i^{th} column of $\mathbf{W}$ implies that f_i supports the negative sentiment class, we know the model is not correct here. If this word cloud appears in a product categorization task, this is also problematic because the phrases in the word cloud are not discriminative of any product category. Hence, we provide options for the users to disable the features which correspond to any problematic word clouds so that the features do not play a role in the classification. To enable this to happen, we modify M_c to be M'_c where $\mathbf{p} = M'_c(\mathbf{f}) = \text{softmax}((\mathbf{W} \odot \mathbf{Q})\mathbf{f} + \mathbf{b})$ and $\mathbf{Q} \in \mathbb{R}^{|C| \times d}$ is a masking matrix with $\odot$ being an element-wise multiplication operator. Initially, all elements in $\mathbf{Q}$ are ones which enable all the connections between the features and the output. To disable feature f_i, we set the i^{th} column of $\mathbf{Q}$

Exp	Dataset	$\|\mathcal{C}\|$	Train / Dev / Test
1	Yelp	2	500 / 100 / 38000
	Amazon Products	4	100 / 100 / 20000
2	Biosbias	2	3832 / 1277 / 1278
	Waseem	2	10144 / 3381 / 3382
	Wikitoxic	2	- / - / 18965
3	20Newsgroups	2	863 / 216 / 717
	Religion	2	- / - / 1819
	Amazon Clothes	2	3000 / 300 / 10000
	Amazon Music	2	- / - / 8302
	Amazon Mixed	2	- / - / 100000

Table 1: Datasets used in the experiments.

to be a zero vector. After disabling features, we then freeze the parameters of M_f and fine-tune the parameters of M'_c (except the masking matrix $\mathbf{Q}$) with the original training dataset $\mathcal{D}$ in the final step.

4 Experimental Setup

All datasets and their splits used in the experiments are listed in Table 1. We will explain each of them in the following sections. For each classification task, we ran and improved three models, using different random seeds, independently of one another, and the reported results are the average of the three runs. Regarding the models, we used 1D CNNs with the same structures for all the tasks and datasets. The convolution layer had three filter sizes [2, 3, 4] with 10 filters for each size (i.e., $d = 10 \times 3 = 30$). All the activation functions were ReLU except the softmax at the output layer. The input documents were padded or trimmed to have 150 words ($L = 150$). We used pre-trained 300-dim GloVe vectors (Pennington et al., 2014) as non-trainable weights in the embedding layers. All the models were implemented using Keras and trained with Adam optimizer. We used iNNvestigate (Alber et al., 2018) to run LRP on CNN features. In particular, we used the LRP-ϵ propagation rule to stabilize the relevance scores ($\epsilon = 10^{-7}$). Finally, we used Amazon Mechanical Turk (MTurk) to collect crowdsourced responses for selecting features to disable. Each question was answered by ten workers and the answers were aggregated using majority votes or average scores depending on the question type (as explained next).

5 Exp 1: Feasibility Study

In this feasibility study, we assessed the effectiveness of word clouds as visual explanations to reveal the behavior of CNN features. We trained CNN models using small training datasets and evaluated the quality of CNN features based on responses from MTurk workers to the feature word clouds. Then we disabled features based on their average quality scores. The assumption was: if the scores of the disabled features correlated with the drop in the model predictive performance, it meant that humans could understand and accurately assess CNN features using word clouds. We used small training datasets so that the trained CNNs had features with different levels of quality. Some features detected useful patterns, while others overfitted the training data.

5.1 Datasets

We used subsets of two datasets: *(1)* **Yelp** – predicting sentiments of restaurant reviews (positive or negative) (Zhang et al., 2015) and *(2)* **Amazon Products** – classifying product reviews into one of four categories (Clothing Shoes and Jewelry, Digital Music, Office Products, or Toys and Games) (He and McAuley, 2016). We sampled 500 and 100 examples to be the training data for Yelp and Amazon Products, respectively.

5.2 Human Feedback Collection and Usage

We used human responses on MTurk to assign ranks to features. As each classifier had 30 original features ($d = 30$), we divided them into three ranks (A, B, and C) each of which with 10 features. We expected that features in rank A are most relevant and useful for the prediction task, and features in rank C least relevant, potentially undermining the performance of the model. To make the annotation more accessible to lay users, we designed the questions to ask whether a given word cloud is (mostly or partially) relevant to one of the classes or not, as shown in Figure 3. If the answer matches how the model really uses this feature (as indicated by $\mathbf{W}$), the feature gets a positive score from this human response. For example, if the CNN feature of the word cloud in Figure 3 is used by the model for the negative sentiment class, the scores of the five options in the figure are -2, -1, 0, 1, 2, respectively. We collected ten responses for each question and used the average score to sort the features descendingly. After sorting, the 1^{st}-10^{th} features, 11^{th}-20^{th} features, and 21^{st}-30^{th} features are considered as rank A, B, and C, respectively.[3] To show the effects of feature disabling, we compared the original model M with the modified

[3]The questions and scoring criteria for the Amazon Products dataset, which is a multiclass classification task, are slightly different. See Appendix B for details.

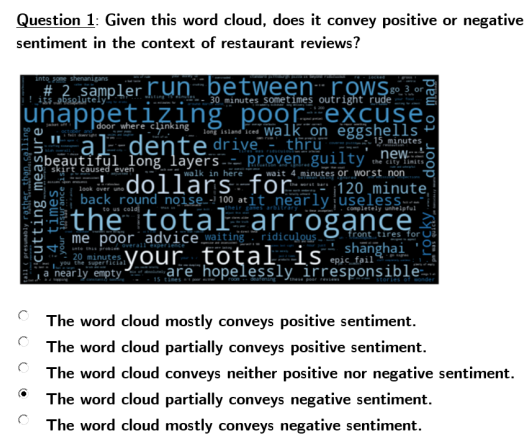

Figure 3: A user interface in Experiment 1 (Yelp).

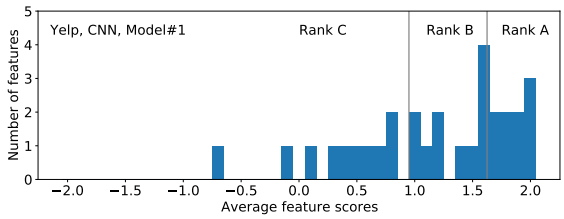

Figure 4: The distribution of average feature scores in a CNN model trained on the Yelp dataset.

Figure 5: Examples of word clouds of CNN features in ranks A, B, and C (Experiment 1, Yelp – sentiment).

model M' with features in rank X disabled where X $\in$ {A, B, C, A and B, A and C, B and C}.

5.3 Results and Discussions

Figure 4 shows the distribution of average feature scores from one of the three CNN instances for the Yelp dataset. Examples of the word clouds from each rank are displayed in Figure 5. We can clearly see dissimilar qualities of the three features. Some participants answered that the rank B feature in Figure 5 was relevant to the positive class (probably due to the word 'delicious'), and the weights of this feature in $\mathbf{W}$ agreed (Positive:Negative = 0.137:-0.135). Interestingly, the rank C feature in Figure 5 got a negative score because some participants believed that this word cloud was relevant to the positive class, but actually the model used this feature as evidence for the negative class (Positive:Negative = 0.209:0.385).

Considering all the three runs, Figure 6 (top) shows the average macro F1 score of the original model (the blue line) and of each modified model. The order of the performance drops is AB > A > AC > BC > B > Original > C. This makes sense because disabling important features (rank A and/or B) caused larger performance drops, and the overall results are consistent with the average feature scores given by the participants (as in Figure 4). It confirms that using word clouds is an effective way to assess CNN features. Also, it is worth noting that the macro F1 of the model slightly increased when we disabled the low-quality features (rank C). This shows that humans can improve the model by disabling irrelevant features.

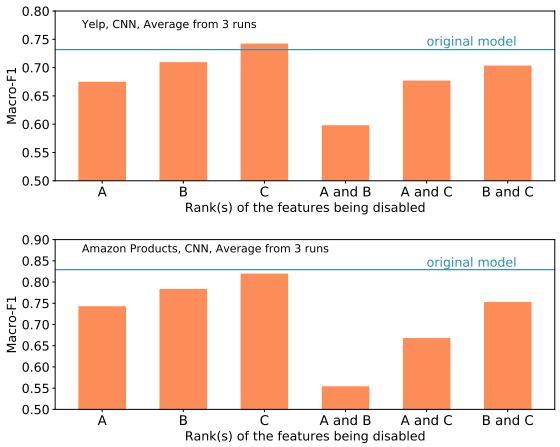

Figure 6: The average macro F1, from the three runs, of all the CNN models for the Yelp dataset (top) and the Amazon Products dataset (bottom).

The CNNs for the Amazon Products dataset also

behaved in a similar way (Figure 6 – bottom), except that disabling rank C features slightly undermined, not increased, performance. This implies that even the rank C features contain a certain amount of useful knowledge for this classifier.[4]

6 Exp 2: Training Data with Biases

Given a biased training dataset, a text classifier may absorb the biases and produce biased predictions against some sub-populations. We hypothesize that if the biases are captured by some of the learned features, we can apply FIND to disable such features and reduce the model biases.

6.1 Datasets and Metrics

We focus on reducing gender bias of CNN models trained on two datasets – **Biosbias** (De-Arteaga et al., 2019) and **Waseem** (Waseem and Hovy, 2016). For Biosbias, the task is predicting the occupation of a given bio paragraph, i.e., whether the person is 'a surgeon' (class 0) or 'a nurse' (class 1). Due to the gender imbalance in each occupation, a classifier usually exploits gender information when making predictions. As a result, bios of female surgeons and male nurses are often misclassified. For Waseem, the task is abusive language detection – assessing if a given text is abusive (class 1) or not abusive (class 0). Previous work found that this dataset contains a strong negative bias against females (Park et al., 2018). In other words, texts related to females are usually classified as abusive although the texts themselves are not abusive at all. Also, we tested the models, trained on the Waseem dataset, using another abusive language detection dataset, **Wikitoxic** (Thain et al., 2017), to assess generalizability of the models. To quantify gender biases, we adopted two metrics – false positive equality difference (FPED) and false negative equality difference (FNED) (Dixon et al., 2018). The lower these metrics are, the less biases the model has.

[4]We also conducted the same experiments here with bidirectional LSTM networks (BiLSTMs) which required a different way to generate the word clouds (see Appendix C). The results on BiLSTMs, however, are not as promising as on CNNs. This might be because the way we created word clouds for each BiLSTM feature was not an accurate way to reveal its behavior. Unlike for CNNs, understanding recurrent neural network features for text classification is still an open problem.

6.2 Human Feedback Collection and Usage

Unlike the interface in Figure 3, for each word cloud, we asked the participants to select the relevant class from three options (Biosbias: surgeon, nurse, it could be either / Waseem: abusive, non-abusive, it could be either). The feature will be disabled if the majority vote does not select the class suggested by the weight matrix $\mathbf{W}$. To ensure that the participants do not use their biases while answering our questions, we firmly mentioned in the instructions that gender-related terms should not be used as an indicator for one or the other class.

6.3 Results and Discussions

The results of this experiment are displayed in Figure 7. For Biosbias, on average, the participants' responses suggested us to disable 11.33 out of 30 CNN features. By doing so, the FPED of the models decreased from 0.250 to 0.163, and the FNED decreased from 0.338 to 0.149. After investigating the word clouds of the CNN features, we found that some of them detected patterns containing both gender-related terms and occupation-related terms such as "his surgical expertise" and "she supervises nursing students". Most of the MTurk participants answered that these word clouds were relevant to the occupations, and thus the corresponding features were not disabled. However, we believe that these features might contain gender biases. So, we asked one annotator to consider all the word clouds again and disable every feature for which the prominent n-gram patterns contained any gender-related terms, no matter whether the patterns detect occupation-related terms. With this new disabling policy, 12 out of 30 features were disabled on average, and the model biases further decreased, as shown in Figure 7 (Debugged (One)). The side-effect of disabling 33% of all the features here was only a slight drop in the macro F1 from 0.950 to 0.933. Hence, our framework was successful in reducing gender biases without severe negative effects in classification performance.

Concerning the abusive language detection task, on average, the MTurk participants' responses suggested us to disable 12 out of 30 CNN features. Unlike Biosbias, disabling features based on MTurk responses unexpectedly increased the gender bias for both Waseem and Wikitoxic datasets. However, we found one similar finding to Biosbias, that many of the CNN features captured n-grams which were both abusive and related to a gender such as 'these

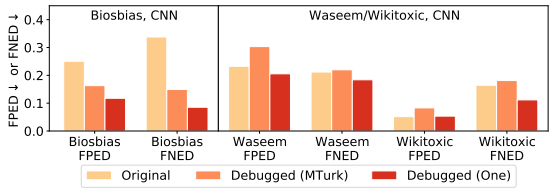

Figure 7: The average FPED and FNED of the CNN models in Experiment 2 (the lower, the better).

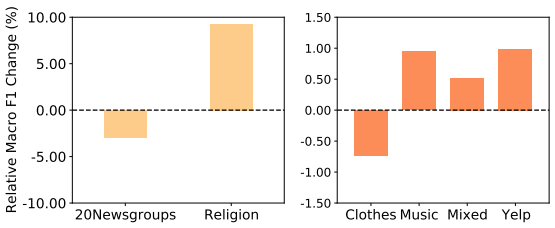

Figure 8: The relative Macro F1 changes (in %) of the CNN models for both tasks in Experiment 3.

girls are terrible' and 'of raping slave girls', and these features were not yet disabled. So, we asked one annotator to disable the features using the new "brutal" policy – disabling all which involved gender words even though some of them also detected abusive words. By disabling 18 out of 30 features on average, the gender biases were reduced for both datasets (except FPED on Wikitoxic which stayed close to the original value). Another consequence was that we sacrificed 4% and 1% macro F1 on the Waseem and Wikitoxic datasets, respectively. This finding is consistent with (Park et al., 2018) that reducing the bias and maintaining the classification performance at the same time is very challenging.

7 Exp 3: Dataset Shift

Dataset shift is a problem where the joint distribution of inputs and outputs differs between training and test stage (Quionero-Candela et al., 2009). Many classifiers perform poorly under dataset shift because some of the learned features are inapplicable (or sometimes even harmful) to classify test documents. We hypothesize that FIND is useful for investigating the learned features and disabling the overfitting ones to increase the generalizability of the model.

7.1 Datasets

We considered two tasks in this experiment. The first task aims to classify "Christianity" vs "Atheism" documents from the **20 Newsgroups** dataset[5]. This dataset is special because it contains a lot of artifacts – tokens (e.g., person names, punctuation marks) which are not relevant, but strongly co-occur with one of the classes. For evaluation, we used the **Religion** dataset by Ribeiro et al. (2016), containing "Christianity" and "Atheism" web pages, as a target dataset. The second task is sentiment analysis. We used, as a training dataset, **Amazon Clothes**, with reviews of clothing, shoes, and jewelry products (He and McAuley, 2016), and as test sets three out-of-distribution datasets – **Amazon Music** (He and McAuley, 2016), **Amazon Mixed** (Zhang et al., 2015), and the **Yelp** dataset (which was used in Experiment 1). Amazon Music contains only reviews from the "Digital Music" product category which was found to have an extreme distribution shift from the clothes category (Hendrycks et al., 2020). Amazon Mixed compiles the reviews from various kinds of products, while Yelp focuses on restaurant reviews.

7.2 Human Feedback Collection and Usage

We collected responses from MTurk workers using the same user interfaces as in Experiment 2. Simply put, we asked the workers to select a class which was relevant to a given word cloud and checked if the majority vote agreed with the weights in $\mathbf{W}$.

7.3 Results and Discussions

For the first task, on average, 14.33 out of 30 features were disabled and the macro F1 scores of the 20Newsgroups before and after debugging are 0.853 and 0.828, respectively. The same metrics of the Religion dataset are 0.731 and 0.799. This shows that disabling irrelevant features mildly undermined the predictive performance on the in-distribution dataset, but clearly enhanced the performance on the out-of-distribution dataset (see Figure 8, left). This is especially evident for the Atheism class for which the F1 score increased around 15% absolute. We noticed from the word clouds that many prominent words for the Atheism class learned by the models are person names (e.g., Keith, Gregg, Schneider) and these are not applicable to the Religion dataset. Forcing the models to use only relevant features (detecting terms like 'atheists' and 'science'), therefore, increased the macro F1 on the Religion dataset.

Unlike 20Newsgroups, Amazon Clothes does not seem to have obvious artifacts. Still, the re-

[5] http://qwone.com/~jason/20Newsgroups/

sponses from crowd workers suggested that we disable 6 features. The disabled features were correlated to, but not the reason for, the associated class. For instance, one of the disabled features was highly activated by the pattern "my .... year old" which often appeared in positive reviews such as "my 3 year old son loves this.". However, these correlated features are not very useful for the three out-of-distribution datasets (Music, Mixed, and Yelp). Disabling them made the model focus more on the right evidence and increased the average macro F1 for the three datasets, as shown in Figure 8 (right). Nonetheless, the performance improvement here was not as apparent as in the previous task because, even without feature disabling, the majority of the features are relevant to the task and can lead the model to the correct predictions in most cases.[6]

8 Discussion and Conclusions

We proposed FIND, a framework which enables humans to debug deep text classifiers by disabling irrelevant or harmful features. Using the proposed framework on CNN text classifiers, we found that *(i)* word clouds generated by running LRP on the training data accurately revealed the behaviors of CNN features, *(ii)* some of the learned features might be more useful to the task than the others and *(iii)* disabling the irrelevant or harmful features could improve the model predictive performance and reduce unintended biases in the model.

8.1 Generalization to Other Models

In order to generalize the framework beyond CNNs, there are two questions to consider. First, what is an effective way to understand each feature? We exemplified this with two word clouds representing each BiLSTM feature in Appendix C, and we plan to experiment with advanced visualizations such as LSTMVis (Strobelt et al., 2018) in the future. Second, can we make the model features more interpretable? For example, using ReLU as activation functions in LSTM cells (instead of tanh) renders the features non-negative. So, they can be summarized using one word cloud which is more practical for debugging.

In general, the principle of FIND is understanding the features and then disabling the irrelevant ones. The process makes *visualizations* and *interpretability* more actionable. Over the past few years, we have seen rapid growth of scientific research in both topics (visualizations and interpretability) aiming to understand many emerging advanced models including the popular transformer-based models (Jo and Myaeng, 2020; Voita et al., 2019; Hoover et al., 2020). We believe that our work will inspire other researchers to foster advances in both topics towards the more tangible goal of model debugging.

8.2 Generalization to Other Tasks

FIND is suitable for any text classification tasks where a model might learn irrelevant or harmful features during training. It is also convenient to use since only the trained model and the training data are required as input. Moreover, it can address many problems simultaneously such as removing religious and racial bias together with gender bias even if we might not be aware of such problems before using FIND. In general cases, FIND is at least useful for model verification.

For future work, it would be interesting to extend FIND to other NLP tasks, e.g., question answering and natural language inference. This will require some modifications to understand how the features capture relationships between two input texts.

8.3 Limitations

Nevertheless, FIND has some limitations. First, the word clouds may reveal sensitive contents in the training data to human debuggers. Second, the more hidden features the model has, the more human effort FIND needs for debugging. For instance, BERT-base (Devlin et al., 2019) has 768 features (before the final dense layer) which require lots of human effort to perform investigation. In this case, it would be more efficient to use FIND to disable attention heads rather than individual features (Voita et al., 2019). Third, it is possible that one feature detects several patterns (Jacovi et al., 2018) and it will be difficult to disable the feature if some of the detected patterns are useful while the others are harmful. Hence, FIND would be more effective when used together with disentangled text representations (Cheng et al., 2020).

Acknowledgments

We would like to thank Nontawat Charoenphakdee and anonymous reviewers for helpful comments. Also, the first author wishes to thank the support from Anandamahidol Foundation, Thailand.

[6] See Appendix F for the full results from all experiments.

References

Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. 2018. innvestigate neural networks! *arXiv preprint arXiv:1808.04260*.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany. Association for Computational Linguistics.

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*, pages 159–168, Copenhagen, Denmark. Association for Computational Linguistics.

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):1–46.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913, Brussels, Belgium. Association for Computational Linguistics.

Pengyu Cheng, Martin Renqiang Min, Dinghan Shen, Christopher Malon, Yizhe Zhang, Yitong Li, and Lawrence Carin. 2020. Improving disentangled text representation learning with information-theoretic guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7530–7541, Online. Association for Computational Linguistics.

Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 120–128, New York, NY, USA. Association for Computing Machinery.

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

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90, Vancouver, BC, Canada. Association for Computational Linguistics.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 266–274, Baltimore, Maryland, USA. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 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)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Association for Computational Linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65, Brussels, Belgium. Association for Computational Linguistics.

Ayush Jaiswal, Daniel Moyer, Greg Ver Steeg, Wael AbdAlmageed, and Premkumar Natarajan. 2019. Invariant representations through adversarial forgetting. *arXiv preprint arXiv:1911.04060*.

Jae-young Jo and Sung-Hyon Myaeng. 2020. Roles and utilization of attention heads in transformer-based neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3404–3417, Online. Association for Computational Linguistics.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112, Denver, Colorado. Association for Computational Linguistics.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Piyawat Lertvittayakumjorn and Francesca Toni. 2019. Human-grounded evaluations of explanation methods for text classification. 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)*, pages 5195–5205, Hong Kong, China. Association for Computational Linguistics.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 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*, pages 681–691, San Diego, California. Association for Computational Linguistics.

Frederick Liu and Besim Avci. 2019. Incorporating priors with feature attribution on text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6274–6283, Florence, Italy. Association for Computational Linguistics.

Hui Liu, Qingyu Yin, and William Yang Wang. 2019. Towards explainable NLP: A generative explanation framework for text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5570–5581, Florence, Italy. Association for Computational Linguistics.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. 2019. Layer-wise relevance propagation: an overview. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209. Springer.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018. Reducing gender bias in abusive language detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2799–2804, Brussels, Belgium. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. 2009. *Dataset Shift in Machine Learning*. The MIT Press.

Marco Tulio 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*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how BERT works. *CoRR*, abs/2002.12327.

Burr Settles. 2011. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478, Edinburgh, Scotland, UK. Association for Computational Linguistics.

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*, pages 3145–3153. JMLR. org.

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2018. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676.

Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies*, 67(8):639–662.

Stefano Teso and Kristian Kersting. 2019. Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 239–245.

Nithum Thain, Lucas Dixon, and Ellery Wulczyn. 2017. Wikipedia talk labels: Toxicity.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of Abusive Language: the Problem of Biased Datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 602–608, Minneapolis, Minnesota. Association for Computational Linguistics.

Tongshuang Wu, Daniel S. Weld, and Jeffrey Heer. 2019. Local decision pitfalls in interactive machine learning: An investigation into feature selection in sentiment analysis. *ACM Trans. Comput.-Hum. Interact.*, 26(4).

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267, Rochester, New York. Association for Computational Linguistics.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1031–1040, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

A Layer-wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) is a technique for explaining predictions of neural networks in terms of importance scores of input features (Bach et al., 2015). Originally, it was devised to explain predictions of image classifiers by creating a heatmap on the input image highlighting pixels that are important for the classification. Then Arras et al. (2016) and Arras et al. (2017) extended LRP to work on CNNs and RNNs for text classification, respectively.

Consider a neuron k whose value is computed using n neurons in the previous layer,

$$x_k = g(\sum_{j=1}^{n} x_j w_{jk} + b_k)$$

where x_k is the value of the neuron k, g is a non-linear activation function, w_{jk} and b_k are weights and bias in the network, respectively. We can see that the contribution of a single node j to the value of the node k is

$$z_{jk} = x_j w_{jk} + \frac{b_k}{n}$$

assuming that the bias term b_k is distributed equally to the n neurons. LRP works by propagating the activation of a neuron of interest back through the previous layers in the network proportionally. We call the value each neuron receives a relevance score (R) of the neuron. To back propagate, if the relevance score of the neuron k is R_k, the relevance score that the neuron j receives from the neuron k is

$$R_{j \leftarrow k} = \frac{z_{jk}}{\sum_{j'=1}^{n} z_{j'k}} R_k$$

To make the relevance propagation more stable, we add a small positive number ϵ (as a stabilizer) to the denominator of the propagation rule:

$$R_{j \leftarrow k} = \frac{z_{jk}}{\epsilon + \sum_{j'=1}^{n} z_{j'k}} R_k$$

We used this propagation rule, so called LRP-ϵ, in the experiments of this paper. For more details about LRP propagation rules, please see Montavon et al. (2019).

To explain a prediction of a CNN text classifier, we propagate an activation value of the output node back to the word embedding matrix. After that, the relevance score of an input word equals the sum of relevance scores each dimension of its word vector receives. However, in this paper, we want to analyze the hidden features rather than the output, so we start back propagating from the hidden features instead to capture patterns of input words which highly activate the features.

B Multiclass Classification

As shown in Figure 9, we used a slightly different user interface in Experiment 1 for the Amazon Products dataset which is a multiclass classification task. In this setting, we did not provide the options for mostly and partly relevant; otherwise, there would have been nine options per question which are too many for the participants to answer accurately. With the user interface in Figure 9, we gave a score to the feature f_i based on the participant answer. To explain, we re-scaled values in the i^{th} column of $\mathbf{W}$ to be in the range [0,1] using min-max normalization and gave the normalized value of the chosen class as a score to the feature f_i. If the participant selects None, this feature gets a zero score. The distribution of the average feature scores for this task (one CNN) is displayed in Figure 10.

Figure 9: A user interface in Experiment 1 (Amazon Products).

C Bidirectional LSTM networks

To understand BiLSTM features, we created two word clouds for each feature. The first word cloud contains top three words which gain the highest positive relevance scores from each training example, while the second word cloud does the same but for the top three words which gain the lowest negative relevance scores (see Figure 11).

Furthermore, we also conducted Experiment 1 for BiLSTMs. Each direction of the recurrent layer had 15 hidden units and the feature vector was obtained by taking element-wise max of all the hidden states (i.e., $d = 15 \times 2 = 30$). We adapted the code of (Arras et al., 2017) to run LRP on BiLSTMs. Regarding human feedback collection, we collected feedback from Amazon Mechanical Turk workers by splitting the pair of word clouds into two and asking the question about the relevant class independently of each other. The answer of the positive relevance word cloud should be consistent with the weight matrix $\mathbf{W}$, while the answer of the negative relevance word cloud should be the opposite of the weight matrix $\mathbf{W}$. The score of a BiLSTM feature is the sum of its scores from the positive word cloud and the negative word cloud.

The results of the extra BiLSTM experiments are shown in Table 4 and 5. Table 4 shows unexpected results after disabling features. For instance, disabling rank B features caused a larger performance drop than removing rank A features. This suggests that how we created word clouds for each BiLSTM feature (i.e., displaying top three words with the highest positive and lowest negative rel-

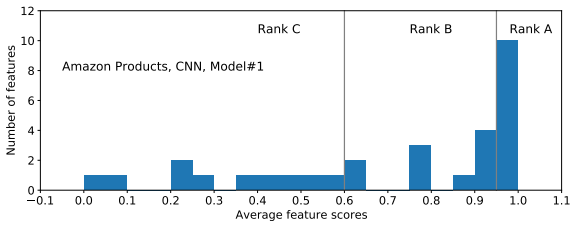

Figure 10: The distribution of average feature scores in a CNN model trained on the Amazon Products dataset.

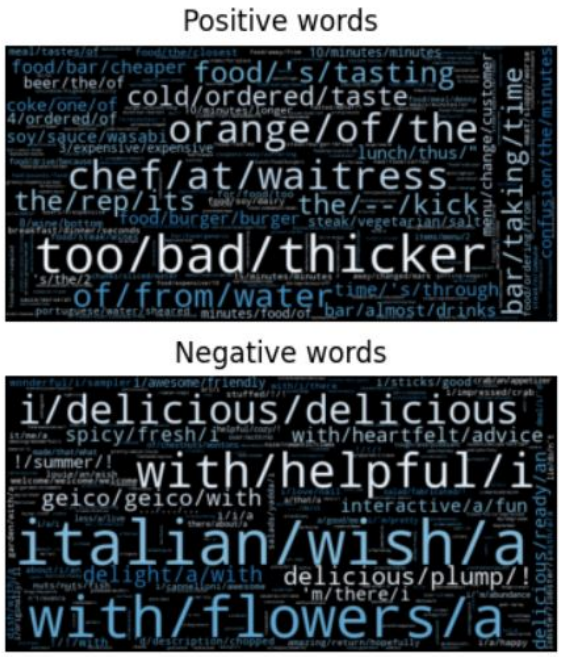

Figure 11: A pair of word clouds which represent one BiLSTM feature.

evance) might not be an accurate way to explain the feature. Nevertheless, another observation from Table 4 is that even when we disabled two-third of the BiLSTM features, the maximum macro F1 drop was less than 5%. This suggests that there is a lot of redundant information in the features of the BiLSTMs.

D Metrics for Biases

In this paper, we used two metrics to quantify biases in the models – False positive equality difference (FPED) and False negative equality difference (FNED) – with the following definitions (Dixon et al., 2018).

$$FPED = \sum_{t \in T} |FPR - FPR_t|$$

$$FNED = \sum_{t \in T} |FNR - FNR_t|$$

where T is a set of all sub-populations we consider (i.e., $T = \{\text{male}, \text{female}\}$). FPR and FNR stand for false positive rate and false negative rate, respectively. The subscript t means that we calculate the metrics using data examples mentioning the sub-population t only. We used the following keywords to identify examples which are related to or mentioning the sub-populations.

Male gender terms:

"male", "males", "boy", "boys", "man", "men", "gentleman", "gentlemen", "he", "him", "his", "himself", "brother", "son", "husband", "boyfriend", "father", "uncle", "dad"

Female gender terms:

"female", "females", "girl", "girls", "woman", "women", "lady", "ladies", "she", "her", "herself", "sister", "daughter", "wife", "girlfriend", "mother", "aunt", "mom"

E Additional Details for Reproducibility

E.1 Data Sources and Pre-processing

- **Yelp** and **Amazon Mixed**: We sampled examples from the datasets provided by Zhang et al. (2015) here[7].
- **Amazon Products**, **Amazon Clothes**, **Amazon Music**: We sampled examples from the datasets provided by He and McAuley (2016) here[8].
- **Biosbias**: We created the dataset using the code provided by De-Arteaga et al. (2019) here[9]. All the bios are from Common Crawl August 2018 Index.
- **Waseem**: The authors of (Waseem and Hovy, 2016) kindly provided the dataset to us by email. We considered "racism" and "sexism" examples as "Abusive" and "neither" examples as "Non-abusive".
- **Wikitoxic**: The dataset can be downloaded here[10]. We used only examples which were given the same label by all the annotators.
- **20Newsgroups**: We downloaded the standard splits of the dataset using scikit-learn[11]. The

[7] https://github.com/zhangxiangxiao/Crepe
[8] http://jmcauley.ucsd.edu/data/amazon/
[9] https://github.com/Microsoft/biosbias
[10] https://figshare.com/articles/Wikipedia_Talk_Labels_Toxicity/4563973
[11] https://scikit-learn.org/

header and the footer of each text were removed.

- **Religion**: We used the dataset provided by Ribeiro et al. (2016) here[12].

E.2 Number of Model Parameters

Convolutional Neural Networks

- Fixed word embeddings: 120,000,600
- Convolutional layers: 27,030
- Final (masked) dense layer:
  - Binary classification: 62 (+60)
  - 4-class classification: 124 (+120)

Bidirectional LSTM networks

- Fixed word embeddings: 120,000,600
- Bidirectional LSTM layers: 37,920
- Final (masked) dense layer:
  - Binary classification: 62 (+60)
  - 4-class classification: 124 (+120)

E.3 Computing Infrastructure Used

- CPU: Intel Core i9-9900X (3.5GHz)
- GPU: 11GB NVIDIA GeForce RTX 2080 Ti
- RAM: 32GB Corsair Vengeance DDR4

F Full Experimental Results

Tables 2-9 in this section report the full results of all the experiments and datasets. All the results shown are averaged from three runs. Boldface numbers are the best scores in the columns. They are further underlined if they are significantly better than the scores of all the other models. We conducted the statistical significance analysis using approximate randomization test with 1,000 iterations and a significance level α of 0.05 (Noreen, 1989; Graham et al., 2014).

[12] https://github.com/marcotcr/lime-experiments

Model: CNNs	Test dataset: Yelp			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	**0.758 ± 0.04**	0.666 ± 0.05	0.720 ± 0.04	0.732 ± 0.04
Disabling A	0.711 ± 0.04	0.584 ± 0.02	0.660 ± 0.03	0.676 ± 0.04
Disabling B	0.742 ± 0.03	0.618 ± 0.13	0.695 ± 0.06	0.710 ± 0.06
Disabling C	0.754 ± 0.04	**0.730 ± 0.06**	**0.742 ± 0.05**	**0.743 ± 0.04**
Disabling AB	0.681 ± 0.02	0.334 ± 0.10	0.570 ± 0.03	0.599 ± 0.04
Disabling AC	0.710 ± 0.02	0.606 ± 0.07	0.668 ± 0.04	0.678 ± 0.03
Disabling BC	0.732 ± 0.04	0.630 ± 0.14	0.694 ± 0.07	0.705 ± 0.06

Table 2: Results (Average ± SD) of Experiment 1: Yelp, CNNs

Model: CNNs	Test dataset: Amazon Products					
	Clothes F1	Music F1	Office F1	Toys F1	Accuracy	Macro F1
Original	**0.806 ± 0.02**	**0.960 ± 0.00**	0.789 ± 0.03	**0.748 ± 0.01**	**0.825 ± 0.00**	**0.829 ± 0.00**
Disabling A	0.724 ± 0.02	0.827 ± 0.06	0.722 ± 0.03	0.679 ± 0.03	0.738 ± 0.02	0.744 ± 0.02
Disabling B	0.773 ± 0.02	0.956 ± 0.00	0.711 ± 0.02	0.688 ± 0.02	0.779 ± 0.02	0.785 ± 0.02
Disabling C	0.786 ± 0.01	0.958 ± 0.01	**0.795 ± 0.02**	0.734 ± 0.02	0.817 ± 0.00	0.821 ± 0.00
Disabling AB	0.515 ± 0.08	0.586 ± 0.17	0.530 ± 0.04	0.512 ± 0.04	0.536 ± 0.05	0.556 ± 0.05
Disabling AC	0.578 ± 0.11	0.745 ± 0.05	0.652 ± 0.04	0.579 ± 0.01	0.638 ± 0.03	0.669 ± 0.01
Disabling BC	0.768 ± 0.02	0.948 ± 0.01	0.663 ± 0.06	0.627 ± 0.07	0.750 ± 0.04	0.754 ± 0.04

Table 3: Results (Average ± SD) of Experiment 1: Amazon Products, CNNs

Model: BiLSTMs	Test dataset: Yelp			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	**0.810 ± 0.01**	**0.774 ± 0.03**	**0.794 ± 0.01**	**0.799 ± 0.01**
Disabling A	**0.810 ± 0.00**	0.767 ± 0.01	0.791 ± 0.01	0.798 ± 0.00
Disabling B	0.800 ± 0.00	0.745 ± 0.01	0.776 ± 0.01	0.785 ± 0.01
Disabling C	0.803 ± 0.00	**0.774 ± 0.01**	0.790 ± 0.01	0.793 ± 0.00
Disabling AB	0.781 ± 0.01	0.720 ± 0.02	0.754 ± 0.02	0.763 ± 0.02
Disabling AC	0.800 ± 0.00	0.758 ± 0.01	0.781 ± 0.00	0.787 ± 0.00
Disabling BC	0.787 ± 0.01	0.730 ± 0.02	0.762 ± 0.01	0.769 ± 0.01

Table 4: Extra results (Average ± SD) of Experiment 1: Yelp, BiLSTMs

Model: BiLSTMs	Test dataset: Amazon Products					
	Clothes F1	Music F1	Office F1	Toys F1	Accuracy	Macro F1
Original	0.764 ± 0.01	**0.958 ± 0.00**	**0.792 ± 0.02**	**0.760 ± 0.02**	**0.818 ± 0.01**	**0.820 ± 0.01**
Disabling A	0.735 ± 0.03	0.940 ± 0.02	0.770 ± 0.02	0.733 ± 0.01	0.793 ± 0.01	0.796 ± 0.01
Disabling B	0.747 ± 0.00	0.939 ± 0.02	0.765 ± 0.02	0.741 ± 0.01	0.798 ± 0.01	0.801 ± 0.01
Disabling C	**0.769 ± 0.02**	0.946 ± 0.01	**0.792 ± 0.03**	0.759 ± 0.04	0.816 ± 0.02	0.817 ± 0.02
Disabling AB	0.636 ± 0.09	0.884 ± 0.04	0.720 ± 0.02	0.665 ± 0.04	0.727 ± 0.03	0.734 ± 0.02
Disabling AC	0.718 ± 0.02	0.828 ± 0.08	0.758 ± 0.03	0.683 ± 0.03	0.745 ± 0.04	0.754 ± 0.04
Disabling BC	0.702 ± 0.03	0.881 ± 0.05	0.702 ± 0.07	0.699 ± 0.03	0.750 ± 0.03	0.752 ± 0.03

Table 5: Extra results (Average ± SD) of Experiment 1: Amazon Products, BiLSTMs

Model: CNNs	Test dataset: Biosbias					
	Surgeon F1	Nurse F1	Accuracy	Macro F1	FPED ↓	FNED ↓
Original	**0.957 ± 0.00**	**0.943 ± 0.00**	**0.951 ± 0.00**	**0.950 ± 0.00**	0.250 ± 0.02	0.338 ± 0.02
Disabling (MTurk)	0.943 ± 0.01	0.925 ± 0.01	0.935 ± 0.01	0.934 ± 0.01	0.163 ± 0.01	0.149 ± 0.03
Disabling (One)	0.942 ± 0.01	0.924 ± 0.01	0.934 ± 0.01	0.933 ± 0.01	**0.118 ± 0.00**	**0.085 ± 0.01**

Table 6: Results (Average ± SD) of Experiment 2: Biosbias, CNNs

Model: CNNs	Test dataset: Waseem					
	Not Abusive F1	Abusive F1	Accuracy	Macro F1	FPED ↓	FNED ↓
Original	**0.876 ± 0.00**	**0.682 ± 0.01**	**0.821 ± 0.00**	**0.783 ± 0.00**	0.232 ± 0.03	0.212 ± 0.02
Disabling (MTurk)	0.865 ± 0.00	0.671 ± 0.01	0.808 ± 0.00	0.770 ± 0.00	0.303 ± 0.02	0.220 ± 0.04
Disabling (One)	0.856 ± 0.01	0.614 ± 0.04	0.791 ± 0.02	0.743 ± 0.02	**0.205 ± 0.03**	**0.184 ± 0.03**

Model: CNNs	Test dataset: Wikitoxic					
	Not Abusive F1	Abusive F1	Accuracy	Macro F1	FPED ↓	FNED ↓
Original	**0.973 ± 0.00**	0.179 ± 0.03	**0.948 ± 0.00**	0.601 ± 0.02	**0.052 ± 0.01**	0.164 ± 0.03
Disabling (MTurk)	0.967 ± 0.01	**0.230 ± 0.05**	0.936 ± 0.02	**0.609 ± 0.04**	0.083 ± 0.04	0.181 ± 0.05
Disabling (One)	0.970 ± 0.00	0.191 ± 0.01	0.942 ± 0.01	0.598 ± 0.01	0.053 ± 0.00	**0.112 ± 0.02**

Table 7: Results (Average ± SD) of Experiment 2: Waseem & Wikitoxic, CNNs

Model: CNNs	Test dataset: 20Newsgroups			
	Atheism F1	Christian F1	Accuracy	Macro F1
Original	**0.828 ± 0.01**	**0.875 ± 0.01**	**0.855 ± 0.01**	**0.853 ± 0.01**
Disabling (MTurk)	0.798 ± 0.01	0.853 ± 0.01	0.830 ± 0.01	0.828 ± 0.01

Model: CNNs	Test dataset: Religion			
	Atheism F1	Christian F1	Accuracy	Macro F1
Original	0.567 ± 0.03	0.787 ± 0.01	0.715 ± 0.02	0.731 ± 0.01
Disabling (MTurk)	**0.700 ± 0.15**	**0.834 ± 0.04**	**0.789 ± 0.07**	**0.799 ± 0.06**

Table 8: Results (Average ± SD) of Experiment 3: 20Newsgroups & Religion, CNNs

Model: CNNs	Test dataset: Amazon Clothes			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	**0.862 ± 0.01**	**0.862 ± 0.01**	**0.862 ± 0.01**	**0.862 ± 0.01**
Disabling (MTurk)	0.857 ± 0.01	0.855 ± 0.01	0.856 ± 0.01	0.856 ± 0.01

Model: CNNs	Test dataset: Amazon Music			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	0.640 ± 0.02	**0.722 ± 0.01**	0.687 ± 0.01	0.695 ± 0.01
Disabling (MTurk)	**0.668 ± 0.01**	**0.722 ± 0.01**	**0.697 ± 0.01**	**0.701 ± 0.01**

Model: CNNs	Test dataset: Amazon Mixed			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	0.784 ± 0.01	0.799 ± 0.00	0.792 ± 0.01	0.793 ± 0.00
Disabling (MTurk)	**0.793 ± 0.00**	**0.801 ± 0.00**	**0.797 ± 0.00**	**0.797 ± 0.00**

Model: CNNs	Test dataset: Yelp			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	0.767 ± 0.02	0.800 ± 0.00	0.785 ± 0.01	0.789 ± 0.01
Disabling (MTurk)	**0.786 ± 0.00**	**0.804 ± 0.00**	**0.795 ± 0.00**	**0.796 ± 0.00**

Table 9: Results (Average ± SD) of Experiment 3: Sentiment Analysis (Amazon Clothes), CNNs