Project Machine Learning
— Milestone 3 —

Johannes Constantin Keller, Moritz Wassmer, Ivan Akunevich

September 3, 2024

1 Introduction

During the first milestone, we focused on the architecture of BERT for predicting the toxicity task. In the second milestone, we focused on model selection and hyperparameter optimization. We also proposed a confidence measure for the predictions and provided estimates for the generalization error.

In Milestone 3, we will first give a brief overview the pipeline of the model. Afterwards, we summarize the results of the final model and provide a more in depth analysis of the predictions of the model.

2 Prediction Methodology

In this section, we are going to summarize the final pipeline of the model as well as other chosen hyperparameters.

Model For the toxicity prediction task, we utilized a BERT (*Bidirectional Encoder Representations from Transformers*)- based architecture (Devlin et al., 2019). Specifically, we were using *BERT-Base* which comprises a pipeline of an embedding layer, twelve transformer-encoders and finally a toxicity-classification layer. In short, the configuration is: $H = 768, L = 12, A = 12, S = 512$, with H as the dimensionality of the word vectors, L as the number of encoders, A as the number of attention-heads per encoder and S as the maximum token input length.

In the embedding layer the comments are transformed into a token representation. As a preprocessing step, *WordPiece-Embedding* (Wu et al., 2016) is performed. The words are split into their components to enable the model to learn the meaning of word pieces. Additionally the embedding layer adds a positional embedding and a segment embedding to mark the belonging of tokens to a sentence. We utilize a vocabulary size of 30,522 tokens and a maximum sequence length of 512 tokens.

Each encoder applies the *self-attention* mechanism to the input by employing twelve attention heads in parallel. Every attention head extracts different features. This process is performed repeatedly by passing the input through a chain of twelve encoders, each capturing structurally hierarchical meaning from the input at different depth.

We use pretrained weights for the embedding and the encoders. Additionally we fine-tune the model on the last layer, the toxicity-head. In the *CLS* token at the beginning of each embedded text, aggregated information about the entire input sequence is stored. The classification into six non-exclusive categories of toxicity is performed on this token.

Additional Hyperparameters Among the several approaches to solve the multi-label classification tasks, in Milestone 2, we chose the transformation into a multi-class problem. We realized this by assigning a binary class to each output-neuron of the toxicity-head. On this, the *binary cross-entropy* (BCE) loss is computed and averaged over all classes to obtain a single loss value for the optimization. Equation 1 is showing the BCE loss $\ell_{n,c}$ per class c, where n is the number of labels per batch, $y_{n,c}$ is the label and $x_{n,c}$ is the model's prediction.

$$\ell_{n,c} = -\left[p_c y_{n,c} \cdot \log \sigma(x_{n,c}) + (1 - y_{n,c}) \cdot \log(1 - \sigma(x_{n,c}))\right] \tag{1}$$

This method is computationally most efficient and most easily interpretable in comparison to others.

As an alternative to common approaches to address an imbalanced data set like *under*- or *oversampling*, we decided to apply weights (p_c) in the loss function to the positive label instances only[1]. This is also due to computational complexity reasons and maintaining the integrity of the data set. Errors in these samples are penalized higher according to:

$$p_c = \frac{dataset\ size}{(number\ of\ comments\ labeled\ c) \cdot (number\ of\ classes)} \tag{2}$$

We chose the *Adam*-optimizer (Kingma and Ba, 2015) which was used in the original BERT implementation (Devlin et al., 2019) and proposed by Sun et al. (2019) for fine-tuning. Adam calculates moving averages and squared moving averages of past gradients to update the gradient effectively. The utilized parameters for Adam are $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

The learning rate is determined by the *Slanted Triangular Discriminative Learning rate* (STDL) method. This approach has two aspects:

A *slanted triangular learning rate*, as in Howard and Ruder (2018), which increases linearly in the first ten percent of the iterations to the desired base learning rate and then drops linearly. This results in a pronounced exploratory phase at the beginning, which helps at reaching a global optimum and an exploitative phase in the end, which enables convergence.

The other aspect is the *discriminative learning rate* scheduling, that Howard and Ruder (2018) proposed. This means, a decay factor is applied to the learning rate layer-wise to affect deeper layers of the model less. Each layer of a deep neural network captures different kind of information. Since the lower layers are more responsible for general understanding, these should not be changed as much during fine-tuning. Otherwise this could result in *catastrophic forgetting* (McCloskey and Cohen, 1989). Our base learning rate is $\eta^L = 1e - 05$; the formula for the learning rate η of layer $k - 1$ is: $\eta^{k-1} = \xi \cdot \eta^k$, where ξ is a decay factor. We applied $\xi = 0.95$, meaning the learning rate decreases in deeper layers.

In general, our applied learning rate of $1e - 05$ is much lower than proposed learning rates for fine-tuning BERT as e.g. in Sun et al. (2019). This is due to the pronounced sensitivity of the gradient to the learning rate caused by the applied class weights to the loss function.

We utilize a batch size of 16 samples for reasons of computational complexity; a value also suggested by Devlin et al. (2019).

As described in Milestone 2, we had the optimal result after four epochs with the STDL-method. Therefore, we train the model for four epochs, also suggested by Devlin et al. (2019).

At last, dropout is a countermeasure to overfitting. The dropout rate p defines the fraction of randomly selected neurons which *activations* are set to zero during training. This reduces complexity and emphasizes the response of the rest of the neurons. We use $p = 0.1$, as proposed by Sun et al. (2019).

3 Evaluation

In this section, we are going to analyze the performance of the model and assess the suitability of the algorithm for solving the task in terms of classification performance. Afterwards, we are going to analyze how confident the model is about its predictions. Then we apply XAI (*eXplainable Artificial Intelligence*) methods to investigate how understandable the inner workings of the model are. In the end, we analyze fairness aspects, i.e. the presence of biases within the model.

3.1 Performance

This section provides a brief introduction of the central performance measure on which the model selection was performed in Milestone 2.

[1]Details about the positive instances weights: `https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html`

ROC-AUC The applied performance measure is AUC-ROC (*Area Under The Curve - Receiver Operating Characteristics*), as described by Fawcett (2006). The derived AUC score is, generally speaking, an indicator of how well a model is able to differentiate between given classes.

To obtain this value, the *True Positive Rate* (TPR) and *False Positive Rate* (FPR) are evaluated on the ROC-curve across various thresholds and then aggregated as an approximation of the area under this curve. Equation 3 shows the formulas of TPR and FPR:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{3}$$

The AUC score ranges from 0.0 (indicating the model predicts the opposite of the label) across 0.5 (indicating the prediction of classes is completely arbitrary) to 1.0 (indication of perfect distinction).

In our case, the ROC-AUC is computed for each binary class separately and can be understood as measure of the model's ability to distinguish between the classes $true$ and $false$ for a given label. These values are averaged per class across the validation set.

Results Table 1 presents the ROC-AUC scores for each respective label which was obtained from the selected model during the validation phase.

Average	Toxic	Severe Toxic	Obscene	Threat	Insult	Identity hate
0.98	0.97	0.99	0.98	0.99	0.97	0.99

Table 1: Averaged ROC-AUC scores for the classes of toxicity.

These scores are relatively high for all classes and can be considered close to an optimal distinction performance, as they match with the top ranking entries of the Kaggle leaderboard[2].

3.2 Confidence

In this section, we introduce the Shannon-based entropy as a confidence measure and discuss whether it helps in identifying instances where the model is uncertain.

Shannon-Based Confidence We employed a confidence measure based on the *Shannon entropy*: the *Shannon-based confidence* (cf. Bukowski et al. (2021)). It makes use of the entropy as measure of uncertainty for a prediction, turning it into a confidence by subtracting it from one. For a n-dimensional probability vector v_n the Shannon-based entropy is defined as:

$$Conf_{Shannon}(v_n) = 1.0 - H_n(v_n) \tag{4}$$

The entropy $H_n(v_n)$ is computed for each class. The resulting probability vector $v_2 = (p, (1-p))^T$ accounts for the probability of the presence p and the probability of the absence $(1-p)$ of the class. Therefore the entropy H_2 considers two states:

$$H_2 = -[p \cdot log(p) - (1-p) \cdot log(1-p)] \tag{5}$$

A Shannon-based confidence value close to one signifies perfect certainty of the model about its prediction. Conversely, a confidence score close to zero indicates no certainty about the prediction. The confidence scores are averaged over the validation set per class.

Results In Table 2, the averaged Shannon-confidence values associated with each class are presented which were obtained during the validation phase.

Average	Toxic	Severe Toxic	Obscene	Threat	Insult	Identity hate
0.95	0.92	0.97	0.95	0.98	0.93	0.96

Table 2: Averaged Shannon-based confidences for the classes of toxicity.

[2]https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/leaderboard

Overall, the model shows a high confidence in its predictions. Especially the classes "severe toxic", "threat" and "identity hate" rank above the average, recording the best confidence scores. The reason for this might be the classifiability by class-specific keywords as described in section 3.3.

The classes "toxic" and "insult" are the only ones falling below a confidence of 95 percent. The intuition for this might be the slightly more vague definition of these labels, therefore the interpretation is more up to the human assigning the labels to the data set. The concept of these classes is hence somewhat less easy to specify by keywords.

The confidence scores align with the ROC-AUC scores for the respective classes (see Table 1). The classes "severe toxic", "threat" and "identity hate" score all at 0.99, which indicates almost perfect distinction abilities of the model. The ROC-AUC values for the other classes are slightly lower at 0.97 - 0.98.

3.3 Transparency

There are several reasons for implementing transparency for AI systems. For example, explanations can be used to justify decisions and improve and control AI systems (Adadi and Berrada, 2018).

In this section, we will first review the literature of XAI methods. Afterwards, we explain *Integrated Gradients* and why we chose this method. Finally, we evaluate the explanations by doing experiments based on interesting samples.

XAI methods There is a huge landscape of XAI methods to choose from (Adadi and Berrada, 2018). Each XAI method comes with its own benefits and challenges. Since we are using BERT as a model, several model-specific explanation methods have to be excluded. To narrow down the search space of explanation methods, we set the following constraints: As we infer the behaviour of the model from individual predictions instead of understanding the model as a whole, we are interested in *local methods* only. As we want to keep the complexity of the explanation method low, we focus on *post-hoc methods*, i.e. methods that can be applied after training (Adadi and Berrada, 2018). Additionally, we only consider *feature attribution methods*. Feature attribution methods try to measure the contribution of each individual feature to the model's output (Adadi and Berrada, 2018).

Even when only considering local, post-hoc prediction explanation methods, there is a large number of methods to choose from. Many of these fall into one of the following categories:

Perturbation based explanation methods explain predictions by adjusting the input and observing how the model reacts (Fong and Vedaldi, 2017; Ivanovs et al., 2021). *Gradient based explanation methods* rely on the gradient of the model's output with respect to the input features to attribute relevance to each feature in influencing the model's prediction (Selvaraju et al., 2017; Sundararajan et al., 2017; Shrikumar et al., 2017; Simonyan et al., 2013). *Surrogate/-Sampling based methods* involve constructing simpler, easier interpretable models (surrogates) or generating representative samples from the original model's data distribution to gain insights into its decision-making process (Ribeiro et al., 2016; Smilkov et al., 2017). *Propagation-based methods* focus on tracing the propagation of input features through the model's layers or components to identify which features are most influential on the model's predictions (Bach et al., 2015; Montavon et al., 2017).

We decided to use gradient-based methods for the following reasons: Since we want to explain many data points in a reasonable time, the explanation method should be fast. Compared to perturbation-/occlusion based methods and surrogate/-resampling methods, gradient-based methods are usually faster to compute (Ancona et al., 2017). In contrast to propagation-based methods, gradient-based methods are easier to implement and do not introduce sensitive hyperparameters such as which rules to choose in *Layer wise Relevance Propagation* (LRP), as described by Bach et al. (2015).

We decided to use *Integrated Gradients* over other gradient based methods (Simonyan et al., 2013; Shrikumar et al., 2017) for reasons explained below.

Integrated Gradients Integrated Gradients as introduced by Sundararajan et al. (2017) is a theoretically well-founded method that is superior to other gradient-based attribution methods because it follows two important axioms:

Completeness (Sundararajan et al., 2017). Consider a function F, which is approximated by a neural network and an input x to that network as well as a baseline x' in a sense that $F(x')$ would be a *neutral prediction*. In case of a text-based classification this would be the zero token for instance. Completeness means that all attributions sum up to the difference between $F(x)$ and $F(x')$. Completeness also implies another axiom which is *Sensitivity*, defined as:

> "An attribution method satisfies Sensitivity[...] if for every input and baseline that differ in one feature but have different predictions then the differing feature should be given a non-zero attribution." Sundararajan et al. (2017), p. 3

This is fulfilled because if the baseline and the input only differ in one variable, the difference in the corresponding output would be equal to the attribution of this variable.

Sensitivity may not be fulfilled by common gradient-based attribution methods. This is because the transfer function possibly flattens an input, meaning the gradient becomes zero. However, at the same time, the function value for that input is different from that for the baseline. The result is a focus of the gradient on irrelevant features.

Implementation Invariance (Sundararajan et al., 2017). If two networks are *functionally equivalent*, meaning for the same input they yield the same output regardless of their implementation, their attributions are always identical for the same input. If this axiom is violated the attributions can be prone to focus on irrelevant features as well.

Integrated Gradients integrates over all points α on a straightline path from the baseline x' to the input x. With $F(x)$ as the model's output and $\frac{\partial F(x)}{\partial x_i}$ as the partial derivative of F along the i^{th} dimension, Integrated Gradients is defined as:

$$IntegratedGrads_i = (x_i - x'_i) \cdot \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \cdot (x - x'))}{\partial x_i} d\alpha \tag{6}$$

In practice, the integral is approximated efficiently by summing over m discrete steps along the path from x' to x. In the context of our application, an Integrated Gradient is calculated per component of a token vector, summed up and normalized per token for one class. A general attribution score per class is obtained from the sum of all tokens of a text.

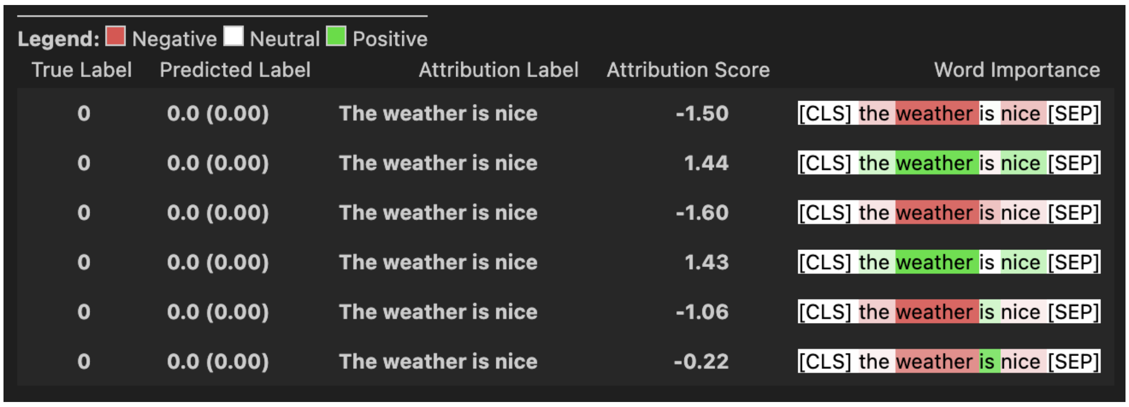
Results To calculate the Integrated Gradients and visualize them, the *Captum library* [3] was used. Captum is an open-source model interpretation library for PyTorch, which helps understand and interpret the behavior of neural networks.

Each token of the input sentence is analyzed to calculate the attribution score to each label separately since tokens can influence each class differently. As visible in Figure 1, the model outputs a true label, a predicted label with a prediction score, an initial input sentence, also called an attribution label, an average attribution score of the whole input, and finally, a visual interpretation of the word importance.

The word importance returns an attribution score for each token in the sequence. Tokens that are colored in green contribute more to a positive classification of the current label compared to other tokens. Following the same idea, the model highlights the tokens with red, if they contribute negatively to the prediction. Tokens that do not influence the model (with attribution scores around 0) are represented with white color.

All visualizations follow the order of the initial labels of the training dataset (toxic, severe toxic, obscene, threat, insult, identity hate). For example, the first row of the discussed image will correspond to the attribution to the neuron, responsible for "toxic" prediction, the second for "severe toxic" and so on.

[3] `https://captum.ai`

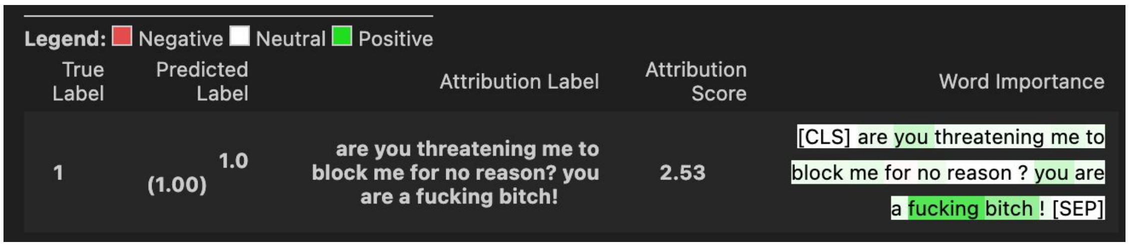


Legend: 🟥 Negative ⬜ Neutral 🟩 Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
0	0.0 (0.00)	The weather is nice	-1.50	[CLS] the weather is nice [SEP]
0	0.0 (0.00)	The weather is nice	1.44	[CLS] the weather is nice [SEP]
0	0.0 (0.00)	The weather is nice	-1.60	[CLS] the weather is nice [SEP]
0	0.0 (0.00)	The weather is nice	1.43	[CLS] the weather is nice [SEP]
0	0.0 (0.00)	The weather is nice	-1.06	[CLS] the weather is nice [SEP]
0	0.0 (0.00)	The weather is nice	-0.22	[CLS] the weather is nice [SEP]

Figure 1: Influence of tokens. Attribution scores are calculate for (toxic, severe toxic, obscene, threat, insult, identity hate) classes

Analyzing the comment text (Figure 2), which belongs to the "toxic" class, it can be observed that it is correctly predicted. The model gave the highest attribution scores to "fucking" and "bitch", while on the other hand, it also highlighted "you" with light green, even though this is not intuitive. The former two potentially represent typical keywords for the class.



Legend: 🟥 Negative ⬜ Neutral 🟩 Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
1	1.0 (1.00)	are you threatening me to block me for no reason? you are a fucking bitch!	2.53	[CLS] are you threatening me to block me for no reason ? you are a fucking bitch ! [SEP]

Figure 2: 'Toxic' class comment

In the majority of cases, the model is able to correctly identify tokens, that have a high contribution to a positive classification, and we conclude that the Integrated Gradient is well-suited as an explanation tool for our specific task.

3.4 Fairness

We refer to fairness as the absence of biases against groups (Zhou et al., 2022). The necessity for fair AI Systems is currently not only motivated ethically, but also by law. In the EU there is a law within the General Data Protection Regulation[4] (GDPR) that prohibits biased decision making against individuals based on factors such as sexuality, ethnicity, religious beliefs, etc.

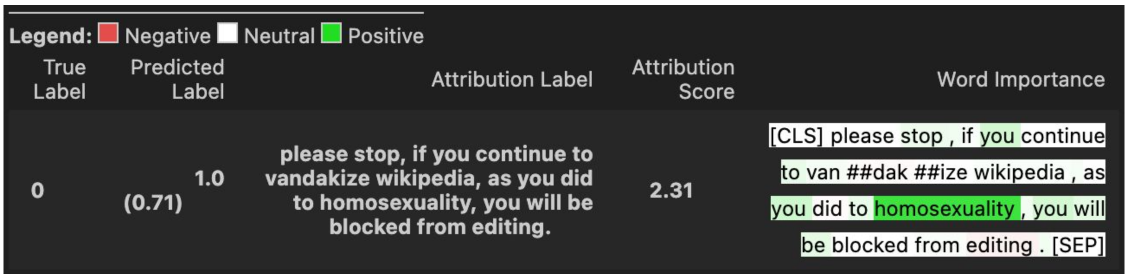
One of the issues we observed within section 3.3 is that the model seems to be sensitive to the presence of certain keywords. The authors of the original Kaggle challenge noticed this behavior and came to the conclusion, that their training data is biased towards negative usage of these keywords. One example would be the keyword "gay". In online forums keywords like that are usually used in a negative context. To find ways to deal with these biases, the authors even launched a second competition[5].

In this section, we will explain the annotation procedure of the dataset from the new Kaggle challenge. Afterwards we will analyze the performance and confidence of our model with respect to the new annotation groups to estimate the magnitude of bias of the model is towards certain groups.

[4]https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504
[5]https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/overview

Additional annotations The dataset from the new Kaggle challenge involves extended annotations, processed by human raters. These raters annotated comments with labels related to diverse toxic conversational attributes, indicating whether specific identities were mentioned in a comment. These identity groups comprise labels like "male", "female", "homosexual_gay_or_lesbian", "christian", "jewish", "muslim", "black", "white" or "psychiatric_or_mental_illness" for instance. The identity group values range from 0.0 to 1.0, representing the fraction of raters who believed the identity fits the comment. To simplify the processing, we introduced a threshold of 0.5. For evaluation purposes, we only use the identities with more than 500 samples.

Results As stated before in the description of subsection 3.4, the training data could be biased towards a negative usage of keywords. We observed such a pattern empirically while testing samples with low confidence for certain labels. The following Figure 3 represents an example of biased model behavior. Intuitively, the input sentence would not be categorized as ill-intended or correspond to any of the predicted classes. However, due to the fact that "homosexuality", which has a high attribution score, seems to be more often used in a negative context in online forums, the model prediction is heavily impacted, leading to the wrong classification.

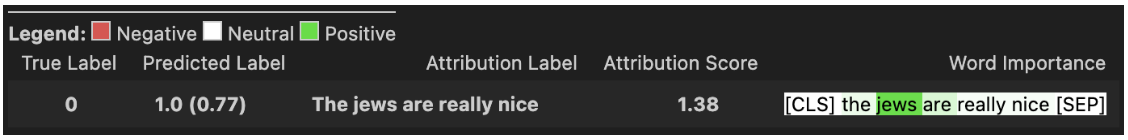


Legend: 🟥 Negative ⬜ Neutral 🟩 Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
0	1.0 (0.71)	please stop, if you continue to vandakize wikipedia, as you did to homosexuality, you will be blocked from editing.	2.31	[CLS] please stop , if you continue to van ##dak ##ize wikipedia , as you did to homosexuality , you will be blocked from editing . [SEP]

Figure 3: Predicted as "toxic" sample

We illustrate the pattern with a synthetic example (Figure 4), where the input sentence was incorrectly classified as 'identity hate'. The reason is the same as in the previous example, as the word "Jews" is rather used in a negative context than in a positive context in online forums.



Legend: 🟥 Negative ⬜ Neutral 🟩 Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
0	1.0 (0.77)	The jews are really nice	1.38	[CLS] the jews are really nice [SEP]

Figure 4: Predicted as "identity hate" sample

To further investigate the bias issue of the new dataset with the new annotations, we investigate average AUC scores (Table 3) for different identity groups. It can be observed that the model's discrimination performance for some identities is not that strong. For instance, "Black" or "Psychiatric or Mental Illness", the AUC values are considerably low. It is also lower than the AUC scores of samples that do not correspond to the defined identities, which we call 'non-identity' comments (for example, comparing AUC scores of samples ('Male'= 1) to ('Male' = 0)).

Summing up, the outcomes indicate a noticeable bias in the training data. This is potentially due to the fact that adversarial comments or any kind of toxic abuse are usually directed at minorities.

Identity	**AUC**	**AUC (non-identities)**
Male	0.90	0.92
Female	0.88	0.92
Homosexual, Gay, or Lesbian	0.84	0.92
Christian	0.90	0.92
Jewish	0.84	0.92
Muslim	0.86	0.92
Black	0.82	0.92
White	0.88	0.92
Psychiatric or Mental Illness	0.80	0.92

Table 3: AUC averaged over all classes for different identities compared to the corresponding non-identities

3.5 Business Use Case

To evaluate the AI system, it is important to keep the intended usage of the model in mind. On the website of the Kaggle competition[6], they explain their intentions. The competition was hosted by the *conversationAI* team, which is part of *Alphabet*. It utilizes a dataset of user comments from *Wikipedia's talk page edits*. ConversationAI intended to improve their AI models in detecting toxicity, since their models have been making mistakes. In summary, the intended use of the models is content moderation.

Content Moderation Recently, there has been a lot of effort in trying to automatize content moderation for example due to the new regulatory measures such as *EU Code of Conduct on Hate Speech* or the the German *NetzDG*. These measures try to put pressure on platforms to take down illegal content faster (Gorwa et al., 2020).

Platforms are fully responsible for all decision that the underlying system makes. Making mistakes while classifying toxicity comes at high cost, independently of the mistake being a *false positive* or *false negative*. Besides legal concerns, there are numerous ethical concerns for the moderation of content on platforms. In case of a false positive, removing a comment is basically censorship and violates the freedom of speech. On the other hand, identity hate is illegal and thus should not be tolerated. Since AI models are not perfect, there are going to be mistakes. One key question therefore is how to prevent mistakes other than just improving raw classification performance.

Business Value Regarding the use of the model for content moderation, there are many challenges but also opportunities. Assuming that our test error reflects the error on real world data, the model can definetely help in detecting toxic comments. Even though the model rarely makes mistakes, it is not sufficient to use the model in isolation. Therefore humans still should be considered as a part of the decision processes.

One way of implementing a more reliable model could be to put a human into the loop, when the confidence of the AI model is low. If the confidence falls below a certain threshold, a human can check the predictions. By adjusting the confidence threshold, it would be possible to balance costs for manual labor and accuracy of the model.

One other way is to manually validate every positive prediction, i.e. comments classified as toxic. The intuition is, that there are other ways like buttons to report toxic comments, where false negatives can be corrected by the community itself. This yields the risk of misuse of report on the other hand.

4 Discussion and Outlook

All in all, when looking at the raw classification performance and confidence, the model performs very well on the task. However, when taking the additional annotations of the new Kaggle competition into account, biases are obvious. There are ways to eliminate these biases, as described by Mehrabi et al. (2021). One idea is assigning weights for each data point in the learning function. This way, one could encourage the model to put more focus on learning from instances, where keywords like e.g. ”gay” are used in

[6]`https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/overview`

a non-toxic context. Nevertheless, there are other possibilities which is reflected by the diversity of solutions from the second Kaggle competition.

We found that applying the Shannon-based confidence helped in identifying samples, where the model is likely to make mistakes. It is easy and fast to compute. We found it to be a useful tool to determine which samples should be validated by a human.

Regarding transparency, we found IG to be a useful tool for examining the predictions. The explanations reflected the learned biases that the authors of the kaggle competition highlighted as a motivation for the second kaggle competition (see subsection 3.4). For low confidence predictions, we can see, why the model is struggeling. Further research could include trying out other XAI methods.

To conclude, we find the model to be able to help in content moderation, when putting a human in the process for validation. Nevertheless, there are various regulations that have to be considered as well. Further research could include cost estimations for the deployment of such a system.

References

A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018. doi: 10.1109/ACCESS.2018.2870052.

M. Ancona, E. Ceolini, C. Öztireli, and M. H. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2017. URL `https://api.semanticscholar.org/CorpusID:3728967`.

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS One*, 10(7):e0130140, 2015. doi: 10.1371/journal.pone.0130140.

M. Bukowski, J. Kurek, I. Antoniuk, and A. Jegorowa. Decision confidence assessment in multi-class classification. *Sensors*, 21(11), 2021. ISSN 1424-8220. doi: 10.3390/s21113834. URL `https://www.mdpi.com/1424-8220/21/11/3834`.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2005.10.010. URL `https://www.sciencedirect.com/science/article/pii/S016786550500303X`. ROC Analysis in Pattern Recognition.

R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3449–3457, 2017. doi: 10.1109/ICCV.2017.371.

R. Gorwa, R. Binns, and C. Katzenbach. Algorithmic content moderation: Technical and political challenges in the automation of platform governance. *Big Data & Society*, 7(1):2053951719897945, 2020. doi: 10.1177/2053951719897945. URL `https://doi.org/10.1177/2053951719897945`.

J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL `https://aclanthology.org/P18-1031`.

M. Ivanovs, R. Kadikis, and K. Ozols. Perturbation-based methods for explaining deep neural networks: A survey. volume 150, pages 228–234, 2021. doi: https://doi.org/10.1016/j.patrec.2021.06.030. URL `https://www.sciencedirect.com/science/article/pii/S0167865521002440`.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021. ISSN 0360-0300. doi: 10.1145/3457607. URL `https://doi.org/10.1145/3457607`.

G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2016.11.008. URL `https://www.sciencedirect.com/science/article/pii/S0031320316303582`.

M. Ribeiro, S. Singh, and C. Guestrin. “why should I trust you?”: Explaining the predictions of any classifier. In J. DeNero, M. Finlayson, and S. Reddy, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-3020. URL `https://aclanthology.org/N16-3020`.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. doi: 10.1109/ICCV.2017.74.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML’17, page 3145–3153. JMLR.org, 2017.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. URL `https://api.semanticscholar.org/CorpusID:1450294`.

D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *ArXiv*, abs/1706.03825, 2017. URL `https://api.semanticscholar.org/CorpusID:11695878`.

C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In M. Sun, X. Huang, H. Ji, Z. Liu, and Y. Liu, editors, *Chinese Computational Linguistics*, pages 194–206, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32381-3.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/sundararajan17a.html`.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`.

J. Zhou, F. Chen, and A. Holzinger. *Towards Explainability for AI Fairness*, pages 375–386. 04 2022. ISBN 978-3-031-04082-5. doi: 10.1007/978-3-031-04083-2_18.