Automated Debugging of Deep Text Classifiers

FIRST AUTHOR, First Affiliation, USA
SECOND AUTHOR, Second Affiliation, USA
THIRD AUTHOR, Third Affiliation, USA
FOURTH AUTHOR, Fourth Affiliation, USA

Deep Learning models are heavily reliant on large datasets. Often, it is very difficult to get a large, yet perfect dataset. Many datasets might have biases against certain sub populations or they might not work in real-world classification problems because of over-fitting. The model inherits these problems when such datasets are used. To overcome these problems, certain models have been proposed that incorporate human feedback to reduce irrelevant features. But human feedback is very energy and time consuming. In this paper, we propose a model which replaces the human feedback process and automatically reduces the irrelevant features. Experimental results show that for most of the cases our automated model performs almost as good as the state-of-the-art models which use human feedback and in certain cases our model outperforms these models.

CCS Concepts: • **Natural Language Processing**; • **Text Classification**; • **Interpretability**; • **Feature Reduction**;

Additional Key Words and Phrases: Artificial Intelligence, Deep Learning, Datasets, Human Dependency Reduction

ACM Reference Format:
First Author, Second Author, Third Author, and Fourth Author. 2022. Automated Debugging of Deep Text Classifiers. *ACM Trans. Graph.* 37, 4, Article 111 (August 2022), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep learning is all the rage right now since 2012 when Alex Krizhevsky came up with AlexNet, a deep learning model, and won the ImageNet challenge. The accuracy was 84% while the previous state-of-the-art model's accuracy was 75%. But deep learning models have one crucial drawback which is that it requires a large labeled dataset. Large dataset are hard to come by. And even if we get hold on some of these datasets, they still have a lot of problems. Many of these datasets have biases in them. For example, [Rudinger et al. 2018] have shown that many datasets have implicit gender bias in them. Another problem with datasets is overfitting, that is the model cannot generalize in the real world [Domingos 2012] because of the quirks of the training dataset. Overfitting is also a problem for the task of transfer learning.

Authors' addresses: First Author, first.author@example.com, First Affiliation, USA; Second Author, second.author@example.com, Second Affiliation, USA; Third Author, third.author@example.com, Third Affiliation, USA; Fourth Author, fourth.author@example.com, Fourth Affiliation, USA.

0730-0301/2022/8-ART111 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

There have been various ways to overcome these problems. [Park et al. 2018] tried to reduce the gender bias in abusive language detection datasets by three ways. They used debiased word embedding. They augmented the data by swapping words of one particular gender by words indicating the opposite gender. They also used transfer learning by using less biased dataset for training. [Liu and Avci 2019] also tried to reduce bias in two ways. First, they neutralized identity terms. Second, they made the deep learning model focus on toxic terms. Researchers have looked into many possible solutions for overfitting. [Bishop 1995] reduced overfitting by reducing the network structure and parameters. [Krogh and Hertz 1991] used regulariztion, which means adding cost to the loss function for large weights. [Srivastava et al. 2014] used dropout layers or randomly dropped units to reduce overfittitng. [Wan et al. 2013] introduced DropConnect, a generalization of dropout. Another way to solve these problems is to use human input during the training. [Teso and Kersting 2019] has shown that humans trust machines that are interactive and understandable. They have also shown that making a model interactive and understandable can improve its predictive and explanatory powers. [Stumpf et al. 2009] tried to make interactive deep learning models by conducting various experiments.

But these interactive models have a major problem. As [Wu et al. 2019] have shown that adding human feedback into a machine learning model can actually decrease the model's performance because of humans do not take into account the proper context while giving feedback. This results in overfitting the model. [Lertvittayakumjorn et al. 2020] tried to overcome this by adding the human feedback not during the training phase but after the training phase. They created word clouds from each feature after the model's training, showed that to a human, asked him/her if the feature is relevant or not, and from his/her feedback, the feature was enabled or disabled. Unfortunately, using human in the process could be extremely exhaustive and time consuming.

In this paper, we propose a framework which would automate the whole human feedback process. This framework exploits layer-wise relevance propagation, a method which helps to understand predictions of complex models in terms of input variables [Arras et al. 2016]. Then it aggregates all the information and predicts the features that are less relevant and disables those features. The main difference between our work and the existing works is that existing work uses human feedback to debug the model whereas we automate the whole process which saves energy and time.

We have conducted 13 experiments divided into three parts. We have conducted these experiments on 10 different datasets. The first part is a Bias Reduction. The second part is Transfer Learning. The third part is Text Classification. For all of these experiments, we

have got comparable and sometimes better results than the state-of-the-art frameworks. The main contributions of this paper are as follows:

- We try to figure out how human mind works when a human observes a word cloud and makes decision based on it. We come up with a hypothesis which replicates the same type of outcome for classification tasks.
- Instead of using human feedback we automate the whole process to save human energy, time and cost and get comparable and sometimes better results than using humans.
- We conduct 13 experiments on 10 different datasets that demonstrate the usefulness of our framework.

The rest of this paper is organized as follows. Section 2 explains related work about analyzing, explaining and debugging text-classifiers. Section 3 proposes Automated Debugging of Deep Text Classifiers", our framework. Section 4 explains the experimental setup followed by Sections 5-7 which explains our 13 different experiments divided into three parts. Section 8 discusses generalization of the framework and concludes the paper. Code and datasets of this paper are available at *https://github.com/shamirtowsif/DeepTextClassifiers*.

2 RELATED WORK

2.1 Analyzing deep NLP models

There is an extensive research on interpreting deep learning models for NLP datasets. For years, researchers have been trying to figure out how deep learning models actually work. For example, it is not clear to us how the deep learning models achieve compositionality. Inspired by research done in Computer Vision, [Li et al. 2016] had come up with various visualizing techniques to interpret the deep learning models. In doing so, they had shown that the reason of Long short-term memory (LSTM) being so successful was because it was able to maintain a very sharp focus on very important keywords. It is also very good at capturing negative asymmetry. [Karpathy et al. 2015] also did some research in trying to interpret deep learning models. They used character-level language models to provide an analysis of Deep learning models' representations, predictions, error types, and showed that LSTMs perform better on characters that require long-range reasoning. They show that LSTMs have interpretable cells that monitor long range dependencies such as line lengths, quotes and brackets. [Jacovi et al. 2018] has focused on CNN instead of LSTM. Convolutional Neural Network (CNN) was mainly used for image datasets. A lot of research had been done to find out the effects of the filters of CNN on images. But not much was known about the effects of these filters on discrete sequences. [Jacovi et al. 2018] have shown that filters capture several semantic classes of n-grams and global max-pooling induces the most important n-grams. BERT, which stands for Bidirectional Encoder Representation from Transformers, provides representations obtained from Transformers [Devlin et al. 2019]. According to [Lertvittayakumjorn et al. 2020], BERT's middle layers capture syntactic information while final layers capture task specific information. From these studies it is clear that each feature before the final output layer contains the most crucial information for the classification task.

2.2 Explaining predictions from text classifiers

There are many works that have tried to explain the predictions of text classifiers. [Ribeiro et al. 2016] proposed Local Interpretable Model-agnostic Explanations (LIME) technique which explains the predictions by learning an interpretable model locally around the prediction. They also proposed a method to explain models by presenting representative individual predictions and their explanations by framing the task as a submodular optimization problem. [Lei et al. 2016] has done some similar work. They have proposed a model that has a built-in generator that generates rationales that are sufficient enough for prediction. In this paper they limit themselves to extractive rationales. From this point of view, their rationales are simply subsets of the words from the input text that satisfy two key properties. First, the selected words should represent short and coherent pieces of text and, second, the selected words must be sufficient for prediction as a substitute of the original text. [Lertvittayakumjorn and Toni 2019] have come up with two different explanation methods. One is a gradient based analysis and the other is a decision tree based analysis even though core thrust of their work is evaluating different types of explanation methods. They evaluate both of their methods as well as random baselines and well-known existing methods using three evaluation tasks which they have proposed in their paper. Their results show the dissimilar qualities of the explanation methods and reveal the degree to which these methods could serve for each task. [Lundberg and Lee 2017] have noticed that even though various methods to explain a model have been proposed, there is no way to tell how these methods are related to one another and which method is preferable. Which is why they have come up with a unified framework called SHapley Additive exPlanations (SHAP). SHAP gives each feature an importance value for a particular prediction. Its novel components include 1) the identification of a new class of additive feature importance measures, and 2) theoretical results showing there is a unique solution in this class with a set of desirable properties. The new class combines six existing methods. It is significant because several recent methods in the class do not have the proposed desirable properties. They gain some insights from that unification. From those insights they presented new methods that show improved computational performance and better consistency with human intuition. [Shrikumar et al. 2017] came up with a method where they decompose a prediction by backpropagating the contributions of all the neurons to all the features of the input. This is how they give scores to different input features. They call this method DeepLIFT. This method compared the activation of each neuron to its 'reference activation'. Contribution scores were then assigned according to the difference. This method optionally gave separate consideration to positive and negative contributions. By doing that this method could also reveal dependencies which are missed by other approaches. In a single backward pass, scores could then be computed properly. They applied their method to models trained on MNIST and simulated genomic data, and showed significant advantages over gradient-based methods. The methods and frameworks that we have discussed so far were experimented on text datasets. [Bach et al. 2015] proposed a model that works on image classification dataset. Their proposed model decomposes nonlinear classifiers pixel-wise. They introduced a methodology

that allows to visualize the contributions of single pixels to predictions for kernel-based classifiers (over Bag of Words features) and multilayered neural networks. Heatmaps can be visualized from these pixels. A human expert is then provided with these heatmaps. The validity of the classification decision can be intuitively verified by the expert. The expert can also focus on regions of potential interest. [Bach et al. 2015] evaluated their method for classifiers trained on PASCAL VOC 2009 images, synthetic image data containing geometric shapes, the MNIST handwritten digits data set and for the pre-trained ImageNet model available as part of the Caffe open source package. They call their model Layer-Wise Relevance Propagation (LRP). [Arras et al. 2016] use LRP but on text datasets for text classification. More precisely, it was used by them to explain the predictions of a CNN trained on a topic categorization task. What words are relevant for a specific prediction of the CNN has been highlighted by their analysis. Their technique was compared to standard sensitivity analysis, both qualitatively and quantitatively, using a "word deleting" perturbation experiment, a PCA analysis, and various visualizations. All of their experiments validate the suitability of LRP for explaining the CNN predictions, which is also in line with results reported in recent image classification studies. [Lertvittayakumjorn et al. 2020] also use LRP and human feedback in their framework to improve their models. In our proposed framework, we replace the human feedback and automate the process. In particular, we come up with a module to compute how relevant a particular input feature is to an output prediction.

2.3 Debugging text classifiers using human feedback

There have been many research works that use human feedback on classifiers. One of the most notable one is [Stumpf et al. 2009]. They tried to make deep learning models interactive by conducting three experiments. First, they tried to find out whether users would like to give feedback or not. Then they tried to figure out what type of feedback the users would like to provide. After that they pondered whether these feedback can be added to the model or not. Finally, they wanted to figure out what would be the impact of such feedback. Taken together, the results of their experiments show that supporting rich interactions between users and machine learning systems is feasible for both user and machine. It was also shown that the potential of rich human computer collaboration via on-the-spot interactions was a promising direction for machine learning systems and users to collaboratively share intelligence. [Kulesza et al. 2015] have come up with a similar model called "Explanatory Debugging". Here they make a model which explains to the users how it makes each prediction and then takes correction from users, if any. They presented the underlying principles of their approach. They also built a prototype of their model. It was shown by an empirical evaluation of their model that the model increased participants' understanding of the learning system by 52%. Their model also allowed participants to correct its mistakes up to twice as efficiently as participants using a traditional learning system. [Settles 2011] came up with a model called DUALIST that also incorporated human feedback. The model learns from labels on both features (e.g., words) and instances (e.g., documents). But all these models are extension of shallow models. [Settles 2011] came up with a new semi-supervised training algorithm developed for this setting. It was 1) fast enough to support real-time interactive speeds, and 2) at least as accurate as preexisting methods for learning with mixed feature and instance labels. It is shown in user studies that with only a few minutes of human annotators' effort, near state-of-the-art classifiers were produced on several corpora and in a variety of application domains.

[Ribeiro et al. 2016] had realized that despite widespread adoption, machine learning models had remained mostly black boxes. But understanding the reasons behind predictions was a very important task in assessing trust since it was fundamental to take action based on a prediction. It was also very fundamental in deciding when to choose whether to deploy a new model. These understandings can provide us the insights of the model. These insights can be used to transform an untrustworthy model or prediction into a trustworthy one. [Ribeiro et al. 2016] proposed a technique called LIME that tried to explain predictions to users by incorporating human feedback and later doing feature engineering which was able to improve the trustworthiness of the model. [Teso and Kersting 2019] came up with an interactive learning framework called Explanatory Interactive Learning (XIL). In their framework in each step, the query of the learner was explained to the user. The learner also explained queries of any active classifier to visualize explanations of the corresponding predictions. It was demonstrated that the learned model's predictive and explanatory powers as well as trust can be boosted by using text (e.g. SVMs) and image classification (e.g. neural networks) experiments as well as a user study.

There are some pitfalls of using human feedback as well. [Wu et al. 2019] have shown that adding human feedback into machine learning model can actually decrease the model's performance. The reason is when humans are providing the inputs they are only thinking about the local context and not the global context. [Wu et al. 2019] looked into how those trends were affected by the dataset, learning algorithm, and the training set size. Because of these factors they faced consistent generalization issues. It was suggested by their results that interactive machine learning systems with rapid iterations could be dangerous. One reason was that if they encouraged local actions divorced from global context then the overall performance of the models would be degraded. They also talked about implications of their feature selection results to the broader area of interactive machine learning systems and research. [Lertvittayakumjorn et al. 2020] tried to overcome this by adding the human feedback not during the training phase but after the training phase. But it takes a lot of human effort, time and it is also costly. Which is why we aim to automate the human feedback procedure to reduce the irrelevant features of the model.

3 AUTOMATED DEBUGGING OF DEEP TEXT CLASSIFIERS

3.1 Motivation

Deep Learning models can be divided into two parts. They are feature extraction part and classification part. Feature extraction part transforms an input into a dense vector called feature vector. The classification part puts the feature vector through a dense layer with a softmax activation. This dense layer gives out a final prediction. There are many types of deep learning models such as CNN models,

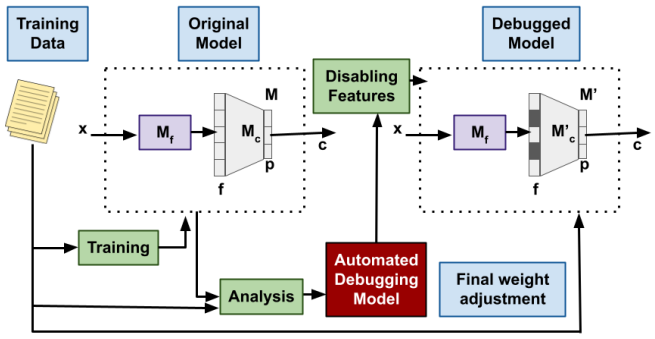

Fig. 1. Overview of the Automated Debugging Text Classifier

RNN models or Transformers. These models are not usually transparent. In other words, a user does not understand what is going on inside them. Hence a user cannot use their own knowledge and experience to improve or modify these models.

If we understand what patterns and characteristics are captured by each feature, we can disable the features which are irrelevant for the classification part. [Lertvittayakumjorn et al. 2020] used LRP to produce word clouds for each feature, which a user can look at and then decide whether that feature is relevant or not. If the feature is not relevant then the user disables it. Since humans are used, this whole process is extremely time and energy consuming. In this paper, we come up with a framework that automates this whole process of human feedback. Figure 1 shows the overview of our debugging framework. As we can see at first we have the original model and the training data. After the model has been trained, we analyze it. That analysis is used by the Automated Debugging Module. It is represented by the red box. This module disables certain features and improves the model. This improved model is the final Debugged Model.

3.2 Notation

Let us assume that there is a text classification task with C classes and $\mathcal{V}$ vocabulary. The dataset that this task uses is $\mathcal{D}$.
$\mathcal{D} = (x_1, y_1), ..., (x_N, y_N)$ where N is the number of documents in the dataset. x_i is the i-th document which contains L words. So we can write $x_i = [x_{i1}, x_{i2}, ..., x_{iL}]$ and y_i is the i-th label that belongs to C. $\mathcal{M}$ is a classifier. If we put x_i document into the classifier $\mathcal{M}$ then the classifier classifies the document into y_i which belongs to C. So we can write, $\mathcal{M}(x) = y$. $\mathcal{M}$ is divided into two parts. They are $\mathcal{M}_f$, feature extraction part and $\mathcal{M}_c$ the classification part. So we can also write $\mathcal{M}(x) = (\mathcal{M}_f \circ \mathcal{M}_c)(x)$. $\mathcal{M}_f$ classifies the input x into a feature vector f, $\mathcal{M}_f(x) = f$. The classification part $\mathcal{M}_c$ classifies the feature vector f into a probability vector p, $\mathcal{M}_c(f) = \text{softmax}(Wf + b) = p$, where $p \in [0, 1]^{|C|}$. Here f is the feature vector, $f = [f_1, f_2, ..., f_d] \in R^d$. d is the dimention of the f. W is the weight matrix where $W \in R^{|C| \times d}$ and b is bias where $b \in R^{|C|}$.

3.3 Understanding the Model

In [Lertvittayakumjorn et al. 2020], they create word clouds for each feature and show it to the users. They also show what class the feature should support. They get it from the weight matrix W. The size of the n-grams in the word cloud is decided from the score they get from the Layerwise Relevance Propagation (LRP). LRP analyzes the patterns of input that activate features to understand the model. LRP operates by propagating the prediction backward in the neural network, using a set of purposely designed propagation rules. The score signifies the relevance of the n-grams to the output. The user looks into the word cloud and decides whether the feature is relevant to the output or not. The proposed approach uses the hypothesis that when a user looks into the word cloud, he/she notices the larger words and tries to make a correlation between the larger words and the correct class. We try to automate this human process and test the hypothesis in our framework.

Algorithm 1 is used to compute the feature score. Suppose we have a document, x_i and f_j is one of those features that we extract from that document. LRP gives us a relevance score, r_{ij}, from that feature. There might be many n-grams with one particular relevance score. n-grams with similar relevance scores are very similar themselves. There is no point in taking all of them. So we take the first n-gram and count the number of n-grams with that particular relevance score. Suppose for the document, x_i, and the feature, f_j, we get some n-grams. For each n-gram we get the count c_{ij}. What class the feature should support is known from the weight, W. Suppose that class is C. For each n-gram in the feature, we compute the similarity of that n-gram with C. To calculate the similarity between an n-gram and the class, C, we use soft cosine measure.

Soft cosine similarity takes the semantics into consideration [Sidorov et al. 2014]. Here we convert the words into vectors and compute the cosine similarity of the vectors. We do that by using the following procedure: We preprocess the n-gram and the text label of the class, C. We build a dictionary and a TF-IDF model using the n-gram and the text label of the class. We load a 300 dimensional wiki glove embedding to build a sparse similarity matrix. Using this sparse similarity matrix and soft cosine measure, we get the similarity score. This way, even if there is no common word between the n-gram and the class, C, we would still get a meaningful similarity score. Let the vector of the n-gram belonging to document x_i and the feature f_j be A and the vector of the text label of the class, C, be B. We have to compute the soft cosine similarity of A and B. Suppose the value we get is s_{ij}. So we can write $s_{ij} = \text{soft\_cosine}(A, B)$.

Let the size of vectors A and B be n. So we can write A as $A = A_1, A_2, ..., A_n$ and B as $B = B_1, B_2, ..., B_n$. Here A_i and B_i refers to each feature of the original vector. Then the soft cosine similarity of these two vectors, A and B can be computed by the following equation:

$$\text{soft\_cosine}(A,B) = \frac{\sum_i^n \sum_j^n A_i B_j \text{cosine}(A_i, B_j)}{\sqrt{\sum_i^n \sum_j^n A_i A_j \text{cosine}(A_i, A_j)} . \sqrt{\sum_i^n \sum_j^n B_i B_j \text{cosine}(B_i, B_j)}} \tag{1}$$

$\text{cosine}(A_i, B_j)$ is the cosine similarity of two features, A_i and B_j, of the vectors A and B, respectively. If A_i is a feature of A and the size of A_i is m then we can write feature A_i as $A_i = A_{i1}, A_{i2}, ..., A_{im}$.

Then the equation to compute the cosine similarity of two features, A_i and B_j, is given below.

$$\text{cosine}(A_i, B_j) = \frac{\sum_{k=1}^{m} A_{ik} B_{jk}}{\sqrt{\sum_{k=1}^{m} A_{ik}^2} \sqrt{\sum_{k=1}^{m} B_{jk}^2}} \tag{2}$$

Algorithm 1 A pseudocode to compute the feature score

```
Require: class_name, features, documents
  fs ← [ ]                                              ▷ feature scores
  for each f ∈ features do
      rs ← [ ]                                          ▷ relevance scores
      cs ← [ ]                                          ▷ counts
      ss ← [ ]                                          ▷ similarity scores
      for each d ∈ documents do
          for each n ∈ d[f] do                          ▷ ngram
              r ← LRP(n)                                ▷ relevance score
              c ← LRP(n)                                ▷ count
              s ← soft_cosine(n, class_name)            ▷ similarity score
              rs.insert(r)
              cs.insert(c)
              ss.insert(s)
          end for
      end for
      mrs ← max(rs)                                     ▷ max relevance score
      mc ← max(cs)                                      ▷ max count
      mss ← max(ss)                                     ▷ max similarity score
      for each r ∈ rs and c ∈ cs and s ∈ ss do
          feature_score ← r/mrs + c/mc + s/mss
          fs.insert(feature_score)
      end for
      final_fs ← sum(fs)                                ▷ final feature score
  end for
```

We get relevance scores, similarity scores, and counts for all the d features and N documents. We normalize these scores. For a particular document, x_i, and a particular feature, f_i, we add their relevance score, similarity score and count to get the feature score, $\mathcal{F}_{ij}$. For a particular feature, f_j, and all the N documents, we add their feature scores to get the summed feature score, $\mathcal{F}_i = \sum_{j=1}^{N} \mathcal{F}_{ij}$. So for d number of features we should get d number of summed feature scores. Table 1 explains the process more clearly.

3.4 Disabling Features

After training we select the most valid and relevant features of the class. For d features we get d feature scores. We identify features with the top scores. We test the model on the validation dataset after disabling different number of features. Then we select the number of features for which we got the highest number of validation accuracy. If we get the highest validation score for m number of features then we disable $d - m$ number of features. We start disabling features which has the lowest feature score and move upwards incrementally. We disable the rest features. After disabling the features the classification part of the model $\mathcal{M}_c$ becomes $\mathcal{M}'_c$. So we can rewrite equation to classify the feature vector into a probability vector as $\mathcal{M}'_c(f) = softmax((W \odot Q)f + b) = p$. Here $Q \in R^{|C| \times d}$ is a masking matrix. $\odot$ is an element wise multiplication operator. Initially Q is all one. When we disable features we turn respective values in Q to zero. Then we train the debugged model.

4 EXPERIMENTAL SETUP

Table 2 lists all the 10 datasets and their splits used in our experiments. On these datasets, we run 13 different experiments. For each experiments, we run the model three times with different random seeds. We average the results and report them. We run our experiments on two different deep learning models, one dimensional(1D) CNN and BiLSTM. The 1D CNN uses filter sizes of [2, 3, 4]. For each size, we use 10 filters. For every layer, we use ReLU as an activation layer except for the output layer where we use softmax to get a probability vector. We trim or pad each example so that there are 150 words. For each of these words, we use 300 dimensional glove vectors as embedding. We also use Keras trained with Adam optimizers to implement the models. We use innvestigate library [Alber et al. 2019] to compute the relevance scores. This library uses $LRP - \epsilon$ to stabilize the relevance scores.

5 BIAS REDUCTION

There are certain datasets which contain a lot of biases in them. If a classifier is trained with such a dataset then features of that classifier will also capture the biases. We test our framework on these datasets to see whether it can automatically find out such features and disable them.

5.1 Datasets and Metrics

We use **Biosbias**, **Waseem** and **Wikitoxic** dataset for this task. [De-Arteaga et al. 2019] have studied gender bias in occupation classification task. To do so they have used **Biosbias** dataset, where the task is about predicting one's occupation given one's bio paragraph. The occupation can be nurse or surgeon. Because more women are nurse and more men are surgeon, any classifier usually exploits gender terms to classify whether a bio paragraph belongs to a nurse or a surgeon. Which means that a lot of the times male nurses and female surgeons get misclassified. [Waseem and Hovy 2016] using ideas from critical race theory label a publicly available dataset of 16k tweets which is called **Waseem**. They analyze the impact of various extra-linguistic features on the dataset in conjunction with character n-grams for hate speech detection. The task of hate speech detection classifies whether a text is abusive or not. [Park et al. 2018] has shown that this dataset is biased towards women. That is if the tweet is related to women then the classifier classifies it as abusive. We have also trained our models on the Waseem dataset and tested them on another dataset called Wikitoxic. [Thain et al. 2017] used a dataset called **Wikitoxic** which includes over 100k labeled discussion comments from English Wikipedia. We try to predict whether these comments are toxic/abusive or not. Our goal to to reduce the bias of the trained models. [Dixon et al. 2018] came up with a new approach to measuring and mitigating unintended bias in machine learning models. They came up with two metrics 1) FPED - false

	f_1	f_2	...	f_d
x_1	$\mathcal{F}_{11} = r_{11} + c_{11} + s_{11}$	$\mathcal{F}_{12} = r_{12} + c_{12} + s_{12}$	...	$\mathcal{F}_{1j} = r_{1d} + c_{1d} + s_{1d}$
x_2	$\mathcal{F}_{21} = r_{21} + c_{21} + s_{21}$	$\mathcal{F}_{22} = r_{22} + c_{22} + s_{22}$	...	$\mathcal{F}_{2j} = r_{2d} + c_{2d} + s_{2d}$
...	...	...	...	...
x_N	$\mathcal{F}_{N1} = r_{N1} + c_{N1} + s_{N1}$	$\mathcal{F}_{N2} = r_{N2} + c_{N2} + s_{N2}$	...	$\mathcal{F}_{Nj} = r_{Nd} + c_{Nd} + s_{Nd}$
	$\sum_{i=1}^{N} \mathcal{F}_{i1} = \mathcal{F}_1$	$\sum_{i=1}^{N} \mathcal{F}_{i2} = \mathcal{F}_2$	...	$\sum_{i=1}^{N} \mathcal{F}_{id} = \mathcal{F}_d$

Table 1. Computation of feature score for each document and each feature

Exp	Dataset	$\|C\|$	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 2. Dataset used in the experiments

positive equality difference and 2) FNED - false negative equality difference. The lower the value of these metrics the better the model.

5.2 Results and Discussions

Our results and the number of features used are listed in Table 3, Table 4 and Table 5. As we can see all our accuracies are better than [Lertvittayakumjorn et al. 2020]. For FPED and FNED metrics, the lower the values the better. As we can see except for Biosbias dataset we are getting better results. The reason we believe we do not get better results for Biosbias is because [Lertvittayakumjorn et al. 2020] use around 17 features while our framework uses around 18 features. It seems that our framework still uses some irrelevant features.

6 TRANSFER LEARNING

According to [Quiñonero-Candela et al. 2009], Dataset Shift is a task where we train a model in one dataset distribution but test the model in another one. This is to test how well the model functions in the real world.

6.1 Datasets

At first we used a dataset called **20 Newsgroups**. This dataset is great because it contains a lot of artifacts such as a person's names, punctuation marks etc which has very little relevance to the actual label but co-occur with them. We used this dataset to train and test as well as just to train but test with another dataset. The dataset that we used to test was used in [Ribeiro et al. 2016]. To explain predictions of any classifier in a very lucid way [Ribeiro et al. 2016] used a dataset called **Religion**. [He and McAuley 2016] in their paper, built a model for the One-Class Collaborative Filtering setting to estimate users' fashion-aware personalized ranking functions based on their feedback. They use many datasets. One of them is **Amazon Clothes**. We use this dataset to train and test. We also use this just to train but test on other datasets such as **Amazon Music** [He and McAuley 2016], **Amazon Mixed** [Zhang et al. 2015] and **Yelp**.

6.2 Results and Discussions

Our results and the number of features used are given in Table 6, Table 7, Table 8, Table 9, Table 10 and Table 11. As we see in all of the cases, our results are very close to those of [Lertvittayakumjorn et al. 2020]. And in two cases, 20Newsgroups and Amazon Clothes, we get better results.

7 TEXT CLASSIFICATION

We experiment our framework on datasets where we automate the process of selecting the relevant features while disabling the irrelevant features. We train the model on various datasets. We evaluate the quality of the features and disable the irrelevant features. We compute the F1 score and the accuracy.

7.1 Datasets

[Zhang et al. 2015] has constructed several large scale datasets to show that character-level convolutional networks could achieve state-of-the-art or competitive results. One of those large scale datasets is **Yelp** (which we used in Expermient 2) which predicts sentiments of restaurants reviews. The sentiments are positive and negative. This dataset has 500 samples for training. [He and McAuley 2016] created novel models to estimate users' fashion-aware personalized ranking functions based on their past feedback. To evaluate their models they used a dataset called **Amazon Products**. This dataset is used to classify product reviews into one of the four categories (Clothing Shoes and Jewelry, Digital Music, Office Products or Toys, and Games). This dataset has 100 samples for training.

7.2 Results and Discussions

First we use 1D CNN model into our framework. We train and test the model with the Yelp dataset and Amazon Products dataset. Our results and the number of features used can be seen in Table 14 and Table 12. For both the Yelp dataset and the Amazon Products dataset, our accuracy is better than the accuracy of [Lertvittayakumjorn et al. 2020]. Then we use BiLSTM model in our framework. We also train and test the model with the Yelp dataset and Amazon Products dataset. Our new results can be seen in Table 15 and Table 13. For the Yelp dataset our results are close to [Lertvittayakumjorn et al. 2020] and for the Amazon Products dataset our results are not so

Model: CNNs	Train and Test Dataset: Biosbias						
	Surgeon F1	Nurse F1	Accuracy	Macro F1	FPED↓	FNED↓	feature
FIND	0.943 ± 0.00	0.925 ± 0.01	0.935 ± 0.01	0.934 ± 0.01	**0.163 ± 0.01**	**0.149 ± 0.03**	20 ± 0.00
Our	**0.958 ± 0.00**	**0.945 ± 0.00**	**0.953 ± 0.00**	**0.952 ± 0.00**	0.235 ± 0.00	0.316 ± 0.03	18.333 ± 8.08

Table 3. Results (Average ± SD) of Experiment 1: Biosbias, CNNs

Model: CNNs	Train and Test Dataset: Waseem						
	Not Abusive F1	Abusive F1	Accuracy	Macro F1	FPED↓	FNED↓	feature
FIND	0.865 ± 0.00	0.671 ± 0.01	0.808 ± 0.00	0.770 ± 0.00	0.303 ± 0.02	0.220 ± 0.04	20 ± 0.00
Our	**0.876 ± 0.00**	**0.707 ± 0.00**	**0.826 ± 0.00**	**0.792 ± 0.00**	**0.273 ± 0.01**	**0.181 ± 0.01**	24.666 ± 2.88

Table 4. Results (Average ± SD) of Experiment 1: Waseem, CNNs

Model: CNNs	Train Dataset: Waseem Test Dataset: Wikitoxic						
	Not Abusive F1	Abusive F1	Accuracy	Macro F1	FPED↓	FNED↓	feature
FIND	0.967 ± 0.01	0.230 ± 0.05	0.936 ± 0.02	0.609 ± 0.04	0.083 ± 0.04	0.181 ± 0.05	20 ± 0.00
Our	**0.971 ± 0.00**	**0.272 ± 0.03**	**0.944 ± 0.01**	**0.633 ± 0.01**	**0.058 ± 0.03**	**0.178 ± 0.05**	21.333 ± 1.52

Table 5. Results (Average ± SD) of Experiment 1: Wikitoxic, CNNs

Model: CNNs	Train and Test Dataset: 20Newsgroups				
	Atheism F1	Christian F1	Accuracy	Macro F1	feature
FIND	0.798 ± 0.01	0.853 ± 0.01	0.830 ± 0.01	0.828 ± 0.01	20 ± 0.00
Our	**0.818 ± 0.03**	**0.871 ± 0.01**	**0.849 ± 0.02**	**0.849 ± 0.01**	14.7 ± 6.4

Table 6. Results (Average ± SD) of Experiment 2: 20Newsgroups, CNNs

Model: CNNs	Train Dataset: 20Newsgroups Test Dataset: Religion				
	Atheism F1	Christian F1	Accuracy	Macro F1	feature
FIND	**0.700 ± 0.15**	**0.834 ± 0.04**	**0.789 ± 0.07**	**0.799 ± 0.06**	20 ± 0.00
Our	0.651 ± 0.07	0.8 ± 0.01	0.753 ± 0.02	0.748 ± 0.02	10.7 ± 0.6

Table 7. Results (Average ± SD) of Experiment 2: Religion, CNNs

Model: CNNs	Train and Test Dataset: Amazon Clothes				
	Negative F1	Positive F1	Accuracy	Macro F1	feature
FIND	0.857 ± 0.01	0.855 ± 0.01	0.856 ± 0.01	0.856 ± 0.01	20 ± 0.00
Our	**0.867 ± 0.00**	**0.868 ± 0.00**	**0.868 ± 0.00**	**0.868 ± 0.00**	22 ± 2.00

Table 8. Results (Average ± SD) of Experiment 2: Amazon Clothes, CNNs

Model: CNNs	Train Dataset: Amazon Clothes Test Dataset: Amazon Music				
	Negative F1	Positive F1	Accuracy	Macro F1	feature
FIND	**0.688 ± 0.01**	**0.722 ± 0.01**	**0.697 ± 0.01**	**0.701 ± 0.01**	20 ± 0.00
Our	0.678 ± 0.00	0.705 ± 0.00	0.692 ± 0.00	0.693 ± 0.00	24.7 ± 0.6

Table 9. Results (Average ± SD) of Experiment 2: Amazon Music, CNNs

good. We believe these type of results are unique to BiLSTM model. But further research is needed.

8 CONCLUSION

We propose an automated debugging text classifier framework which automates the process of finding irrelevant features and then

Model: CNNs	Train Dataset: Amazon Clothes Test Dataset: Amazon Mixed				
	Negative F1	Positive F1	Accuracy	Macro F1	feature
FIND	**0.793 ± 0.00**	**0.801 ± 0.00**	**0.797 ± 0.00**	**0.797 ± 0.00**	20 ± 0.00
Our	0.791 ± 0.00	0.797 ± 0.00	0.794 ± 0.00	0.794 ± 0.00	14.3 ± 2.5

Table 10. Results (Average ± SD) of Experiment 2: Amazon Mixed, CNNs

Model: CNNs	Train Dataset: Amazon Clothes Test Dataset: Yelp				
	Negative F1	Positive F1	Accuracy	Macro F1	feature
FIND	0.786 ± 0.00	**0.804 ± 0.00**	**0.795 ± 0.00**	**0.796 ± 0.00**	20 ± 0.00
Our	**0.787 ± 0.01**	0.783 ± 0.01	0.786 ± 0.00	0.788 ± 0.00	18.7 ± 7.5

Table 11. Results (Average ± SD) of Experiment 2: Yelp, CNNs

Model: CNNs	Train and Test Dataset: Amazon Products						
	Clothes F1	Music F1	Office F1	Toys F1	Accuracy	Macro F1	feature
FIND	0.786±0.01	**0.958±0.01**	**0.795±0.02**	0.734±0.02	0.817±0.00	0.821±0.00	20±0.00
Our	**0.811±0.01**	0.952±0.00	0.784±0.02	**0.745±0.02**	**0.822±0.00**	**0.826±0.00**	10±0.00

Table 12. Results (Average ± SD) of Experiment 3: Amazon Products, CNNs

Model: BiLSTMs	Train and Test Dataset: Amazon Products						
	Clothes F1	Music F1	Office F1	Toys F1	Accuracy	Macro F1	feature
FIND	**0.769 ± 0.02**	**0.946 ± 0.01**	**0.792 ± 0.03**	**0.759 ± 0.04**	**0.816 ± 0.02**	**0.817 ± 0.02**	20 ± 0.00
Our	0.591 ± 0.17	0.840 ± 0.04	0.720 ± 0.03	0.676 ± 0.04	0.714 ± 0.05	0.730 ± 0.04	10 ± 0.00

Table 13. Results (Average ± SD) of Experiment 3: Amazon Products, BiLSTMs

Model: CNNs	Train and Test Dataset: Yelp				
	Negative F1	Positive F1	Accuracy	Macro F1	feature
FIND	0.754 ± 0.04	**0.730 ± 0.06**	0.742 ± 0.05	0.743 ± 0.04	20 ± 0.00
Our	**0.786 ± 0.01**	0.706 ± 0.01	**0.753 ± 0.01**	**0.766 ± 0.01**	16 ± 4.58

Table 14. Results (Average ± SD) of Experiment 3: Yelp, CNNs

Model: BiLSTMs	Train and Test Dataset: Yelp				
	Negative F1	Positive F1	Accuracy	Macro F1	feature
FIND	**0.803 ± 0.00**	**0.774 ± 0.01**	**0.790 ± 0.01**	**0.793 ± 0.00**	20 ± 0.0
Our	0.772 ± 0.02	0.715 ± 0.03	0.747 ± 0.01	0.756 ± 0.01	24 ± 5.3

Table 15. Results (Average ± SD) of Experiment 3: Yelp, BiLSTMs

disables them. We find that 1) it is possible to automatically find interesting properties of various deep learning models without the use of humans 2) some of the learned features are relevant while others are not 3) disabling irrelevant features can improve the model.

8.1 Generalizations to other Models

Can we make our framework generalizable to other models? We have already worked on CNN and BiLSTM. Our next goal is to make it work with LSTMVis [Strobelt et al. 2017]. Second question is: Can we make the model more interpretable? Over the past few years we have seen rapid growth in the research to make the deep learning models more interpretable [Zhang and Zhu 2018]. We believe that if better techniques are developed to make the models more interpretable the better our framework will work.

8.2 Generalizations to other Tasks

Our model has been used for text classification and to reduce gender bias. It can be used to reduce religious and racial bias as well. Our model can also be extended for question answering as well as natural language inference tasks.

REFERENCES

M. Alber, S. Lapuschkin, P. Seegerer, M. Hägele, K. Schütt, G. Montavon, W. Samek, K. Müller, S. Dähne, and P. Kindermans. 2019. iNNvestigate neural networks! *J. Mach. Learn. Res.* 20, 93 (2019), 1–8.

L. Arras, F. Horn, G. Montavon, K. Müller, and W. Samek. 2016. Explaining Predictions of Non-Linear Classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, 1–7. https://doi.org/10.18653/v1/W16-1601

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

C. M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA.

M. De-Arteaga, A. Romanov, H. Wallach, J. Chayes, C. Borgs, A. Chouldechova, S. Geyik, K. Kenthapadi, and A. T. 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*. 120–128.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

L. Dixon, J. Li, J. Sorensen, N. Thain, and L. Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 67–73.

P. Domingos. 2012. A Few Useful Things to Know about Machine Learning. *Commun. ACM* 55, 10 (Oct. 2012), 78–87. https://doi.org/10.1145/2347736.2347755

R. He and J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th WWW conference*. 507–517.

A. Jacovi, O. S. Shalom, and Y. Goldberg. 2018. Understanding Convolutional Neural Networks for Text Classification. In *BlackboxNLP@EMNLP*.

A. Karpathy, J. Johnson, and L. Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

A. Krogh and J. Hertz. 1991. A Simple Weight Decay Can Improve Generalization. In *NIPS*.

T. Kulesza, M. Burnett, W. Wong, and S. Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*. 126–137.

T. Lei, R. Barzilay, and T. Jaakkola. 2016. Rationalizing Neural Predictions. In *Proceedings of the 2016 EMNLP*. Association for Computational Linguistics, Austin, Texas, 107–117. https://doi.org/10.18653/v1/D16-1011

P. Lertvittayakumjorn, L. Specia, and F. Toni. 2020. Human-in-the-loop Debugging Deep Text Classifiers. In *EMNLP*.

P. Lertvittayakumjorn and F. Toni. 2019. Human-grounded Evaluations of Explanation Methods for Text Classification. In *Proceedings of the 2019 EMNLP)*. Association for Computational Linguistics, Hong Kong, China, 5195–5205. https://doi.org/10.18653/v1/D19-1523

J. Li, X. Chen, E. Hovy, and D. Jurafsky. 2016. Visualizing and Understanding Neural Models in NLP. In *Proceedings of the 2016 NAACL*. Association for Computational Linguistics, 681–691. https://doi.org/10.18653/v1/N16-1082

F. Liu and B. Avci. 2019. Incorporating Priors with Feature Attribution on Text Classification. In *ACL*.

S. M. Lundberg and S. Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st NIPS* (Long Beach, California, USA) *(NIPS'17)*. 4768–4777.

J. H. Park, J. Shin, and P. Fung. 2018. Reducing Gender Bias in Abusive Language Detection. In *Proceedings of the 2018 EMNLP*. 2799–2804. https://doi.org/10.18653/v1/D18-1302

J. Quiñonero-Candela, M. Sugiyama, N. D. Lawrence, and A. Schwaighofer. 2009. *Dataset shift in machine learning*. Mit Press.

M. T. Ribeiro, S. Singh, and C. Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

R. Rudinger, J. Naradowsky, B. Leonard, and B. Van Durme. 2018. Gender Bias in Coreference Resolution. In *Proceedings of the 2018 NAACL, Volume 2 (Short Papers)*. Association for Computational Linguistics, New Orleans, LA, 8–14. https://doi.org/10.18653/v1/N18-2002

B. Settles. 2011. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the 2011 EMNLP*. 1467–1478.

A. Shrikumar, P. Greenside, and A. Kundaje. 2017. Learning important features through propagating activation differences. In *International Conference on Machine Learning*. PMLR, 3145–3153.

Grigori Sidorov, Alexander Gelbukh, Helena Gómez-Adorno, and David Pinto. 2014. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas* 18, 3 (2014), 491–504.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 1929–1958.

H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. 2017. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 667–676.

S. Stumpf, V. Rajaram, L. Li, W. Wong, M. Burnett, T. Dietterich, E. Sullivan, and J. Herlocker. 2009. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies* 67, 8 (2009), 639–662. https://doi.org/10.1016/j.ijhcs.2009.03.004

S. Teso and K. Kersting. 2019. Explanatory Interactive Machine Learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (Honolulu, HI, USA) *(AIES '19)*. ACM, 239–245. https://doi.org/10.1145/3306618.3314293

N. Thain, L. Dixon, and E. Wulczyn. 2017. Wikipedia Talk Labels: Toxicity. https://doi.org/10.6084/m9.figshare.4563973.v2

L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. 2013. Regularization of Neural Networks using DropConnect. In *Proceedings of the 30th ICML (Proceedings of Machine Learning Research, Vol. 28)*, S. Dasgupta and D. McAllester (Eds.). PMLR, Atlanta, Georgia, USA, 1058–1066. https://proceedings.mlr.press/v28/wan13.html

Z. Waseem and D. Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*. 88–93.

T. Wu, D. S. Weld, and J. Heer. 2019. Local Decision Pitfalls in Interactive Machine Learning: An Investigation into Feature Selection in Sentiment Analysis. *ACM Trans. Comput.-Hum. Interact.* 26, 4, Article 24 (June 2019), 27 pages. https://doi.org/10.1145/3319616

Q. Zhang and S. Zhu. 2018. Visual interpretability for deep learning: a survey. *arXiv preprint arXiv:1802.00614* (2018).

X. Zhang, J. Zhao, and Y. LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28 (2015), 649–657.