Investigating the Application of Knowledge Sharing to Improve Hate Speech Modelling

Erin Aho[1]

[1]University of Nottingham, ppxea1@nottingham.ac.uk

August 29, 2022

Content Warning: This paper contains examples of hateful language that some may find offensive. In some areas, examples from the dataset are used to provide a better context for discussions.

Sections that contain offensive language are marked with [CW].

Abstract

Creating performant NLP hate detection models is a difficult task, due to the scarcity of labelled datasets, and the complexity inherent to the problem itself. A rigorous understanding of hate speech in an online medium requires a grasp of a number of NLP sub-problems, including sentiment detection, negation, slang, and an understanding of paralinguistic features, like hashtags and emojis. While advances have been made in methods that help to mitigate over-reliance on identity terms, open-source models are still hampered by the size of available labelled datasets to train models on. This paper aimed to determine if pre-training models on related NLP sub problems with much larger datasets can help to train hate speech detection models. A model framework was created that could be initialised with any RoBERTa-based model and trained with attention-entropy regularisation on the HateXplain dataset, and different pre-trained models were compared. A RoBERTa model pretrained by Cardiff-NLP to detect sentiment on a large Twitter dataset outperformed an emotion detection model as well as a base RoBERTa model checkpointed to work on Twitter. Limitations and further work to better understand the limitations of subtask transfer learning are discussed.

1 Introduction

Hate Speech, defined by the United Nations Human Rights Council as "any kind of communication... that attacks a person or group on the basis of who they are" [1], has been a major problem in online social media platforms since their inception. Hate speech not only causes direct psychological harm to recipients [2], but on a societal scale, has been found to incite intolerance and animosity. In extreme cases, online hate speech has been a precursor to mass violence and genocides, as is the case in the genocide against the Rohingya people within Myanmar [3]. Mitigating the impact of hate speech on social media platforms is of great societal importance, but because of the enormous volume of content on these sites, it is impossible to rely solely on human moderation.

Hate speech detection is a very difficult task, even for trained human labellers. Labelling text as hateful is much more ambiguous than other NLP tasks, such as sentiment classification or part-of-speech tagging. Not only is there no single definition for what constitutes hate speech, there is often a sliding scale for determining if speech is offensive or hateful. Two different people may have different thresholds and views of what constitutes hateful, even when provided a rigid definition to use. Hate speech also quickly evolves and morphs, either via oppressed groups reclaiming slurs for non-hateful usage (e.g. LGBTQ+ groups reclaiming the word "queer") [4], or through extreme groups creating new toxic slang and slurs, as Bogetić has observed within the manosphere and incel communities [5].

2 Related Work

2.1 NLP modeling

NB: A thorough description of the theory behind the model used in this experiment is available in Section 3.3. This section is used to give a high level overview of the current state of NLP Hate Speech modelling.

Hate speech detection is a problem within the broader field of NLP. Many popular NLP model architectures have been applied to hate speech detection, with varying degrees of success. Gaydhani et al. [6] experimented

with Support Vector Machines (SVMs), and Bojkovsky and Pikuliak [7] found good results using word embeddings and models based on bidirectional LSTM models, as well as CNN—LSTM hybrids. Silva et al. [8] found that when comparing popular model types on the HatEval dataset, Transformer models generally performed significantly better, both in English and Spanish toxic language detection.

Transformers were originally proposed by Vaswani et al. in 2017 [9], and immediately produced cutting-edge results in a number of NLP challenges. Unlike RNN and LSTM models, Transformers make use of successive self-attention layers to produce a sequence to sequence mapping, instead of recurrent or convolutional layers. Self attention layers bring two main benefits over RNN and LSTM layers: firstly, they are computationally more efficient (fewer operations per layer, and are significantly more parallelizable); secondly, as each position a self-attention layer is simultaneously connected with all positions in a sequence, they are much better at representing long-range relationships between tokens in a string than an RNN layer or LSTM.

Since the original paper was published in 2017, Transformers have been used in many state of the art NLP models [10, 11]. One of the most widely used extensions of the original transformer is the Bidirectional Encoder Representations from Transformers (BERT), originally published by Devlin et al. in 2019 [12]. BERT extends the original transformer architecture by including a variable number of self-attention heads and encoders, with 12 of each in the BERT model. BERT focuses on bidirectional self-attention, which improves the model's ability to recognize the context of all tokens in both directions in a sequence, whereas other models (such as GPT) have a left-to-right architecture, causing much slower convergence and leads to sub-optimal results.

BERT, like other Transformer models, consist of both a tokenizer, that transforms a string of text into tokens, and a main model. The original BERT paper used a WordPiece embedding, introduced by Wu et al in 2016 [13], but many models now choose to use a Byte Pair Encoder (BPE) as a tokenizer. BPE maps words from a string into sub-word tokens based on the frequency of sub-word pairs in a training corpus. This method was originally proposed as a lossless text compression method by Gage in 1994 [14], but has only recently become popular as a method of embedding text.

BERT was originally trained on a masked-language model pretraining objective, and next-sentence prediction, on the BooksCorpus(800M words) [15] and the English Wikipedia corpus (2500M words), and immediately was able to achieve state-of-the-art results in a number of NLP tasks. Liu et al. [16] further pretrained the original BERT model with optimized hyperparameters to reach an improved set of pretrained weights for BERT, called RoBERTa. Many teams have released pretrained variants of BERT and RoBERTa for different NLP tasks. Cardiff-NLP used semi-supervised fine-tuning of the RoBERTa base checkpoint on a large Twitter corpus to produce RoBERTa-Tw, and used the RoBERTa-Tw checkpoint to further fine-tune a number of models to make predictions on labelled datasets for common NLP challenges, like sentiment, or emotion [17].

Zhou et al. [18] created a novel model architecture that simultaneously trained a mixture of experts of Transformers to make simultaneous predictions on two different tasks, one prediction of hate speech and one of sentiment, on two independent datasets. They found that because of the strong correlation between sentiment and presence of hate speech, their sentiment-knowledge sharing model was able to outperform other baseline models in predicting hate speech. The hypothesis in this paper is that by initialising a model with weights that were pretrained on a related subtask, such as sentiment, or emotion, the model will be able to be trained faster, and provide better final results, as it should be able to transfer the knowledge learnt from a larger dataset on a sub-task, to the task at hand.

2.2 Model biases

A wide variety of approaches have been suggested to overcome these lexical and dialectic biases. Zhang et al [19] proposed a method of adversarial learning that has shown strong results in minimising inherent biases that may exist in a dataset, from being present in the predictions of a model trained on that dataset (e.g, training a model to determine creditworthiness on historic data will train a model that generates discriminatory results).

Zhou et al [20] demonstrate that many of the above model architectures have an overreliance on detecting single identity terms, which lead to biases in training that can cause the model to be unreliable in production environments. Zhou et al describe how common open-source toxic speech detection models will mislabel examples, partly because of biases that may exist in training datasets. They identify sources of lexical and dialectical bias, such as models labelling text written in African American Vernacular English (AAVE) as hateful more than other dialects, or that using a commonly targeted group identifier, even in a neutral or positive way, will be labelled as toxic (e.g. "I identify as a black gay woman" may be labelled as hateful, while "I identify as a straight white man" may be labelled as normal). Vidgen et al. [21] propose a method of training a model to better understand edge cases by creating a human-and-model loop that dynamically adds labelled cases to a training dataset that contain linguistic features that the model struggled with in previous rounds and were able to show that the model in the final round had better F1 and AUROC scores. The models also performed significantly better in HateCheck, a suite of challenging functional tests for hate speech detection models [22].

A shortcoming of the above methods is that they require a-priori knowledge about a set of "identity terms" that create biases in a final model. Attanasio et al. [23] proposed Entropy-based Attention Regulation (EAR)

as a method of mitigating biases that occur due to models overfocusing on identity terms in toxic language detection. They found that by adding a component to the loss function that penalises the attention layers when they focus too heavily on singular terms, they were able to improve AUROC and F1 scores after training a BERT-based model to detect misogyny in both Italian and English datasets.

2.3 Data

Because of the challenging nature of labelling hate speech, most public labelled hate speech datasets are relatively small. Many available datasets range in size between 1000 and 15000 instances [24, 25, 26]. Founta et al. [27] put forward a crowdsourced method of labelling social media dataset and produced a set of data with 80000 labelled instances. The scope of Founta et al's labelling was broader than just hate speech, including labels to identify spam, and abusive text. In 2022, Matthew et al. [28] created a dataset specifically for identifying hate speech, with 20148 instances, and a roughly even split amongst each label, "Normal", "Offensive" and "Hateful". This dataset also had a focus on explainability, with annotators labelling the target group of the hate and the regions of the text that prompted them to assign that label.

3 Method

The hypothesis being tested was whether a transformer model that was pretrained on a relevant subtask for hate speech detection would perform better than one pretrained on a more generic subtask. Based on looking at the current best performing models in literature, the decision was made to use a BERT type model with a small fully-connected head on top for classification. It was also decided to experiment with an entropy-based attention regularisation component in the loss function, to see if the results from Attanasio et al. could be used to further improve the model performance.

To compare performance attained by initialising from different pretrained model checkpoints, a model was initialised with one of the sets of weights from a pretrained model. The fully connected head would be trained to start with the transformer model frozen, and then the full model would be unfrozen and fine tuned. Each model would be hyperparameter tuned, and the best performing version of each model would be evaluated on the same hold-out data partition.

3.1 Dataset

The decision was made to make use of Matthew et al's HateXplain dataset [28], because of the balanced nature of the dataset for hate speech detection and the added tags the authors included which could be used to improve explainability. In the original paper, the authors describe in detail their methodology for data collection and labelling. A set of human annotators were first trained on data labelled by the authors and then were tested to make sure they were able to apply the labels consistently with the authors specifications. The best annotators were selected and used to label the full dataset. Each instance was labelled by three different annotators, and the full set of annotations was provided in the dataset.

To convert the annotations into a format that would be more easily parsed by the machine learning model, the majority annotation of the three annotators was used. In the full dataset of 20148 instances, there are 919 instances which did not have a majority decision (i.e even split across each label). In the original HateXplain paper, the decision was made to remove these instances from the dataset. In this paper, these instances were included during training and given a new label: "Split". It may be useful to understand ambiguous text cases but were discarded when evaluating model performance on a validation set, so that the results are comparable to those obtained in the original paper.

The decision was made to not use the specific attention labels the annotators provided in the original dataset, and not use the identity type labels during training either. While this information was found to be useful in the original paper, because these labels are comparatively uncommon, the decision was made to focus only on the ability for the model in this paper to learn from the text, and how adding in attention entropy or using weights pretrained on different tasks would affect performance, as it would be more generalisable to focus exclusively on the text.

The original data was sampled from both Twitter, and a similar microblogging site called Gab. The text in the dataset is unprocessed, and so required reprocessing to normalise the data before being put into a model.

3.2 Preprocessing [CW]

Preprocessing of text data is very important in training NLP models, particularly for smaller datasets, or noisier data such as text sourced from social media platforms. In general, noisy data, if not handled correctly, can lead to poor model performances, as the signal is obfuscated by irrelevant data. In NLP problems, noise usually presents in the forms of typos, slang, some kinds of punctuation, and entries in other languages. On social media

datasets, other forms of noise can also be common, such as presence of URLs and user-mentions, contractions, emoticons or emojis, and elongated characters, which are commonly used to provide emphasis (e.g elongating "I love candy" can be elongated to "I looooooooooooove candy").

In hate speech datasets, an interesting extra form of noise is often found. Bhalerao et al. [29] describe the phenomenon of adversarial text perturbation, where users will intentionally misspell terms in order to avoid detection by automated moderation. In their paper, they mainly focus on users trying to engage in spam behaviour without detection, but users trying to engage in toxic behaviour act in the same way. In their paper, they highlight 10 main perturbations that were common in their dataset, and defences that can be used to combat them. Examples they outline included:

- Tandem character obfuscation (e.g. "/\pple" instead of apple)
- Replaced unicode (e.g. "øråñgës" instead of oranges)
- Vowel repetition and deletion (e.g "pls likee nd sharee" instead of please like and share)

There are several methods that can be used to handle these attempts at skirting detection. In this paper, it was decided to use a mapping of known uncommon slurs and neologisms to ones that are present in the original training corpuses. One of the adversarial methods, dog-whistles, are particularly hard to work around. Dog-whistles are seemingly innocent messages that carry a hidden meaning for a particular group. By their nature, they are intended to carry double meanings and deniability, and are hard to identify consistently without having a list of them a priori.

Naseem et al. [30] conducted an investigation into what methods of text pre-processing improve the final performance of hate speech models. The methods they found improved performance were:

- Removal of unicodes, URLs, user-mentions, and hashtag symbols
- Replacing emoticons and emojis with words (e.g 😄 -> happy)
- Replacing slang and abbreviations
- Correction of spelling mistakes
- Expanding contractions
- Replacing elongated words
- Removing punctuations
- Lower-casing of words
- Word segmentation
- Removing numbers
- Removing stop-words
- Lemmatization

For the purposes of this project, a few of these methods were not necessary to employ. The RoBERTa model used as a baseline in this project includes a byte-pair encoding tokenizer (BPE), so word segmentation and lemmatization are not required, because the BPE tokenizer already breaks up large words or joint words (such as hashtags) into sub-word tokens. As described by Qiao et al. [31] in their paper investigating the behaviours of BERT, they found that while attention on stop words was non-zero, removing them during preprocessing did not improve the performance of the model. For the most part, with transformer models, the preprocessing done to the text is minimal, and so removing contextual words is likely to reduce the performance. But, because of the noisiness of social media texts when compared to the Wikipedia and Book corpuses that BERT-base has been trained on, there is still a good deal of preprocessing that can be done that should improve the performance of the model by strengthening the signal and reducing the noise that surrounds it.

The full data preprocessing pipeline used in this paper was as follows, with examples of what a text looks like at each stage can be found in table 1.

Step	Process name	Example after process
0	Text input	That's 1 fuuuckin fagït LMAO 😂 !!!!
1	Remove emojis	That's 1 fuuuckin fagït LMAO!!!!
2	Lower case text	that's 1 fuuuckin fagït lmao!!!!
3	Remove accents and non-ascii text	that's 1 fuuuckin fagit lmao!!!!
4	Shorten whitespace to 1 and normalise apostrophes	that's 1 fuuuckin fagit lmao!!!!
5	Replace slang terms	that's 1 fuuuckin faggot laughing my ass off!!!!
6	Replace contractions	that is 1 fuuuckin faggot laughing my ass off!!!!
7	Replace elongated words	that is 1 fuckin faggot laughing my ass off!!!!
8	Convert numbers to words	that is one fuckin faggot laughing my ass off!!!!
9	Remove punctuation	that is one fuckin faggot laughing my ass off
10	Spell correct	that is one fucking faggot laughing my ass off

Table 1: A table showing the preprocessing pipeline used in this project to clean text before being encoded.

Non-ascii characters and accents were removed from the text using functions from scikit-learn's feature extraction module. Ekphrasis was used for contraction unpacking and for adjusting elongated words. The base slang dictionary from Ekphrasis was also used to map terms from slang to their broader meaning, and adjustments were made to map more niche and rare slurs to more common ones that shared meaning, so that the meaning could be better understood by the model.

It has been noted that important context can be contained in punctuation and emojis, particularly in hate speech detection. Punctuation and emojis have historically been used for "dog-whistles", such as the (((triple parenthesis))) used by anti-semites to signify Judaism without being overt [32]. While removing punctuation and emojis does remove some context from the strings, it does also significantly reduce the noise in the dataset, and was shown by Naseem et al.[30] to improve model performance overall.

One final step done in preprocessing the dataset was to remove non-English entries. Looking through the dataset, there were 31 entries that were written in non-English languages. These were mostly Spanish, with a few German and French entries. Different languages follow very different lexical rules and grammatical structures and very rarely share the same words They may even use the same strings of characters to represent different concepts. As such, unless a model has been explicitly trained to work in multiple languages, including entries written entirely in a different language in the training or testing dataset will skew results and invariably harm performance. Because this, these entries were removed from the dataset.

To finally interface the preprocessed datasets with the model API created as part of this project, the preprocessed datasets were saved, and a class was constructed to could sample the dataset in batches, and split it into train, test, and validation sets with a fixed random seed, such that the same splits could be used across any experiment.

To minimize duplicated work during training, the dataset was tokenized in full at the start of the process. Each initialised model weight set was trained on a unique BPE tokenizer with different string mappings. While transformer models can accept variable input sizes to make use of efficient parallelization, the instances within a batch must be grouped into a tensor and so must have a fixed input size. To tackle this, it is common to pad smaller tokenized string with an empty padding token, such that all of the tokenized arrays are as long as each other. They also provide an $attention_mask$, which is a binary array that is used to inform the model which tokens in the sequence are padding and can be ignored. For each model being tested, the dataset was tokenized, and padded so that the length of each instance was fixed, and the token arrays as well as the correlated attention masks, were saved in a format that could be read efficiently during training and evaluation.

3.3 Model Theory

As discussed in the introduction, the current state-of-the-art NLP models are mostly based on the Transformer architecture introduced by Vaswani et al in 2017. Transformer models have largely surpassed CNN or RNN based models for nearly all tasks, both in terms of efficiency in training and predictive power.

To understand the model architecture used in this experiment, it makes sense to first describe in more detail how Transformer models differ from CNN or RNN type models, and explain in more detail some of the terms that will be used in subsequent discussions.

In the original paper put forward by Vaswani et al, they envisioned the Transformer as mostly a sequence transduction architecture. As with most state-of-the-art sequence transduction models, Transformers can be split into an encoder, and a decoder, and a tokenizer/embedding function. The tokenizer is used to embed an input sequence into a vector of tokens which represent the input sequence (usually as integers), the encoder maps the input sequence of those tokens, x, into a latent representation, z, while the decoder maps this latent representation z to an output sequence, y, which can be detokenized to return it to the same format as the

original sequence.

One of the core components of both the encoder and decoder stack are multi-head attention mechanisms. To break this down, "attention" is a way of describing where focus is put in a sequence for making a prediction. Self-attention considers how much relation each part of the sequence has to other parts of the sequence. Within the transformer paper, attention is calculated based on three vectors, called Key, Value, and Query (K, V, Q).

The source of Q, V and K depends on which type of attention mechanism is being discussed. In the original Transformer architecture, both self-attention and encoder-decoder attention layers are used. In self-attention layers, Q, V and K are all outputs of the previous layer (or the sequence embedding for the first self-attention layer), and in the encoder-decoder attention layer, V and K are outputs of the encoder, and Q is the output of the previous decoder layer. This is similar to how RNNs use a hidden state to track the prior entries in a sequence. A simplified, high-level diagram showing the original architecture is given in figure 1.

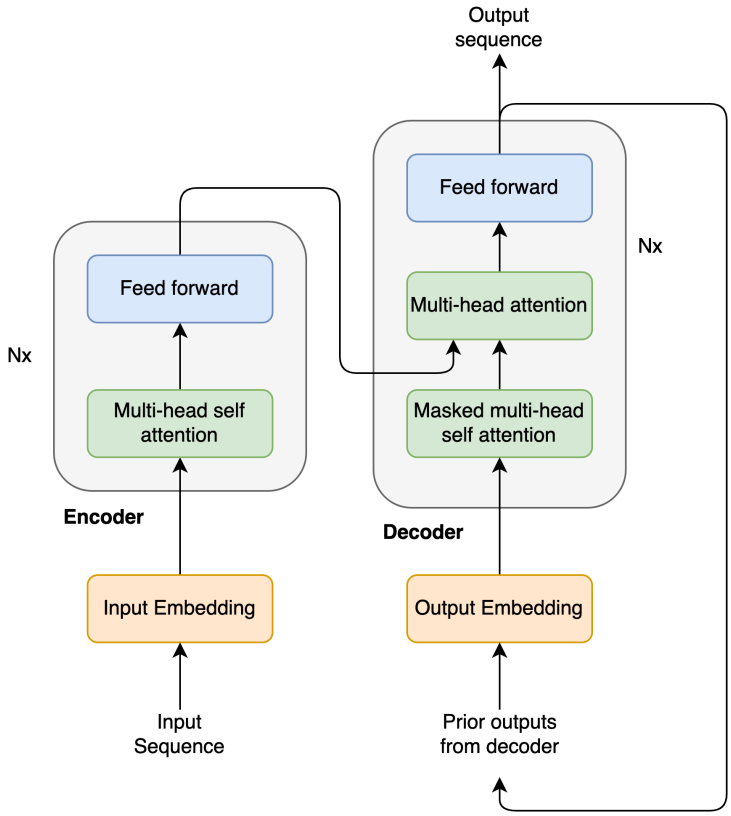

Figure 1: Diagram showing high level architecture of original Transformer model. Normalisation and residual connections within the encoder and decoder are omitted for simplicity. The N outside of the encoder and decoder refer to the number of encoders or decoders are present in the encoder/decoder stacks, which in the original Transformer paper was 6 for each.

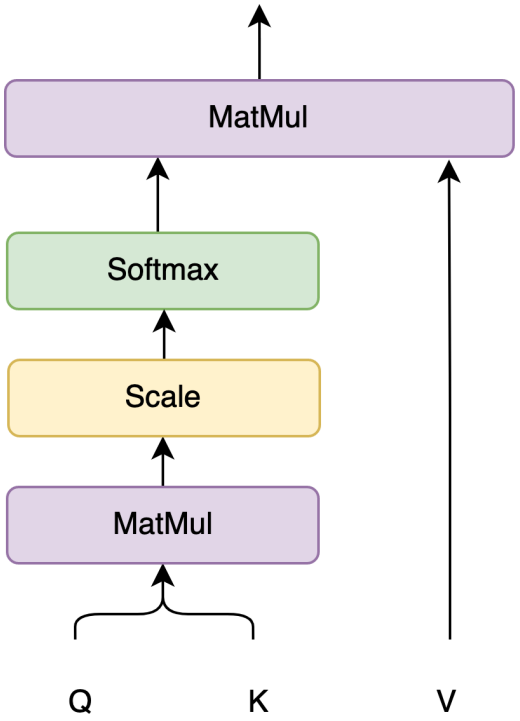

Figure 2: A figure showing a schematic view of the Scaled Dot-Product attention mechanism used in the transformer model. In figure 3, the scaled dot-product attention section refers to this mechanism.

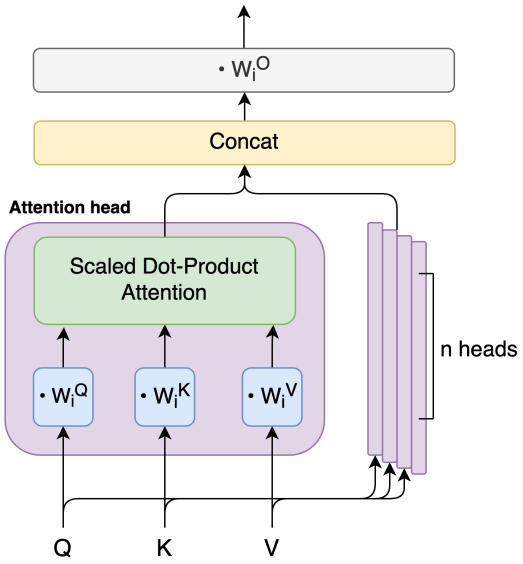

Figure 3: A figure showing the way that multi-head attention functions for input Q, K and V vectors. The attention head is repeated n_{head} times, with a different set of weight matrices in each. This allows each attention head to focus on a different aspect of the underlying signal, and allows for improved parallelisation during training. This is expressed in equation form in eqs 2 and 3.

A detailed view of the attention mechanism is shown in figures 2 and 3.

The authors of the original paper found that instead of using a single weighting and mapping step, they could achieve better performance by introducing "multi-head attention". In multi-head attention, there are a number of independent "attention heads". Inside each head, Q, V and K are multiplied by a weight matrix specific to that head (W_i^Q, W_i^K, W_i^V, where i denotes the index of the head), and then are passed through the scaled dot-product attention mechanism shown in figure 2. After the results from each attention head are calculated, the resultant Q, K and V matrices are concatenated, and multiplied by a final weight matrix, W_i^O, before being passed onto the next layer. Each of the weight matrices for each attention head must be learnt during training.

Expressed algebraically, the scaled dot-product attention is written as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_K}})V \tag{1}$$

Where Q, K, and V are the input Query, Key and Value matrices, and d_K refers to the dimensionality of K.

Given this definition of Attention, the total result of the multi-head attention layer is

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\,\text{head}_1, ..., \text{head}_\text{h}\,)\,W^O \tag{2}$$

$$\text{where head}_i = \text{Attention}(\,QW_i^Q,\ KW_i^K,\ VW_i^V\,) \tag{3}$$

Where Attention is defined in 1, and W_i^Q, W_i^K, W_i^V, and W_i^O are weight matrices with dimensionality fixed to be compatible with Q, and K, and the number of heads h.

In the paper, in order to stop the decoder from considering future positions in the sequence, the scaled dot-attention product in the decoder also includes a step where the Q and K components may be masked.

This self-attention mechanism allows for each position in the sequence to be connected to each other simultaneously, and drastically decreases the required number of sequential operations for all pairs of input/output positions to be connected.

The encoder in the original transformer paper was composed of $n = 6$ stacked identical layers, each composed of a multi-head attention mechanism with $h = 6$ attention heads, followed by a simple feed-forward network, with residual connections and layer normalisation. The decoder stack similarly is consisted of 6 stacked layers, with a masked self-attention, an encoder-decoder attention mechanism, and a feed-forward network, also with normalisation and residual connections.

While this transformer architecture was able to achieve state-of-the-art scores in a number of NLP tasks with significantly less training time, it has since been surpassed by extensions of the original design. In 2019, Devlin et al. released a paper detailing the Bidirectional Encoder Representation from Transformers (BERT)[12]. BERT was designed specifically to understand natural languages, instead of the more general goal of Transformers to be able to be used in any sequence-to-sequence task.

BERT uses the same concepts and design principles as the original Transformer, but forgoes the decoder stack, and instead has only encoder blocks. This allows the model to be more computationally efficient (because no recurrence is needed). It also allows the model to be fully bidirectional (because there is no need for the masking that is present in decoder), and importantly, allows for a single, unified architecture to be used across any representational task.

The BERT paper also uses a method of embedding inputs that is different from the original paper. Develin et al. use WordPiece embeddings from Wu et al.[13] to encode subword tokens. The final representations that are put into the model are a sum of the token, a position embedding that encodes the position of the token in the input sequence, and a segment embedding that can be used to encode which sequence the token is a member of, in the case of the input containing multiple segments (such as Question and Answer segments). However, for the the purposes of this paper, and with the text being classified, only single segment inputs were used.

The input also contains a special classification token at the start, [CLS], which is used as a placeholder for the aggregate sequence representation in classification tasks.

In their paper, Devlin et al. first pre-train the BERT model weights on two unsupervised language learning tasks: masked language modelling (MLM), in which the model is trained to predict masked words based on surrounding words, and next sentence prediction (NSP), in which the model is given two sentences, and trained to predict if sentence B follows sentence A.

The initialised weights for BERT were pretrained with MLM and NSP on the BooksCorpus from Zhu et al. [15], and the English Wikipedia corpus. With these pretrained weights, the authors of the BERT paper fine-tuned the model on 11 different NLP benchmarking tasks, achieving state-of-the art results in each with significantly less computational training required than other models.

Liu et al. [16] replicated the results obtained by the original BERT paper but found that the initial pretrained weights that were published alongside the BERT paper were significantly undertrained. By optimizing the hyperparameters used in pretraining the BERT architecture, and training for more epochs, they were able to improve on the results published by Devlin et al, and released their weights as "RoBERTa", the Robustly Optimized BERT pretraining approach. Since its release, RoBERTa has become one of the premier pretrained models in most NLP task benchmarks. The RoBERTa team also experimented with a variety of different token embedding methods, and decided that byte-pair encoding (BPE) offered the best balance of performance and interpretability. BPE is able to trained to break words into their most frequently occuring substrings, and despite some arguments that it is a suboptimal embedding method [33], it is still one of the most popular tokenizers due to its interpretability and ease of use.

In 2020, Barbieri et al.[17] created the TweetEval benchmark, which is a framework that consists of a number of different classification tasks using various labelled SemEval datasets, which can be used to evaluate the performance of a models on social media tasks.

As part of the paper, they released a pre-trained RoBERTa model, named "RoBERTa-Tw", which is a set of model weights initialised as the original RoBERTa weights, but retrained on a large twitter corpus using the same unsupervised training tasks as BERT/RoBERTa (MLM/NSP). Additionally, they released sets of weights they fine-tuned on some of their classification tasks. For the purposes of this project, the most pertinent of these model weights are the "emotion classification" and "sentiment detection" models. These model weights were accessed via HuggingFace[34].

3.4 Attention Entropy

In their 2019 paper, as discussed in the introduction, Attanasio et al. found promising results in detecting toxic speech by using BERT models with an added loss component, which they named Entropy-based Attention Regularisation (EAR). Information entropy, introduced by Shannon in 1948, is a measure of the average amount of information content that a random variable has. For a given random variable, X, with possible outcomes $[x_0, ..., x_n]$, Shannon defined the entropy contained as :

$$H(X) = -\sum_i P(x_i) \log P(x_i) \tag{4}$$

In their paper, Attanasio et al. use this definition of information entropy by treating each token's attention distribution (averaged across all attention heads) as a probabilistic mass function. They first average the weights over all heads to give the average amount of attention position i gives to position j to be:

$$a'_{i,j} = \frac{1}{h} \sum_{h} a_{h,i,j} \tag{5}$$

From this, they use a softmax operation on the attention $a'_{i,j}$ to produce a probability mass function, $a_{i,j}$:

$$a_{i,j} = \frac{e^{a'_i,j}}{\sum_j e^{a'_i,j}} \tag{6}$$

With the probability mass function $a_{i,j}$, it follows from eq. 4 that the definition of entropy for the token i to be given as:

$$H_i = -\sum_{j=0}^{d_s} a_{i,j} \log a_{i,j} \tag{7}$$

Attanasio et al. provide an intuition as to how this attention entropy might help a model leverage the full context of a given sequence, and why it is particularly useful in hate speech modelling. Due to the nature of the problem, hate speech models are prone to focus too heavily on identity terms that might appear in a sequence. Without a method of penalising a model from over-focusing on such terms, models can quickly learn to *only* pay attention to identity terms and discard nearly all context surrounding them. Naturally, this leads to a poorly optimized local minima, where a model will immediately label any text containing an identity term as "hateful". (e.g. "I am a lesbian and I love my girlfriend", which is not at all hateful, but contains the identity word "lesbian").

By penalising a high attention entropy, the model is encouraged to learn representations that consider the context of a term existing in a sequence. This helps models to avoid misclassifying based on the presence or absence of a single identity term.

To achieve this, a component is added to the loss function of the model during training, that penalises high average attention entropy across the sequence (called "average contextualisation" by Attanasio et al.).

The average contextualisation in layer l is given by the averaged sum of attention across the layer:

$$H^l = \frac{1}{d_s} \sum_{i=0}^{d_s} H_i^l \tag{8}$$

And so we can write a new loss function that penalises a high attention entropy:

$$\mathcal{L} = \mathcal{L}_C + \mathcal{L}_R, \text{ where } \mathcal{L}_R = -\phi \sum_{l} H^l \tag{9}$$

Where $\mathcal{L}$ is the total loss, $\mathcal{L}_C$ is the classification loss, and $\mathcal{L}_R$ is the regularisation loss. ϕ is a hyperparameter that must be optimized.

3.5 Training and Evaluation methodology

To evaluate the hypothesis (that initialisng with a model trained on a subtask yields better model performance), a model was constructed with an archtecture as seen in fig 4.

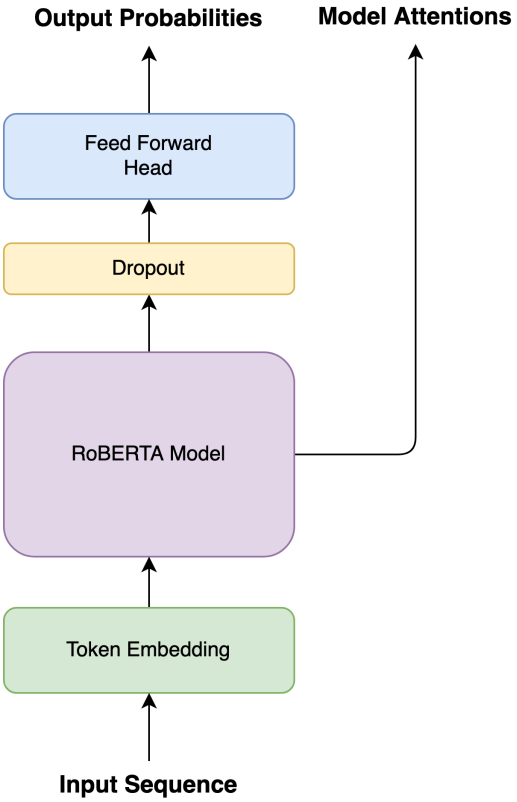

Figure 4: A figure showing a simplified high level overview of the architecture of the model. The RoBERTa model was initialised with weights that could be changed, and has a standard BERT architecture of 12 transformer encoder stacks.

The model was written in such a way that the EAR could be disabled, and the model could be swapped out (including the different token embeddings) as easily as possible.

The final output layer generates probabilities for 4 classes, representing the "Normal, Offensive, Hateful, and Split" labels contained in the dataset. The classification loss used was categorical cross-entropy, and the total loss during training was the sum of the attention entropy component of loss and the categorical accuracy component of loss.

The model was initially trained with an Adam optimizer, but it was quickly discovered that SGD drastically outperformed Adam. The possible reasons for this are discussed in the discussion section.

To compare different model configurations, the same training procedure was conducted each time, as follows.

The dataset is shuffled with a fixed random seed, and split into training, test, and validation splits in a proportion of 7:2:1. The weights in the RoBERTa layers are frozen, and the fully connected head is trained with a learning rate l_{head}, and stops training once classification accuracy in the test dataset does not improve after an epoch. Then, the RoBERTa weights are unfrozen, and the full model is fine-tuned with a learning rate $l_{finetune}$, once again until performance in the test dataset stagnates[1] The model is then evaluated against the validation data split, and the performance against this hold out dataset is the quoted performance.

3.6 Model selection and evaluation

Based on the model definitions above, there were 5 main hyperparameters to tune during model selection:

- l_{head}
- $l_{finetune}$
- Batch size
- Dropout rate
- ϕ (Proportion of entropy of attention in loss function)

[1] A patience of two was used, meaning a single epoch with no improvement would be allowed to continue, but if two epochs were completed with no improvement, the model would be reset to the weights that gave the best test performance.

First, to get an idea of the ranges that the optimal hyperparameters might lie in, the base RoBERTa model was loaded and each hyperparameter was given a broad logarithmically uniform probability distribution and a Bayesian hyperparameter sweep was conducted with 30 iterations. Using this, it was found that optimal hyperparameters likely lay in the following ranges:

$l_{head} \in [10^{-3}\text{–}10^{-2}]$, $l_{finetune} \in [10^{-5}\text{–}10^{-3}]$, Batch size = 32, Dropout rate $\in$ [0.1–0.4], $\phi \in [10^{-3}\text{–}10^{-1}]$

3.7 Evaluating effectiveness of EAR

To evaluate whether including EAR gave positive results to the model learning process, a grid search was conducted with the following hyperparameter sets:

- $l_{head} \in [10^{-3}, 3 \times 10^{-3}, 10^{-2}]$
- $l_{finetune} \in [10^{-5}, 10^{-4}, 10^{-3}]$
- Batch size = 32
- Dropout rate = 0.15
- $\phi \in [None, 10^{-2}, 10^{-1}]$

It was decided to use a grid search instead of a Bayesian search due to the complexity of including ϕ which gives the model an extra dimension to optimize over. This made it difficult to compare the impact of including ϕ because if each variant had the same number of runs to optimize over, the variant without ϕ would have been able to search the hyperparameter space more densely, due to it having a lower dimensionality.

3.8 Evaluating effectiveness of different RoBERTa initialized weights

To compare the effectiveness of transferring RoBERTa models that have been pretrained on different tasks, each RoBERTa variant completed 10 runs with Bayesian hyperparamter optimization with each hyperparameter having a log uniform distribution between the following values:

- $l_{head} \in [10^{-3} - 10^{-1}]$
- $l_{finetune} \in [10^{-4} - 10^{-3}]$
- Batch size = 32
- Dropout rate $\in [0.1 - 0.4]$
- $\phi \in [10^{-3} - 3 \times 10^{-2}]$

and the best performing model was selected. The performance metrics given in the results section are on the validation dataset.

As show in the results section, it was found that the results of Attanasio et al. held in this dataset, and that including EAR as a component in the loss function improved the models ability to train efficiently, and avoid over-focusing on identity terms, and thus could generalise better.

4 different trained RoBERTa models were chosen to use sources of initial weights for the RoBERTa model component of the main model architecture (see fig 4.) These models were:

- RoBERTa-base, the original RoBERTa model weights [16]
- Twitter-base: RoBERTa-base further trained on unsupervised tasks on Twitter dataset (March 2022 version) [17] [2].
- Sentiment: The Twitter-base model fine tuned to classify inputs based on sentiment labels, using the SemEval2017 Subtask-A dataset [17] [2].
- Emotion: The Twitter-base model fine tuned to classify inputs based on the SemEval2018 "Affects in Tweets" dataset, but only considering "Anger, Joy, Sadness, Optimism", as described in the Barbieri et al. paper [2] [17].

These models were each trained 10 times with different hyperparameters as found by a Bayesian hyperparameter sweep with the above ranges, and the best performing models on the validation dataset were used to compare overall model performances. The results of the hyperparameter sweeps are also provided in the results section as box plots.

[2]The weights for these models were accessed via the Cardiff-NLP open access repository available at https://huggingface.co/cardiffnlp

4 Results

NB: In the hyperparameter optimization plots, the categorical accuracies are taken from training, and so include the 4th column, "Split", as described in section 3. This column was excluded for the final model performances quoted in table 2. These entries were not excluded during training, so that the model had more entries to learn from, but they were removed when calculating the final accuracies so the results could be more comparable to the results quoted in the original HateXplain paper by Matthew et al. [28], but a discrepancy is therefore present when looking at the "categorical accuracy" seen during training and during evaluation.

4.1 EAR Investigation

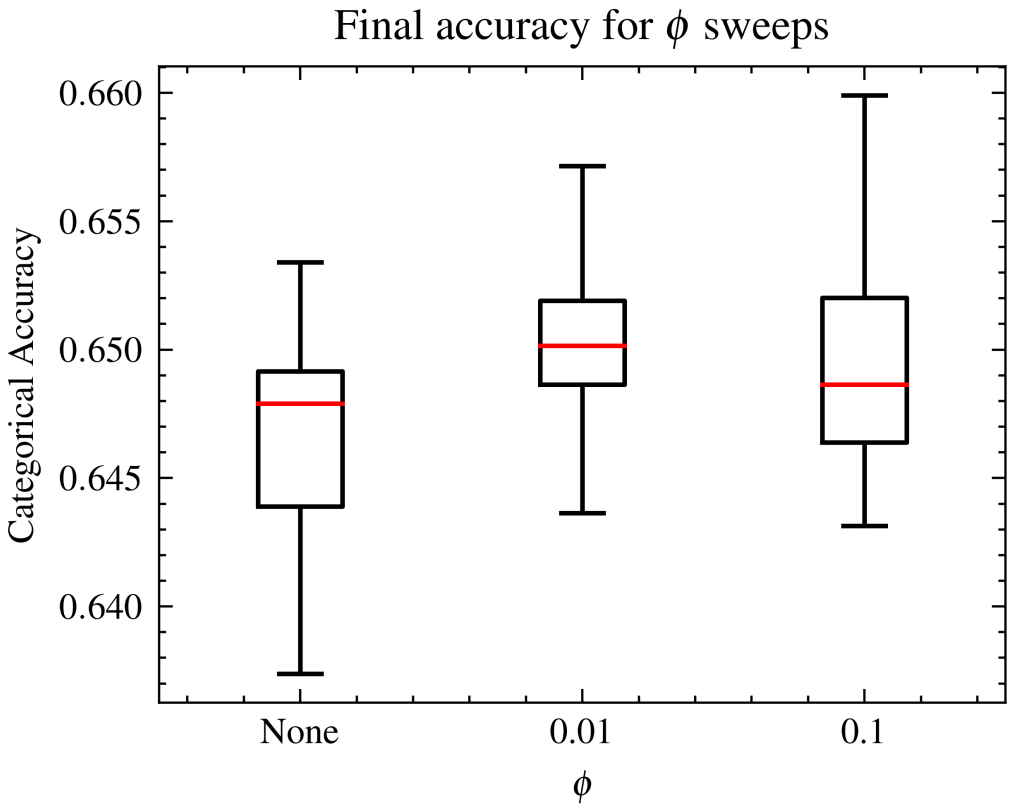

Figure 5: A box plot showing how including phi during training on a base RoBERTa model affected categorical accuracy of predictions on a hold-out dataset.

Figure 5 shows a box plot comparing the model performances attained during a grid hyperparameter search when $\phi = 0.01, 0.1$ and for when ϕ was 0, and no entropy regularisation was present during fine-tuning of the RoBERTa model.

As seen, the accuracies attained for $\phi = 0.01$ and $\phi = 0.1$ were greater than the accuracies attained without ϕ, with the top performing of each value of ϕ being greater than the top performing result from no ϕ, as well as an improved worst performing model in each case of ϕ.

4.2 Knowledge Sharing

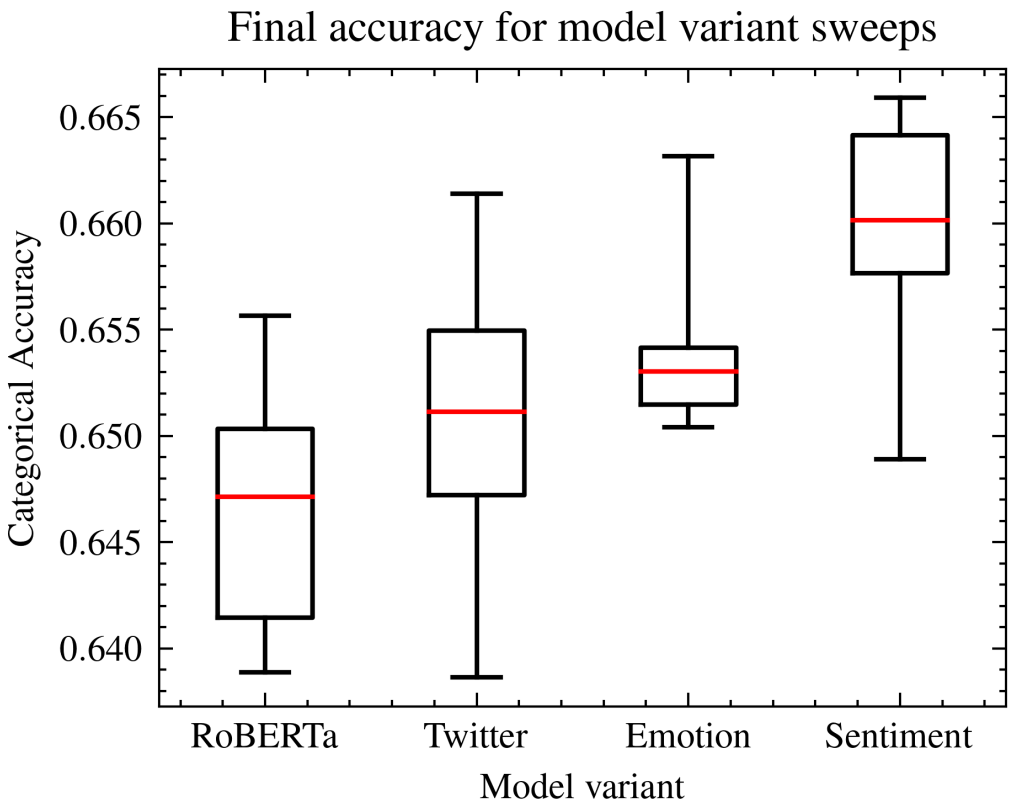

Figure 6: A box plot showing the distribution of final accuracies obtained during training for the 10 runs with each model variant.

Model Variant	**Performance**		
	Acc	Macro F1	AUROC
BERT *	0.690	0.674	0.843
RoBERTa base	0.689	0.686	0.835
BerTweet	0.695	0.695	0.845
Emotion	0.696	0.693	0.843
Sentiment	**0.700**	**0.697**	**0.848**

Table 2: A table showing the model performance of the base BERT model from the HateXplain paper[4](marked *) compared with the best performing models of each RoBERTa model variant.

As seen in table 2 and figure 6, overall, better performance was attained across the hyperparameter sweeps for the emotion and sentiment model variants, when compared to the RoBERTa-Tw weights (called Twitter for clarity in the plot), which was also better than the original RoBERTa model. The RoBERTa base model performed similarly to the base BERT model found in the HateXplain paper, and the best performing sentiment model was able to attain an accuracy of 0.700. Further discussion around these results can be found in section 5.2.

[4]The models from the HateXplain paper with their HateXplain training method were not included, as they included a novel method of training based on attention entropy between the model and annotators, and are not directly comparable to the models in this paper. However, they performed similarly to the Sentiment model.

RoBERTa-Base		**Predicted**			
		Nrml.	**Offn.**	**Hate**	**Split**
True	**Normal**	1136	310	136	0
	Offensive	309	553	199	0
	Hate	102	128	931	0
	Split	70	78	44	0

Twitter-base		**Predicted**			
		Nrml.	**Offn.**	**Hate**	**Split**
True	**Normal**	1162	329	91	0
	Offensive	304	592	165	0
	Hate	120	152	889	0
	Split	76	82	34	0

Sentiment		**Predicted**			
		Nrml.	**Offn.**	**Hate**	**Split**
True	**Normal**	1195	289	98	0
	Offensive	315	553	193	0
	Hate	116	132	913	0
	Split	72	80	40	0

Emotion		**Predicted**			
		Nrml.	**Offn.**	**Hate**	**Split**
True	**Normal**	1213	263	106	0
	Offensive	333	540	188	0
	Hate	112	152	897	0
	Split	76	82	34	0

Table 3: A set of confusion matrices showing the predicted and true labels for the best performing model variants of the hyperparameter sweeps

4.3 Error analysis table [CW]

Original string	**True label**	RoBERTa	Twitter	Emotion	Sentiment
<user>americans hate you so much they would rather vote for an illegal alien who wanted to destroy the usa as much as you did he just did not seem as dangerously stupid	**Normal**	Offensive	Offensive	Normal	Normal
<user>bat ching chong	**Normal**	Hate	Hate	Hate	Offensive
i do not need time to forget you but i just need time to hate you	**Hate**	Normal	Normal	Normal	Normal
haha you keep saying bye yet you yearn for more seems pretty shitskin tier sad tea biscuit	**Offensive**	Hate	Normal	Normal	Offensive
when you are fighting a war as retarded as afghanistan all you have to keep you going are your brothers and he betrayed them i do not even want this shithead tortured i just want him dead you do not waste torture on a cockroach you step on it	**Offensive**	Offensive	Offensive	Hate	Hate

Table 4: A table showing select examples instances that the models struggled to predict correctly. NB: The <user> token is a special token introduced by Matthew et al. in the HateXplain dataset to stand in for one user responding to a different user.

5 Discussion

5.1 Entropy based Attention Regularisation

Based on the results in 5, it is clear that including a component of EAR in the loss function of a model is able to improve the performance of the final model when making predictions on hate speech texts. Additionally, for the best performing sets of hyperparameters for each of the RoBERTa model variants, phi was significant, so it can be seen that even when fine tuning models that have been pretrained on sub-tasks, having a component of entropy based attention regularisation can help the training process.

5.2 Model Variants

As seen in the results, the best performing RoBERTa model variant was the set of weights pretrained on classifying sentiment in social media posts. The model pretrained on classifying emotion performed similarly to the base model pretrained by Cardiff-NLP on the social media text corpus, which both performed better than the base RoBERTa model weights. This can be seen both in the macro F1, categorical accuracy, and AUROC scores of the best models as shown in table 2 and in the overall model results from the hyperparameter sweep, as plotted in figure 6.

These results support the initial hypothesis, that sentiment detection is an important component of identifying hate speech, and that transferring knowledge from sentiment detection training can prove beneficial in training hate speech models. That said, it is important to note that while it appears that there is value in using knowledge from sentiment classification tasks on hate speech detection tasks, an over-reliance on this knowledge transfer is likely to cause problems with overfitting. Röttger et al. [22] highlight in their paper introducing a functional test framework for hate speech models, that many models can already be somewhat over reliant on picking up sentiment expressions, and miss cases where a text can be "hateful" but not negative.

Looking at Table 3, we can see confusion matrices for the best performing models of each variant. Looking at the results, we can see that the sentiment most models struggled most differentiating between offensive and normal instances, and none made any predictions of "split" class at all. This makes intuitive sense, as these the line between these classes is quite blurred at times (see examples in Table 4). The sentiment and emotion models were both better at identifying "normal" texts

Due to time and computational limitations, comparing the performance of models against the HateCheck benchmarks was out of scope of this paper. Further research should consider evaluating against a functional test suite such as HateCheck to better understand where the model is performing well, and where it is struggling.

While initialising weights with a model trained on sentiment detection is likely to make the focus on sentiment noted by Röttger et al. more pronounced, this is not a certainty. As this experiment has found that initialising with sentiment classification weights has improved model performance on the whole, further work may be needed to investigate the potential drawbacks of initialising from a sentiment classification model further.

5.3 Error analysis [CW]

In table 4, we can see a selection of some of the instances that the models are struggling to classify correctly. Both the RoBERTa and Twitter variant models fail to classify the first text correctly, but the Twitter and Emotion models get it correct. The text does contain the words "hate", and disparaging remarks towards "illegal aliens" (which belong to the "refugee" protected group in the HateXplain paper), but the models were able to understand contextually that this text was not explicitly "hate speech". That said, this text is also a good example of the fuzzy lines that exist in the categorisation. A trained human annotator could easily label this as either hateful or offensive, as the text could easily be read to express a view that "illegal aliens" are more likely to "destroy the USA", but the emotion and sentiment models are able to apply the same justifications as the annotators, likely identifying the hypothetical nature of the text.

In the next row, however, we see a good example of the models identifying a case which is quite clearly hateful, but is not labelled by the annotators as such. The text as it uses a slur towards Asian groups and perpetrates a harmful stereotype ("bat" here likely relating to the reported zoonotic origin of the COVID-19 epidemic in East Asia). Many annotators would likely describe this as "hateful" as it uses hateful language, and perpetrates a harmful stereotype which has lead to an up-tick in discrimination and violence against East Asian minorities around the globe [35]. This emphasises the difficulty in generating a consistently labelled hate speech dataset.

The final two entries in the table also serve to show the fuzzy edges of hate speech labelling. Does use of an uncommon slur in a different, non-hateful context, count as hateful? Does calling a hostile combatant in a war a "cockroach" count as hateful? Different annotators may have different opinions on these cases, and this kind of noise and uncertainty in the nature of the labels themselves makes it very difficult to train a model with a high degree of accuracy.

5.4 Choice of Optimizer

As discussed in section 3.5, it was found that SGD outperformed Adam in this project.

The debate around the pros and cons of different optimization methods is far from settled, and many arguments are made in favour of Adam or SGD for different problems. Wilson et al.[36] provide arguments that show that models with adaptive gradient methods, such as Adam, generalise worse than non-adaptive methods on several real datasets. Zhou et al. [37] use mathematical arguments to show SGD is better at escaping local minima in basins, and so SGD generalises better when a loss functions surface contains many sharp local minima.

The nature of hate speech detection is that it provides a noisy, chaotic signal, as the problem itself is hard to define in rigid terms, and more so than with other NLP problems, multiple trained humans are prone to label the same instance differently. This chaotic and noisy signal may lead to many steep local minima, but further research would need to be done to validate this idea.

While SGD has performed better in this datasets, there are trade-offs to not using an adaptive gradient descent algorithm, such as slower convergence. Various authors have published hybrid mechanisms that can transition from Adam early in training to SGD, such as the DSTAdam mechanism proposed by Zeng et al.[38]. Further work could investigate if this hybrid approach can perform well in this kind of project.

6 Limitations of this work

As mentioned in the discussions section, one of the major limitations of this work is that initialising the model from weights pretrained on a subtask may cause unintended biases in the results. The model initialised with weights from sentiment classification tasks did perform better on the dataset than any of the other model variants, but this may introduce an additional focus on the perceived sentiment of the input sequence, and cause the model to struggle further with complex cases where an input is "hateful" but has positive sentiment.

However, one of the major limitations currently in the field of hate speech detection research is availability of large, varied labelled datasets. The ability to transfer knowledge gained on a task with a much more expansive range of data, such as sentiment analysis, which in this project has been shown to improve model performance, is still a valuable tool to improve the effective size of the dataset the model is trained on.

7 Conclusion

In this project, the benefits of entropy-based attention regularisation were able to be replicated on a different dataset, and it was shown that by transferring knowledge gained from a relevant sub-task, such as sentiment detection, that hate speech models can be fine-tuned to perform better than when they are initialised from a generic task checkpoint, like RoBERTa-base.

Hate speech online is a growing problem that causes real harm to many people, and if left unchecked, can have disastrous consequences on societal scales. Improving the ability to understand, and tackle this problem is incredibly important, and it is the hope of the author that the findings described within may help address the issue in future.

8 Ethical considerations

The main ethical complication in publishing models that are trained to predict hate speech is the risk that malicious users could weaponise the findings to learn new methods of avoiding their hate speech from being detected. The main findings of this paper are about optimizations to methods for training models, and so this risk is very small. The potential benefit of these findings help to improve hate speech detection and mitigation online vastly outweighs this potential for misuse.

References

[1] A. Guterres, "United Nations Strategy and Plan of Action on Hate Speech," May 2019.

[2] K. Gelber and L. McNamara, "Evidencing the harms of hate speech," *Social Identities*, vol. 22, pp. 324–341, May 2016.

[3] K. Rapp, "Social media and genocide: The case for home state responsibility," *Journal of Human Rights*, vol. 20, pp. 486–502, Aug. 2021.

[4] M. Popa-Wyatt, "Reclamation: Taking back control of words.," *Grazer Philosophische Studien*, Jan. 2020.

[5] K. Bogetić, "Race and the language of incels: Figurative neologisms in an emerging English cryptolect," *English Today*, pp. 1–11, June 2022.

[6] A. Gaydhani, V. Doma, S. Kendre, and L. Bhagwat, "Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach," Sept. 2018.

[7] M. Bojkovský and M. Pikuliak, "STUFIIT at SemEval-2019 Task 5: Multilingual Hate Speech Detection on Twitter with MUSE and ELMo Embeddings," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, (Minneapolis, Minnesota, USA), pp. 464–468, Association for Computational Linguistics, June 2019.

[8] S. Silva, T. Ferreira, R. Ramos, and I. Paraboni, "Data Driven and Psycholinguistics Motivated Approaches to Hate Speech Detection," *Computación y Sistemas*, vol. 24, Sept. 2020.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Dec. 2017.

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," July 2020.

[11] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," Jan. 2020.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019.

[13] 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, Ł. 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," Oct. 2016.

[14] P. Gage, "A new algorithm for data compression," *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[15] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 19–27, 2015.

[16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," July 2019.

[17] F. Barbieri, J. Camacho-Collados, L. Espinosa Anke, and L. Neves, "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, (Online), pp. 1644–1650, Association for Computational Linguistics, Nov. 2020.

[18] X. Zhou, Y. Yong, X. Fan, G. Ren, Y. Song, Y. Diao, L. Yang, and H. Lin, "Hate Speech Detection Based on Sentiment Knowledge Sharing," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Online), pp. 7158–7166, Association for Computational Linguistics, Aug. 2021.

[19] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating Unwanted Biases with Adversarial Learning," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, (New York, NY, USA), pp. 335–340, Association for Computing Machinery, Dec. 2018.

[20] X. Zhou, M. Sap, S. Swayamdipta, N. A. Smith, and Y. Choi, "Challenges in Automated Debiasing for Toxic Language Detection," Jan. 2021.

[21] B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela, "Learning from the Worst: Dynamically Generated Datasets to Improve Online Hate Detection," June 2021.

[22] P. Röttger, B. Vidgen, D. Nguyen, Z. Waseem, H. Margetts, and J. Pierrehumbert, "HateCheck: Functional Tests for Hate Speech Detection Models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Online), pp. 41–58, Association for Computational Linguistics, Aug. 2021.

[23] G. Attanasio, D. Nozza, D. Hovy, and E. Baralis, "Entropy-based Attention Regularization Frees Unintended Bias Mitigation from Lists," Mar. 2022.

[24] M. Anzovino, E. Fersini, and P. Rosso, "Automatic Identification and Classification of Misogynistic Language on Twitter," in *Natural Language Processing and Information Systems* (M. Silberztein, F. Atigui, E. Kornyshova, E. Métais, and F. Meziane, eds.), Lecture Notes in Computer Science, (Cham), pp. 57–64, Springer International Publishing, 2018.

[25] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 11, pp. 512–515, 2017.

[26] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. Rangel Pardo, P. Rosso, and M. Sanguinetti, "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*, (Minneapolis, Minnesota, USA), pp. 54–63, Association for Computational Linguistics, June 2019.

[27] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis, "Large scale crowdsourcing and characterization of twitter abusive behavior," in *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[28] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee, "HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection," Apr. 2022.

[29] R. Bhalerao, M. Al-Rubaie, A. Bhaskar, and I. Markov, "Data-Driven Mitigation of Adversarial Text Perturbation," Feb. 2022.

[30] U. Naseem, I. Razzak, and P. W. Eklund, "A survey of pre-processing techniques to improve short-text quality: A case study on hate speech detection on twitter," *Multimedia Tools and Applications*, vol. 80, pp. 35239–35266, Nov. 2021.

[31] Y. Qiao, C. Xiong, Z. Liu, and Z. Liu, "Understanding the Behaviors of BERT in Ranking," Apr. 2019.

[32] M. Tuters and S. Hagen, "(((They))) rule: Memetic antagonism and nebulous othering on 4chan," *New media & society*, vol. 22, no. 12, pp. 2218–2237, 2020.

[33] K. Bostrom and G. Durrett, "Byte Pair Encoding is Suboptimal for Language Model Pretraining," pp. 4617–4624, Jan. 2020.

[34] "Hugging Face – The AI community building the future.." https://huggingface.co/.

[35] C. Wu, Y. Qian, and R. Wilkes, "Anti-Asian discrimination and the Asian-white mental health gap during COVID-19," *Ethnic and Racial Studies*, vol. 44, pp. 819–835, Apr. 2021.

[36] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The Marginal Value of Adaptive Gradient Methods in Machine Learning," May 2018.

[37] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and W. E, "Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning," Nov. 2021.

[38] K. Zeng, J. Liu, Z. Jiang, and D. Xu, "A Decreasing Scaling Transition Scheme from Adam to SGD," *Advanced Theory and Simulations*, vol. 5, no. 7, p. 2100599, 2022.