A New Generation of Perspective API: Efficient Multilingual Character-level Transformers

Alyssa Lees*
Jigsaw
New York, USA
alyssalees@google.com

Vinh Q. Tran*
Google Research
New York, USA
vqtran@google.com

Yi Tay*
Google Research
Singapore
yitay@google.com

Jeffrey Sorensen
Jigsaw
New York, USA
sorenj@google.com

Jai Gupta
Google Research
Mountain View, USA
jaigupta@google.com

Donald Metzler
Google Research
Mountain View, USA
metzler@google.com

Lucy Vasserman
Jigsaw
New York, USA
lucyvasserman@google.com

ABSTRACT

On the world wide web, toxic content detectors are a crucial line of defense against potentially hateful and offensive messages. As such, building highly effective classifiers that enable a safer internet is an important research area. Moreover, the web is a highly multilingual, cross-cultural community that develops its own lingo over time. As such, it is crucial to develop models that are effective across a diverse range of languages, usages, and styles. In this paper, we present the fundamentals behind the next version of the Perspective API from Google Jigsaw. At the heart of the approach is a single multilingual token-free Charformer model that is applicable across a range of languages, domains, and tasks. We demonstrate that by forgoing static vocabularies, we gain flexibility across a variety of settings. We additionally outline the techniques employed to make such a byte-level model efficient and feasible for productionization. Through extensive experiments on multilingual toxic comment classification benchmarks derived from real API traffic and evaluation on an array of code-switching, covert toxicity, emoji-based hate, human-readable obfuscation, distribution shift, and bias evaluation settings, we show that our proposed approach outperforms strong baselines. Finally, we present our findings from deploying this system in production.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Natural language processing**; **Supervised learning by regression**; **Transfer learning**.

*Equal contribution, ordered alphabetically.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539147

KEYWORDS

moderation, text classification, multilingual

ACM Reference Format:
Alyssa Lees, Vinh Q. Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. 2022. A New Generation of Perspective API: Efficient Multilingual Character-level Transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539147

1 INTRODUCTION

Developing robust and effective content moderation systems is a crucial component for keeping the web safe from abusive users. Offensive, toxic, and harassing content has the potential to significantly harm users along with wide-ranging negative effects on broader society. To this end, building machine learned systems that are able to detect toxic content is a well-established and highly important research area, and an essential component of modern platforms' content moderation workflows, alongside robust human moderator teams.

Most research in this area has been focused on building specialized models for specific locales, languages, domains, or label distributions [2, 20, 24, 33]. Thus, it is common practice to build monolingual models for specific languages and/or domains. Given that the web is highly multilingual, multi-cultural, and typographically diverse, monolingual systems are likely to under-perform in real applications, as they are unable to handle code-switching, cross-cultural phenomena, or cross-lingual generalization.

Due to the rigidity of feature-based machine learning models, subword tokenization, and byte-pair encoding [17] based deep learning models (e.g., BERT [9]), many of these models are not universally applicable across different languages and/or tasks and are considered more or less static once trained. This makes it difficult to apply a single model across a diverse range of languages, domains, and tasks. It also makes it challenging to incrementally train a model on new downstream applications. Furthermore, rigid vocabularies

are also vulnerable to common adversaries of the social web - misspellings, emojis and obfuscation, all of which are techniques that are commonly used for microaggressions and covert attacks [19].

With these challenges in mind, this paper presents a new generation of toxic content classifiers for Jigsaw's Perspective API. We refer to this generation as UTC (*Unified Toxic Content Classification*), centering around a new modeling framework for highly performant and robust toxic content detection. In summary, UTC is a **single** compact pretrained Charformer-based Transformer [29] that is pretrained on multilingual documents along with comment text from a wide variety of online discussion forums and other sources of user generated content using a seq2seq denoising loss [23].

UTC leverages recent novel advances of Learnable Tokenizers as part of the model architecture [29] and is therefore vocabulary- and token-free. While character-level features or modeling approaches have been explored in the context of toxicity detection [18], our paper proposes the first byte-level pretrained model that remains competitive with (or outperforms) subword models tailored to specific domains or languages. Notably, this vocabulary free property of UTC enables it to be both language agnostic and more robust to domain transfer.

Furthermore, in the design of UTC we address major practical challenges of productionizing character-level Transformers in a latency sensitive, public API setting. We describe the approach in detail in Section 3 and show the impact of our changes in Table 8. The result is a model compact enough to be served in real-life production settings while demonstrating competitive performance versus the winning entries of the 2020 *Jigsaw Multilingual Toxic Comment Classification* Kaggle contest, even though it is >10x more memory (parameter) efficient. Extensive evaluations for model bias [5] also show that UTC is reasonably unbiased across multiple languages. In summary, the primary contributions of this work can be summarized as follows:

- We present Unified Toxic Content Classification (UTC), a modeling framework suitable for efficient character-level multilingual moderation workflows.
- We conduct very extensive and rigorous experiments on multiple tasks and benchmark datasets, from both academic settings and sampled from our production traffic. We show that UTC outperforms strong baselines such as a multilingual BERT model pretrained on comments and state-of-the-art mT5 models.
- For evaluating the robustness and flexibility of the proposed approach, we include benchmarks that specifically test the model's ability to handle code-switching, covert toxicity, emojis, obfuscated text, distribution shifts, and model bias. We show that under all conditions, UTC outperforms or matches strong baselines.
- We present the results of our experience deploying UTC into production Perspective API.

2 RELATED WORK

Although there has been a long history of using machine learning to detect abusive content (e.g., email spam [6]), research using direct text classification started in earnest with the introduction of the modest sized hand labeled data in [8], [22] and [32]. These works also coincide with the launch of the Workshop on Online Abuse and Harms [1] which completed its fifth annual meeting.

There has also been a significant amount of criticism regarding the application of machine learning to conversation moderation, see [36] for a recent survey of the issues and challenges. The nature of online identity and social relationships, and the problems of governance are complex and involve many interacting entities with overlapping jurisdictions. And despite the popularity of some shared, labeled test sets, there is little consensus within the community regarding sampling, annotation standards, annotator recruitment and training, classifier design, or scoring metrics.

One concern regarding the use of machine learning models for moderation is that flaws in the training data, whether due to the process used to collect the data, biases held by the annotators, or underlying societal, historical biases, whether intentional or unconscious, can manifest in models as unintended discriminatory biases. This concern was raised in [7] and [25]. [12] provides a good overview of these concerns. To address these concerns we employ the techniques of data augmentation suggested in [10] and [5], and include a cross-language bias analysis to measure the unintended bias for similar terms across all languages. It is also worth noting the progress towards a more comprehensive taxonomy of abusive content has drawn interdisciplinary attention, and systemic annotation efforts [14] which also inform our work. There has also been criticism regarding current commercial models' performance on tagging abusive, toxic, or hateful content. This includes many examples of adversarial perturbations designed to "fool" moderation models, such as proposed in [11]. While the present work demonstrates improved performance against these types of attacks, the task of improving models in these areas is ongoing.

The English language has dominated research in classifying offensive content, although there have been numerous publications focusing on other specific languages. There is still no agreement regarding the efficacy of training multilingual models versus monolingual models, but for applications with user generated content, not needing to ask or guess what language is being used has clear advantages. Two recent examples of similar work are [31], which found that monolingual models do not universally perform better on sentiment and hate speech classification, and [28] which uses model fusion in an attempt to correct the imbalance in available training resources.

3 UTC: UNIFIED TOXIC CONTENT CLASSIFICATION

This section introduces UTC, the proposed modeling framework in this paper.

3.1 Learnable Tokenizer

The input to our model is a sequence of UTF-8 bytes. After mapping each byte id to an embedding lookup, the input to our model is a tensor $X \in \mathbb{R}^{L_{bytes} \times d_{model}}$ where L_{bytes} is the number of bytes and d_{model} is the number of hidden dimensions. In order to automatically learn subwords in a data-driven fashion, we adopt state-of-the-art Charformer encoders [29].

[1] http://www.workshopononlineabuse.com/

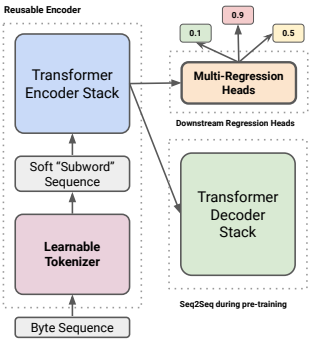

Figure 1: Overview of the UTC architecture.

3.1.1 Learning Latent Subwords Automatically. In this section we review the gradient-based subword tokenization module (GBST) from [29]. GBST dynamically down-samples a sequence of byte embeddings into a sequence of latent subword embeddings in a process that resembles subword tokenization, but can be implemented differentiably. The key idea of GBST is to encourage local composition by performing *position-wise* block scoring. In other words, at every position, we predict a scored list of blocks with each block representing a different size context around the current position. e.g. at a position i, we may consider every block of size 1, 2, 3, and 4 intersecting with position i. Each block is scored using a block scoring network, parameterized by a simple linear transformation, that maps each candidate subword block embedding into a scalar $\in \mathbb{R}$ that denotes its strength of being included in the final subword composition at the current position. In detail, given a sequence of byte embeddings:

(1) The model constructs block candidates of varying sizes. Let the maximum possible block size be M and b be the current block size, we use a non-parameterized strided pooling function $F : \mathbb{R}^{b\times d} \rightarrow \mathbb{R}^d$ that projects a subword block consisting of a sequence of byte embeddings $X_{i:i+b} \in \mathbb{R}^{b\times d}$ to a single subword block representation $X_{b,i} \in \mathbb{R}^d$ for block size b at position i. When applied across the sequence, we compute a sequence of subword blocks X_b:

$$X_b = [F(X_{i:i+b}); F(X_{i+b:i+2b}); \ldots] \quad (1)$$

In practice we set $M = 4$ to enumerate blocks sized 1 to 4. Following previous work, since we enumerate blocks here with a stride of b we apply a 1D convolution of size $b + 1$ before this enumeration step.

(2) Next we use the block scoring network (a linear transformation) to score every block in each of $X_1, \ldots, X_M$. We then upsample every sequence X_b and their scores back to original sequence length L_{bytes} via repetition. At this point, we have a set of block embeddings $X_{b,i}$ and their scores $p_{b,i}$ for every position i and block size b.

(3) We take the softmax of the scores across block size for each position: $P_i = \text{softmax}\left([p_{0,i}, p_{1,i}, \cdots, p_{M,i}]\right)$.

(4) We construct the locally composed sequence representation $\hat{X}$ by reducing over the block size dimension. In particular we take the sum of every $X_{b,i}$ at position i weighed by their block score: $\hat{X}_i = \sum_b^M P_{b,i} X_{b,i}$.

(5) Finally $\hat{X}$ is down-sampled by mean pooling.

We refer interested readers to [29] for fine-grained details.

3.2 Transformer Stack

The Transformer stack in our approach accepts latent subwords from the Learnable Tokenizer as an input and the remainder of the Transformer stack remains identical to a standard Transformer model. Transformer architectures are characterized by stacks of self-attention blocks followed by simple feed-forward layers [30].

3.3 Reconfigurable Seq2Seq Architecture

Our pretraining utilizes a Seq2Seq (Encoder-Decoder) architecture that is optimized by teacher forcing. In practice, we find this denoising loss to be more effective than encoder-only (BERT-based) pretraining. Intuitively, Seq2Seq based masked language modeling also enables sequential and long-term dependencies to be taken into account in the autoregressive generation process. Our Seq2Seq architecture is reconfigurable, i.e., during certain tasks, we may remove the decoder for specialized regression or classification heads while retaining a universal encoder for all tasks. With this formulation, we can retain a unified encoder across all tasks from shared representation learning. While the T5 model [23] enables regression problems to be framed in Seq2Seq architectures, we find that adding regression heads is more natural and effective in practice. Moreover, this supports the case where we have multiple labels per input example. Note that the entire UTC Seq2Seq architecture can also be finetuned on downstream classification tasks.

3.3.1 Seq2Seq Loss. During pretraining, our model optimizes the following cross entropy loss: $L = -\sum_{t=1}^{L}\sum_{i=1}^{n} \log(\pi_i^t) + (1-y_i^t)\log(1-\pi_i^t)$ where π_i^t is the prediction of class i at time step t and y_i^t is the ground truth label of the class i at time step t.

3.3.2 Multi-Regression Heads and Loss Function. While the model's main focus is to predict a single value $\in [0, 1]$ denoting a *toxicity probability*, our method also generalizes to k-way regression to support predicting toxicity subtypes. (For example, if the sample is hateful or obscene). Therefore, for regression tasks, we equip our model with a linear transform that maps the encoder output to a k_r way regression head. This is expressed as: $y_R = W_r(\psi(Y'_{out}))$ where $y_R \in \mathbb{R}^{k_r}$ and Y'_{out} is the output of the last encoder layer. ψ is a non-parametric or parametric pooling function that maps $\mathbb{R}^{L\times d_{model}} \rightarrow \mathbb{R}^{d_{model}}$ (L is the sequence length) and $W_r \in \mathbb{R}^{d\times k_r}$ are learnable parameters of the regression head. We adopt a *first pooling* for ψ in similar spirit to BERT's CLS token and include a dummy task prefix token in front of each example following [23]. Before pooling, we project Y'_{out} using a GeLU MLP layer to the same hidden size, i.e., d_{model} which constitutes the parameters of the pooling layer. For regression head, our model optimizes the sigmoid cross entropy loss.

3.4 Pretraining

Our model is pre-trained on an equal mixture of two data sources: Perpsective Pretraining Corpus (PPC) and the mC4 corpus from mT5 [35]. PPC is a proprietary corpus of ~4.6B message and comment texts from a variety of sources including data historically processed by the Perspective API or shared by partners. Note that

API clients can enable/disable this data storage via an API flag (`doNotStore`[2]). Text in this corpus typically comes from a variety of online forums. We mix the two corpora equally, sampling equally between target languages within the mC4 mixture, while using the natural language distribution in the PPC split. We pretrain our method using the span-based denoising objective in a Seq2Seq fashion using a *mean* span corruption length of 20 bytes and a corruption rate of 15%. We pretrain for 1M steps and batch size of 128 sequences, with the maximum length for each sequence set to be 512 bytes.

4 EXPERIMENTAL SETTINGS

This section provides an overview of our experimental setup.

4.1 Datasets

We conduct three categories of experiments: core multilingual toxic comment classification, robustness evaluation, and evaluation of adaptation to new types of toxicity. For core multilingual toxic comment classification we evaluate on both existing public benchmarks (Multilingual Toxic Comments Challenge) as well as a labeled real world dataset derived from live API traffic (Production-Multilingual). For robustness evaluation, we evaluate model performance when faced with code-switching (a subset of the multilingual datasets), obfuscation (obfuscated CivilComments), and distribution shift (zero-shot TweetEval [3] and CivilComments-WILDS [16]). We also evaluate the model on an identity term bias task based on [5]. For adapting to new types of toxicity, we evaluate finetuning performance on Covert Toxicity [19], and Hatemoji [15]. We refer the reader to Sections 5, 6, and 7 for detailed descriptions of these datasets.

4.2 Models

This section discusses the details about the major models we use in our experiments.

- **Perspective API** Jigsaw's public API for scoring comments for toxicity [13], prior to this work. It should be noted that many of the languages evaluated in the paper are not currently supported by the Perspective API, and as such Perspective results are omitted in such experiments.
- **Custom mBERT** We compare with a strong multilingual BERT [9] baseline that has been pretrained on PPC. The model uses a custom SentencePiece vocabulary of size 200K, created explicitly from the PPC corpus. We refer to this strong production baseline as Custom mBERT and consider it representative of a model highly tailored for the domain. The baseline model consisted of 768 dimensions, 12 layers, 12 heads, consistent with BERT-base [9]. The pre-training consists of MLM Loss and translation pairs with uniform masking at 15%. Pretraining was conducted for 125K steps with batch size of 32K.
- **Multilingual T5 (mT5)** - the state-of-the-art for multilingual natural language processing. mT5 is a pretrained T5 [23] model pre-trained on 100+ languages on the Multilingual C4 corpus.
- **UTC** and **UTC†** - our proposed models described in Section 3. For the vanilla UTC model, we use $d_{model} = 512$, $d_{ff} = 2048, d_{kv} = 64, N_{heads} = 8$. The number of encoder layers is set to 24 and the number of decoder layers is set to 6. For the learned tokenizer, we set the sequence length downsampling rate of 2, and set the convolution filter size to 5. The standard UTC is approximately 102M parameters when deployed in downstream applications. This model size was considered to ensure a fast serving latency. We also consider a larger (but still servable) UTC† model that is approximately $268M$ parameters where $d_{model} = 768$, $d_{ff} = 3072, d_{kv} = 64, N_{heads} = 12$ and number of encoder layers is set to 28. We denote this model as UTC†.

Fine-grained details on each specific baseline can be found in each individual experiment section. Please see Appendix A for additional reproduction details.

5 EXPERIMENTS: MULTILINGUAL

In this section we report the core multilingual toxic comment classification results of the work. With the exception of Perspective API, we finetune and evaluate all models outlined in Section 4.2, on each dataset. using a batch size of 512 until convergence.

5.1 Production-Multilingual

5.1.1 Dataset. A proprietary internal multilingual toxic comment classification training and evaluation set. This dataset is derived from live traffic that is sent to the production Perspective API (with `doNotStore` flag set to false), translated to multiple languages and is exclusively labeled offline by human annotators for toxicity. Given the nature of this dataset, this task represents the performance of our models in production, and is the most important metric by which we compare models. The training sets cover the languages of AR, CS, DE, EN, ES, FR, HI, HI-Latn, ID, IT, JA, KO, NL, PL, PT, RU, SV, ZH along with some low prevalence examples in a few other languages. The training data is 38 million records and is not balanced across languages with a heavy skew towards EN. The evaluation sets are limited to the languages covered in the experimental results : AR, CS, EN, HI-Latn, ID, JA, KO, NL, PL, PT, RU, ZH. This narrower set focuses in on languages where Perspective did not already have a production quality model (at the time of experimentation), plus English where significant Perspective usage comes from. The evaluation sets are roughly balanced in volume across languages and comprise 1.3 million records.

5.1.2 Results. Table 1 reports results on the Production-Multilingual dataset. Overall, UTC outperformed all baselines, with a small UTC model outperforming mT5_{base}, a model with more than twice the size w.r.t. number of parameters. While UTC does not perform as strongly on English as Custom mBERT, the main advantage of UTC is observed in many non-English languages.

5.2 Multilingual Toxic Comments Challenge

5.2.1 Dataset. In this section, we report experimental results on the public dataset featured in the Jigsaw Multilingual Toxic Comments Challenge (JMTCC) hosted by Kaggle. The competition was held in 2020 and comprises of 6 languages besides English: Spanish,

[2]https://developers.perspectiveapi.com/s/about-the-api-methods

Model	Params	Ar	Cs	En	Hi-Latn	Id	Ja	Ko	Nl	Pl	Pt	Ru	Zh	Avg
Perspective API	-	-	-	.974	-	-	-	-	-	-	-	**.907**	-	-
Custom mBERT	235M	.762	.881	**.982**	.832	.812	.649	.855	.842	.853	.878	.803	.925	.840
mT5$_{small}$	148M	.896	.913	.969	.962	.761	.881	.846	.726	.866	.856	.880	.976	.878
mT5$_{base}$	278M	.900	.925	.973	.967	.791	.887	.874	.934	.881	.850	.888	**.997**	.906
UTC	102M	.899	.925	.977	.954	.794	.867	**.938**	.940	.892	.864	.896	.975	.910
UTC†	268M	**.908**	**.934**	.977	**.968**	**.819**	**.896**	.916	**.947**	**.896**	**.888**	**.907**	.974	**.919**

Table 1: Experimental results on the Production-Multilingual dataset. We report AUC-ROC scores.

Model	# Params	AUC-ROC
Kaggle # 1	≈>5B*	**.9536**
Perspective API	-	.8770*
Custom mBERT	235M	.9104
mT5$_{small}$	148M	.9156
mT5$_{base}$	278M	.9239
UTC	102M	.9194
UTC†	268M	**.9367**

Table 2: Results on Jigsaw Multilingual Toxic Comments Challenge (JMTCC). *Turkish is not supported by Perspective API, and is omitted from this result.

French, Italian, Portuguese, Russian, and Turkish. While the evaluation data was multilingual, only English data was provided in the training set. Hence, it was common for participants to make use of translation data to augment the training set. We train our models on the translated data that was shared in the Kaggle discussion forums.

5.2.2 Compared Baselines. Aside from the baselines in Section 4.2, we also report results from the winners of the Jigsaw Multilingual Toxic Comment Classification Kaggle competition, although it is worth noting that our goal is to develop **single** standalone models that can feasibly be deployed in production. Meanwhile, the top Jigsaw Multilingual Toxic Comment Classification Kaggle submissions often involved aggressive ensembling, score scaling techniques etc, that are highly infeasible in practice. Nevertheless, we believe it is beneficial to evaluate how well our standalone single model fares compared to a strong highly engineered upper bound. Based on our interpretation of the Kaggle champion's entry, we estimate the number of model parameters to be $> 5B$ given that they ensemble multiple XLM large models (at least 300M parameters each) along with monolingual models.

5.2.3 Results. Table 2 reports results on the JMTCC dataset. Our results show that our best UTC† achieves 0.9367 AUC-ROC, outperforming all considered *single model* baselines, especially a strong state-of-the-art mT5 baseline. Notably, this result is only slightly worse than the top performing Kaggle #1 result which comprises XLM-Roberta ensembles, pseudo labelling and other commonly used techniques. We consider the result achieved by UTC† to be pretty compelling, given that this is a single model that can actually be used in production applications.

6 EXPERIMENTS: ROBUSTNESS

In this section we do no additional training and evaluate the fine-tuned models from Section 5.1 to evaluate the robustness of our proposed methods.

Model	#Params	JMTCC-CS	Production-CS
Perspective API	-	.7516	.7163
Custom mBERT	235M	.9243	.8106
mT5$_{small}$	148M	.9289	.8661
mT5$_{base}$	278M	.9393	.8730
UTC	102M	.9191	.8755
UTC†	268M	**.9446**	**.9023**

Table 3: Experiments on Code-Switching.

6.1 Code-Switching

The Code-Switching eval sets aim to identify theoretically more difficult multilingual user comments. Both bespoke evaluation datasets below are constructed by filtering the parent superset with the same criteria for multilingual comment identification: test examples are restricted to those where 2 or more languages are present. Samples are included if and only if >= 25% of the example content is identified to be in each of 2 or more languages using a language detection model.[3]. We use two subsets for code-switching based on Production-Multilingual and JMTCC datasets. Details on the breakdown of these code-switching datasets can be found in the supplemental material.

6.1.1 Compared Baselines. The same baseline models evaluated on the Production-Multilingual dataset were employed, including Custom mBERT, the comment domain multilingual BERT model, $mT5_{base}$ and $mT5_{small}$. Similarly, we also included an evaluation using the public Perspective API [13]. It should be noted that not all of the languages included in the code-switching datasets are listed as supported by Perspective and as such Perspective may be disadvantaged in this experiment. To specify a language for Perspective API, we use a language detection model to identify the primary language for each code-switch example. If the language is not supported by Perspective API, then we default to English.

6.1.2 Results. Table 3 reports our experimental results on code-switching evaluation sets. On both JMTCC-CS and Production-CS, UTC† outperforms the best, outperforming the mT5$_{base}$ baseline. In general we find that an off-the-shelf mT5 model also substantially outperforms the Custom mBERT model. Finally, we note that the Perspective API performs poorly since prior to this work, models were separately trained on individual languages and as such they are not well equipped to handle multilingual code-switching. One limitation of this experiment is the dominance of English and Latin based languages in the code-switching evaluation sets. We suspect that the byte level vocabulary in our Charformer model, is advantageous for understanding with character based languages especially in code-switching tasks. The preliminary results give evidence to this conclusion.

[3]https://github.com/google/cld3

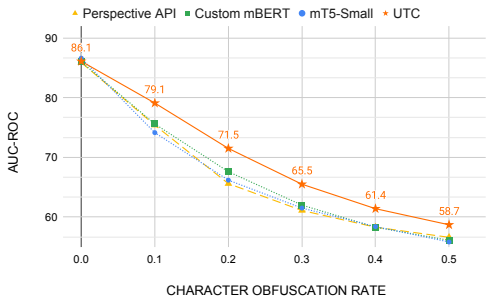

Figure 2: Zero-shot AUC-ROC on English-only CivilComments with 0-50% obfuscation rate.

6.2 Human-Readable Obfuscation

One common technique used to bypass toxicity classification models is to intentionally misspell words in a fashion that is understood by human readers, yet obfuscated to machine learning models [11]. Even though our proposed models are not explicitly trained on these types of adversarial examples, in this section we run zero-shot experiments on synthetically obfuscated data to evaluate model robustness in this area.

6.2.1 Dataset. We construct a synthetically obfuscated variant of the Civil Comments [5] test set. As the dataset is in English only, we manually construct a dictionary of valid substitutions for every letter in the English alphabet. Then for every alphabetical character in each example, we replace the character by a substitute with some probability, which we call the character obfuscation rate. If a character is chosen, then a substitute for the character is chosen uniformly at random from the list of valid substitutes for the character. Substitutes may be any other character or string that may still be readable as original character, e.g. "a" may be substituted with "4", "@", or "/\". Valid substitutions for vowels also include "*" or an empty string, which effectively removes information from the sequence. See Figure 3 for the comprehensive dictionary of valid substitutions, and Figure 4 for examples of text at various character obfuscation rates. Finally, please note that the construction of this dataset is not meant to comprehensively capture a realistic distribution of adversarial examples against toxicity classification models. Instead, we aim to create a controlled and sufficiently challenging dataset to serve as a point of evaluation between models.

6.2.2 Compared Baselines. We evaluate the zero-shot performance of Perspective API (prior to this work), Custom mBERT, mT5-small, and UTC (102M param.) on obfuscated Civil Comments, sweeping the character obfuscation rate from 0 to 50% in increments of 10%. Custom mBERT, mT5, and UTC are the same fine-tuned models from Table 1. No additional training was done for these experiments.

6.2.3 Results. Figure 2 plots the performance of all models across character obfuscation rates. In this experiment we observe that while all models have similar zero-shot performance when there is no obfuscation applied to the English only dataset, UTC outperforms every baseline at every other obfuscation rate greater than 0. Albeit, all models do decay in performance as obfuscation rate increases – however, this is expected as the models rarely see this type of obfuscation during training, and are not fine-tuned on any additional obfuscated data. This result echos a similar finding by previous byte-level models [34]. One natural question to ask might be: if fine-tuned, are the models able to learn to adapt to this type of obfuscation? As an additional result, we found that when fine-tuned on a 30% obfuscated version of Civil Comments, UTC was able to fully recover performance to 86.0 AUC-ROC, while mT5-small recovers to only 84.5 AUC-ROC (-2.1pt from the unobfuscated zero-shot result.) This result illustrates the value of the UTC inductive bias on this particular task.

Model	# Params	Macro. F1
RoBERTa-Retrained (Finetuned avg. of 3) [3]	125M	52.3
RoBERTa-Retrained (Finetuned best) [3]	125M	55.5
BERTweet (Finetuned best) [21]	125M	56.4
Perspective API (Zero-shot)	-	52.0
Custom mBERT (Zero-shot)	235M	51.8
$mT5_{small}$ (Zero-shot)	148M	53.9
$mT5_{base}$ (Zero-shot)	278M	57.7
UTC (Zero-shot)	102M	53.3
UTC† (Zero-shot)	268M	55.1

Table 4: Performance on TweetEval hate classification.

6.3 Distribution Shifts

Toxic content appears on many different surfaces in different forms, targeting many different types of people. In this section we evaluate the performance of our model on two different setups. In the first setup with TweetEval, we evaluate performance on a task with a different labeling process and domain focus. In the second setup, we evaluate the performance of the model on subpopulation shift using CivilComments-WILDS.

6.3.1 TweetEval Hate Classification.

Dataset. We evaluate our models on the TweetEval hate content classification test split [3], which is taken from the SemEval2019 Hateval challenge [4]. The task is to predict whether a given tweet contains hateful language targeted against any of two communities: women and immigrants. This task differs from our UTC pre-training and fine-tuning as it is purely Tweet focused and has been labeled to a different standard (i.e. hateful language.) As this is zero-shot evaluation, we do not do any additional fine-tuning for this experiment.

Compared Baselines. We evaluate our same baselines and models as Section 6.2: Perspective API, Custom mBERT, mT5, and UTC. For mT5 and UTC we evaluate both small and base sized versions. Additionally, we compare our zero-shot results against the current state-of-the-art for the task: RoBERTa Retrained and BERTweet. Both are English-only RoBERTa-based models which have been extensively pre-trained on a large corpus of English tweets, as well as finetuned on the corresponding TweetEval hate classification training set.

6.3.2 Results. Table 4 reports performance on TweetEval hate classification. All zero-shot baselines which were finetuned on Production-Multilingual data showed strong performance in this experiment, with multilingual mT5 and UTC in particular performing on par with an English-only RoBERTa model that had seen additional pretraining on a large Twitter corpus and finetuned on

Model	#Params	Avg Acc	Worst Acc	Gap
DistilBERT ERM [16]	66M	92.2	56.0	36.2
DistilBERT DRO [16]	66M	89.9	70.0	19.9
$mT5_{small}$	148M	94.0	81.8	12.2
UTC	102M	94.2	82.6	11.6

Table 5: Accuracy on CivilComments-WILDS dataset.

TweetEval hate classification training data. Higher results previously reported by [3] and [21] are only observed in "best run" performance where the model saw favorable variance. This experiment demonstrates the effectiveness of our methods in training domain shift robust toxicity classification models.

6.3.3 *CivilComments-WILDS.*

Dataset. Introduced in [16], this dataset augments the Civil Comments dataset with various demographic identities referenced in each example. The goal of this dataset is to evaluate for subpopulation shift: a setting where the model sees all domains (i.e. demographic identities) during evaluation as it does during training, but in different proportions. In particular, for CivilComments-WILDS a model is trained on all demographic identities available in CivilComments, but is evaluated on a single identity at a time – an extreme subpopulation shift. This is repeated individually for each subpopulation, with the aim of maximizing performance on the worst performing subpopulation. Following the original work, we perform our analysis on 8 demographic identities male, female, LGBTQ, Christian, Muslim, other religions, Black, and White. We report accuracy on the complete test split and the worst accuracy from the 8 subpopulations and the gap between the two to show that our model performs better on these subpopulation shifts. [16], showed the existence of a significant gap between the average in-distribution accuracy and the worst subpopulation accuracy. We additionally report this gap for our own evaluated models.

Compared Baselines. We evaluate small-sized mT5 and UTC models in this setting from Table 5.1. We compare to the highest performing DistilBERT results from [16] with respect to both average overall accuracy and worst-group accuracy (DistilBERT using empirical risk minimization, ERM, and group distributionally robust optimization, Group DRO, respectively.) As our models were multilingually fine-tuned, while DistilBERT fine-tuned on only English Civil Comments, for a fair comparison in this subpopulation shift setting our mT5 and UTC models are additionally fine-tuned on Civil Comments before evaluation. We do not use any robust optimization techniques when fine-tuning our models.

Results. Both mT5-small and UTC significantly outperform the baseline results from prior work on all metrics. Our models perform better overall and have almost half to a third smaller of a gap between average and worst group performance than DistilBERT with robust optimization techniques. Although the exact source of this gain is unclear, we posit that the extended amount of multilingual pre-training, pre-finetuning and greater model size may play a significant role here.

6.4 Identity Term Bias

Borkan et al. [5] outlined nuanced bias metrics, to be used in addition to overall model metrics such as AUC-ROC and [10], introduced synthetic, templated datasets for identifying unintended bias in toxicity models. Here we use these tools to evaluate our models for identity term bias.

6.4.1 *Dataset.*

We use a new multilingual version of the synthetic template bias evaluation data set, publicly released in 2021[4]. The examples in this dataset are generated from predefined templates with slots where different words (e.g. identity terms, adjectives, verbs, etc.) can be substituted for related terms in order to test for performance with regards to various subgroups (identities). To obtain a multilingual dataset for each of the 12 target languages we rely on a team of expert native speakers to construct these templates. The final generated dataset consists of $\sim 2M$ examples and has a balanced class distribution. Following [5] we report Subgroup AUC, Background Positive Subgroup Negative (BPSN) AUC, and Background Negative Subgroup Positive (BNSP) AUC for all subgroup-language combinations. Please see [5] for precise definitions of these metrics.

6.4.2 *Results.*

We evaluate model bias for UTC with mT5-small serving as a baseline comparison. Both models remain the same as from Table 1. We visualize the results in Figure 5 and 6 for all language splits. Note that to generate these visualizations we aggregate results with their corresponding English identity term (results with no corresponding English term are not shown here). Overall, the metrics remain strong at >.7 across languages, with UTC performing stronger on more subgroup-language combinations than mT5. As both models are finetuned on the same dataset, we see that the UTC inductive bias may play a role here. However, some subgroup-language combinations still require additional work. For example, there are some terms in Korean and Japanese that demonstrate unwanted bias, such as the BPSN AUC metric for the term *homosexual* where 동성애자 , and 同性愛者 have values ≤.5. This suggests that the non-toxic templates containing the identity term are yielding false positives, or rather the term is correlated with toxicity. As such, further explicit debiasing efforts are still needed.

7 EXPERIMENTS: ADAPTING

In this section we demonstrate that the fine-tuned checkpoints from Section 5.1 are highly adaptable to new types of toxicity by further fine-tuning our models on new challenging tasks.

7.1 Covert Toxicity

We conduct experiments on Covert Toxicity [19], a task of distinguishing if a piece of text contains nuanced toxicity such as microaggressions. We compare with Toxic-BERT and Covert-BERT baselines reported in [19]. We also finetune a monolingual (English only) T5 and mT5 base model as strong baselines.

7.1.1 *Results.*

Table 6 reports results on the CovertToxicity task. We show that UTC and UTC† achieves very competitive results outperforming both Toxic-BERT and Covert-BERT baselines. On this task, the performance of UTC† is competitive to $mT5_{base}$. Interestingly, the multilingual models (mT5 and UTC model) outperform the specialized monolingual English T5 model.

[4] https://medium.com/jigsaw/identifying-machine-learning-bias-with-updated-data-sets-7c36d6063a2c and https://github.com/conversationai/unintended-ml-bias-analysis/tree/2021-refresh

Model	AUC-ROC
Toxic-BERT	.520
Covert-BERT	.590
Monolingual T5$_{base}$	.599
mT5$_{base}$	**.607**
UTC	.604
UTC†	**.607**

Table 6: Results on CovertToxicity.

Model	# Params	Acc.	F1
Kirk et al. [15]	140M	87.9	91.0
mT5$_{small}$	148M	86.0	89.6
mT5$_{base}$	278M	86.8	90.5
UTC	102M	**90.0**	**92.8**
UTC†	268M	**90.8**	**93.3**

Table 7: Performance on English-only HatemojiCheck.

7.2 Emoji-based Hate

7.2.1 Dataset. The English-only Hatemoji dataset comprises of two splits: HatemojiCheck and HatemojiTrain. HatemojiCheck is a manually constructed labeled test suite of 3,930 short-form statements and whether they use emoji-based hateful language.

7.2.2 Compared Baselines. [15] showed that fine-tuning on HatemojiTrain greatly improves performance on HatemojiCheck. Following this, we evaluate our two highest performing models from Section 5.1, mT5 and UTC, by further fine-tuning them on HatemojiTrain then evaluating on HatemojiCheck. Note that these checkpoints have already been finetuned on Production-Multilingual. We use the validation split of HatemojiTrain to pick the best checkpoint for this additional fine-tuning. We additionally compare these results to the best results reported in [15], which is an English-only DeBERTa model optimized for the task.

7.2.3 Results. Table 7 reports results on HatemojiCheck. Even though UTC is multilingual, when finetuned UTC outperforms the best performing model from [15] by a significant margin. We posit that this gain may be attributed to the learned tokenizer, which may effectively update during finetuning to adapt to parsing emojis. On the other hand mT5$_{Small}$, under-performs the Kirk et al. baseline.

8 DEPLOYMENT RESULTS

On December 9th, 2021 Jigsaw launched support for 10 new languages[5] in the Perspective API [1], powered by UTC† (Table 1), making the model available publicly [6] (Note that languages[7] previously available within Perspective API were not impacted by this launch). UTC dramatically increased Perspective's capabilities, as it is the first model to reach our production standards for these 10 languages. All previous production candidates (CNNs and BERT-based architectures) had low overall performance, low performance on bias evaluations, or were too slow to serve for real-time usage.

The model was deployed smoothly with no operational issues, and as of writing this paper, the model averages ~15 QPS and ~200ms median latency (for the 10 newly launched languages only).

[5]Languages launched with UTC: Arabic, Chinese (Simplified), Czech, Dutch, Indonesian, Japanese, Korean, Polish, Hindi, and Hinglish (a mix of English and Hindi transliterated using Latin characters)

[6]https://developers.perspectiveapi.com/s/about-the-api-attributes-and-languages

[7]Languages not yet using UTC: English, French, German, Italian, Portuguese, Russian, and Spanish

Model	# Params	Finetune Steps/s
Byte-level T5 Base	200M	18.3
+ Charformer	134M	26.5
+ Regression Head (UTC)	102M	32.9
+ Increased Scale (UTC†)	268M	15.0

Table 8: Architecture ablation for Production-Multilingual finetuning speed on 64 TPUv3 chips (batch size 128, 512 byte input length.) These values correlate with serving latency.

In our load testing, we have observed that the smaller UTC (102M) model can achieve 45ms median latency at 1K QPS on a single TPUv2 chip with batching. We anticipate our production latency improving further as load increases as our serving infrastructure does not have to wait to accumulate requests for batching. In the future, we hope to migrate to using the smaller and faster UTC (102M) model, as well as explore further performance improvements. We also plan to transition the rest of the languages Perspective serves to UTC over time, as well as expand to additional new languages.

Overall, we consider our results impressive for a *byte-level* Transformer model of this size. We attribute the majority of this performance to the sequence length downsampling done by Charformer, as well as our removal of the decoder during finetuning, effectively reducing both our sequence length and depth dimensions by half respectively. We include an ablation study for the speed of our model with and without these modifications in Table 8. Performance may also be attributed to forgoing tokenization, a process that is hard to parallelize, in favor of Charformer GBST, which can run on specialized hardware (TPU). In addition to quality and performance, there are engineering advantages to our approach. We have found that having one model to support multiple languages significantly simplifies the maintenance of our service as now there are fewer models to maintain. Additionally, our experience forgoing tokenization echoes that of previous literature – we find that preparing models for production is simplified as we no longer need to coordinate model checkpoints with matching vocabularies. Given these results, we are looking forward to expand the usage of this approach in the future.

9 CONCLUSION

This paper presents Jigsaw's new generation of toxic comment classification models, which is currently deployed in production for 10 new languages in the Perspective API. We outline our approach in applying state-of-the-art token-free Charformer to the problem of toxic comment classification and the efficiency techniques we take to enable serving such a byte-level model in production. Through rigorous experiments on real-world and academic benchmarks we demonstrate the effectiveness our approach.

REFERENCES

[1] Tin Acosta, Alyssa Lees, Daniel Borkan, Jeffrey Sorensen, Alyssa Chvasta, Roelle Thorpe, and Lucy Vasserman. 2021. 10 New Languages for Perspective API. https://medium.com/jigsaw/10-new-languages-for-perspective-api-8cb0ad599d7c

[2] Azalden Alakrot, Liam Murray, and Nikola S Nikolov. 2018. Towards accurate detection of offensive language in online communication in arabic. *Procedia computer science* 142 (2018), 315–320.

[3] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 1644–1650. https://doi.org/10.18653/v1/2020.findings-emnlp.148

[4] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Minneapolis, Minnesota, USA, 54–63. https://doi.org/10.18653/v1/S19-2007

[5] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification. In *Companion Proceedings of The 2019 World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 491–500. https://doi.org/10.1145/3308560.3317593

[6] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, and Opeyemi Emmanuel Ajibuwa. 2019. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon* 5, 6 (2019), e01802. https://doi.org/10.1016/j.heliyon.2019.e01802

[7] Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial Bias in Hate Speech and Abusive Language Detection Datasets. In *Proceedings of the Third Workshop on Abusive Language Online*. Association for Computational Linguistics, Florence, Italy, 25–35. https://doi.org/10.18653/v1/W19-3504

[8] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM '17)*. AAAI, Palo Alto, CA, 512–515.

[9] 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[10] 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 (AIES '18)*. Association for Computing Machinery, New York, NY, USA, 67–73. https://doi.org/10.1145/3278721.3278729

[11] Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. All You Need is "Love": Evading Hate Speech Detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security (AISec '18)*. Association for Computing Machinery, New York, NY, USA, 2–12. https://doi.org/10.1145/3270101.3270103

[12] Abigail Z. Jacobs, Su Lin Blodgett, Solon Barocas, Hal Daumé, and Hanna Wallach. 2020. The Meaning and Measurement of Bias: Lessons from Natural Language Processing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20)*. Association for Computing Machinery, New York, NY, USA, 706. https://doi.org/10.1145/3351095.3375671

[13] Google Jigsaw. 2017. Perspective API. https://www.perspectiveapi.com/. Accessed: 2021-02-02.

[14] Brendan Kennedy, Mohammad Atari, Aida M Davani, Leigh Yeh, Ali Omrani, Yehsong Kim, Kris Coombs, Shreya Havaldar, Gwenyth Portillo-Wightman, Elaine Gonzalez, and et al. 2018. The Gab Hate Corpus: A collection of 27k posts annotated for hate speech. https://doi.org/10.31234/osf.io/hqjxn

[15] H Kirk, B Vidgen, P Röttger, and SA Hale. 2021. Hatemoji: A test suite and adversarially-generated dataset for benchmarking and detecting emoji-based hate. (2021).

[16] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. 2021. WILDS: A Benchmark of in-the-Wild Distribution Shifts. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Marina Meila and Tong Zhang (Eds.), Vol. 139. PMLR, 5637–5664. https://proceedings.mlr.press/v139/koh21a.html

[17] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Brussels, Belgium, 66–71. https://doi.org/10.18653/v1/D18-2012

[18] Keita Kurita, Anna Belova, and Antonios Anastasopoulos. 2019. Towards robust toxic content classification.

[19] Alyssa Lees, Daniel Borkan, Ian Kivlichan, Jorge Nario, and Tesh Goyal. 2021. Capturing Covertly Toxic Speech via Crowdsourcing. In *Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing*. Association for Computational Linguistics, Online, 14–20. https://www.aclweb.org/anthology/2021.hcinlp-1.3

[20] Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation (FIRE '19)*. Association for Computing Machinery, New York, NY, USA, 14–17. https://doi.org/10.1145/3368567.3368584

[21] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 9–14. https://doi.org/10.18653/v1/2020.emnlp-demos.2

[22] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 145–153. https://doi.org/10.1145/2872427.2883062

[23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html

[24] Tharindu Ranasinghe, Marcos Zampieri, and Hansi Hettiarachchi. 2019. BRUMS at HASOC 2019: Deep Learning Models for Multilingual Hate Speech and Offensive Language Identification.. In *FIRE (Working Notes)*. CEUR, Aachen, Germany, 199–207.

[25] Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. 2019. The Risk of Racial Bias in Hate Speech Detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 1668–1678. https://doi.org/10.18653/v1/P19-1163

[26] Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. 2018. Mesh-TensorFlow: Deep Learning for Supercomputers. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 10435–10444.

[27] Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, Cambridge, Massachusetts, 4596–4604. https://proceedings.mlr.press/v80/shazeer18a.html

[28] Guizhe Song, Degen Huang, and Zhifeng Xiao. 2021. A Study of Multilingual Toxic Text Detection Approaches under Imbalanced Sample Distribution. *Information* 12, 5 (2021), 1–16. https://doi.org/10.3390/info12050205

[29] Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[31] Cindy Wang and Michele Banko. 2021. Practical Transformer-based Multilingual Text Classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*. Association for Computational Linguistics, Online, 121–129. https://doi.org/10.18653/v1/2021.naacl-industry.16

[32] 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*. Association for Computational Linguistics, San Diego, California, 88–93. https://doi.org/10.18653/v1/N16-2013

[33] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1391–1399. https://doi.org/10.1145/3038912.3052591

[34] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *CoRR* abs/2105.13626 (2021). arXiv:2105.13626 https://arxiv.org/abs/2105.13626

[35] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 483–498. https://doi.org/10.18653/v1/2021.naacl-main.41

[36] W Yin and A. Zubiaga. 2021. Towards generalisable hate speech detection: a review on obstacles and solutions. *PeerJ Computer Science* 7, 5986 (2021), 1–38. https://doi.org/10.7717/peerj-cs.598

A REPRODUCTION DETAILS

A.0.1 Implementation. Our UTC model is implemented in Mesh TensorFlow[8] [26], a wrapper over TensorFlow API that enables distributed model parallelism, along with the T5 library[9]. For Charformer [29], we use the official implementation[10] released by the authors. The overarching model architecture follows the T5.1.1 setup using T5-styled relative attention biases instead of position embeddings.

A.0.2 Optimization and Training Details. This section describes the general setup for our pretraining and finetuning experiments. Dataset specific details are deferred to respective sections. For both pretraining and finetuning, we use the Adafactor optimizer [27]. During pretraining, we use a learning rate equal to the inverse square root of the current training step following [23]. Finetuning is performed using a fixed constant learning rate of 10^{-3}. We apply a dropout of 0.1 during finetuning. Pretraining is conducted with 64 TPU-v3 chips and finetuning is typically conducted with 16 TPU-v3 chips. Pretraining generally takes about 3-4 days to complete.

A.0.3 Reproducibility. Our model is currently available via the production Perspective API[11] for Arabic, Chinese (Simplified), Czech, Dutch, Hindi, Hinglish (a mix of Hindi and English), Indonesian, Japanese, Korean, Polish, and Russian for the "TOXICITY" attribute. Access to the model for English is also released under the "TOXICITY_EXPERIMENTAL" attribute. Even though the interface for the API requires specification of a language, all requests are routed to a single UTC† model.

A.0.4 Dataset Details. Here we include figures to further provide some details about selected datasets used.

Language	Prevalence	Proportion
en	1658	99%
pt	474	28%
es	356	21%
it	353	21%
fr	307	18%
ru	147	9%
ar, bg, co, de, el, hi, ka, ja, tr, zh	33	< 2%
total	1664	100

Table 9: JMTCC Code-switching Eval: Language breakdowns of filtered code-switching examples. Note: as an example contains multiple languages, the total does not correspond to the sum of the columns here.

[8]https://github.com/tensorflow/mesh
[9]https://github.com/google-research/text-to-text-transfer-transformer
[10]https://github.com/google-research/google-research/tree/master/charformer
[11]https://developers.perspectiveapi.com/s/docs

Language	Prevalence	Proportion
en	31101	97%
pt	12499	39%
hi-Latn	8520	26%
id	6589	20%
ru	636	2%
ar	601	2%
es	417	1%
nl	396	1%
de	353	1%
pl	300	1%
fr, ja, it, zh, hi, da, ur, cs, cv, fy, ko, +	2971	11%
total	32196	100

Table 10: Production-Multilingual Code-switching Eval: Language breakdowns of code-switching examples. Note: as an example contains multiple languages, the total does not correspond to the sum of the columns here.

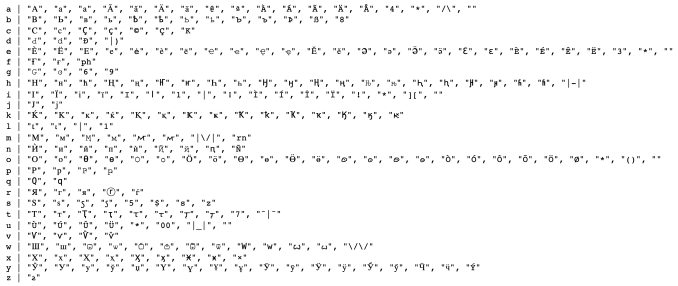

Figure 3: Full list of substitutions used for obfuscation experiments in Section 6.2.

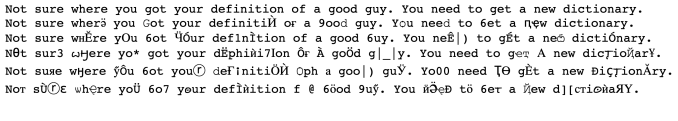

Figure 4: Examples of obfuscation of a sentence sampled from Civil Comments, for character obfuscation rate from 0 (top) to 50% (bottom).

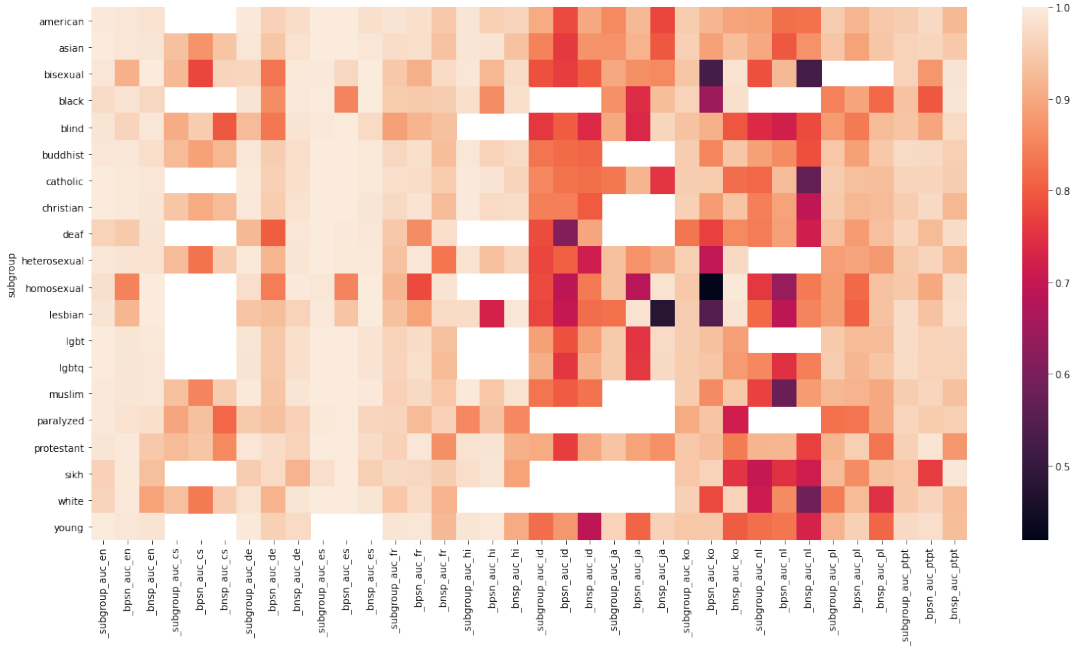

Figure 5: mT5 Unintended Bias Metrics, AUC, BPSN, BNSP per Language on template identity eval set for a subset of localized identity terms

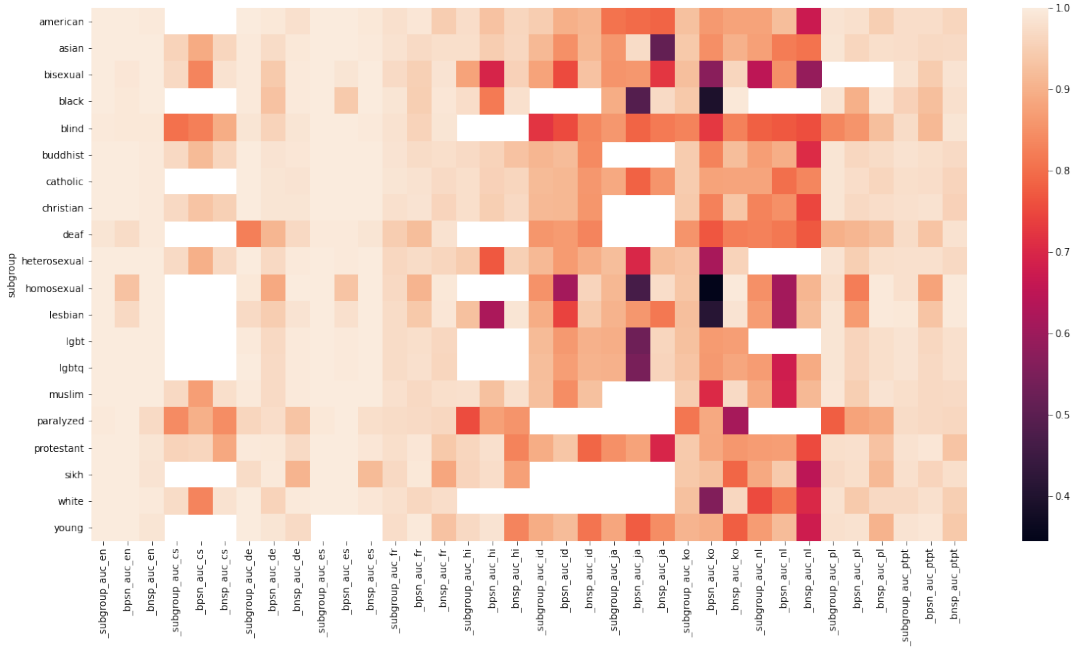

Figure 6: UTC Unintended Bias Metrics, AUC, BPSN, BNSP per Language on template identity eval set for a subset of localized identity terms