# High Accuracy with Low Costs: The Pretrain-Finetune Paradigm for Classification with Transformer-based Language Models

Yuan (Cecilia) Sui*

**Short Title:** The Pretrain-Finetune Paradigm for Political Science

**Keywords:** text analysis, text classification, language models, transfer learning

*PhD Candidate in Political Science, Washington University in St. Louis. Mailing Address: One Brookings Drive, St. Louis, MO 63130, United States. Email: c.sui@wustl.edu

**Abstract**

Political science research increasingly relies on text classification to gauge subtle concepts like toxicity or anger. However, common methods treat words and phrases in isolation, ignoring the relational dynamics where meaning is embedded in the interaction between phrases and their textual environment. Recent transformer-based language models overcome such limitations, but are rarely used in published political science articles, possibly due to misconceptions about their data and computational requirements. To bridge this gap, this article explains the pretrain-finetune paradigm for transformer-based language models and its potential to improve text analysis in political science by harnessing robust models trained on extensive datasets while requiring relatively small amounts of labeled data and computational resources from researchers. To demonstrate its effectiveness, I employ the approach to identify toxic language in conversation threads following U.S. Senators' tweets.

**Word Count: 9,853 (exclude title, abstract, and reference section)**

Text classification is now a standard tool for measurement tasks in political science across substantive domains (Bestvater and Monroe, 2022; Bosley et al., 2022; Goet, 2019; Grimmer and Stewart, 2013; Lucas et al., 2015; Roberts, 2016). However, a common problem with current models is that they struggle to effectively capture context-dependent concepts such as toxicity, sentiment, and emotion (Parekh, 2012; Sellars, 2016). Such limitations arise from their inability to selectively focus on the most relevant words and the inattention to word order. This is particularly true for bag-of-words and dictionary-based methods (Jacobs et al., 2021; Osmundsen et al., 2021; Osnabrügge, Hobolt and Rodon, 2021; Siegel and Badaan, 2020; Wasow, 2020). Even somewhat more sophisticated models, such as text embeddings,[1] face challenges when dealing with multiple meanings of words in different settings (Alrababa'h et al., 2021; Bestvater and Monroe, 2022). This is because they rely on static representations and may not capture the nuances of how a word's meaning shifts in different contexts. For instance, consider the word "*left*" in the two sentences below: the context gives "*left*" different meanings, something missed by all models using static representations.

> Sentence 1: The *left* advocates for more government intervention in the economy to ensure equity and social welfare.
>
> Sentence 2: After the debate, he *left* the stage.

Recent advancements in transformer-based language models offer a promising solution to this issue. Transformers excel at capturing long-range dependencies and relationships given their ability to weigh the importance of different words in a sequence based on their context, i.e., the attention mechanism (Vaswani et al., 2017). Such architecture provides both enhanced language representation and model interpretability. Despite the potential of transformer-based language models, the adoption of them in political science remains

[1] A text embedding is a numerical vector that represents text, with tokens of similar meanings having similar representations, typically in a low-dimensional vector space (Collobert and Weston, 2008).

limited. This is, in part, because the overall methodology may still appear inaccessible for applied scholars due to a lack of clear explanation and misconceptions about the amount of labeled data and computational resources required.

To address this issue, in this article, I provide a conceptual overview of the pretrain-finetune paradigm (PFP), focusing on offering an approachable explication of the methods, their potential applications, and inherent limitations. The PFP comprises two stages: pre-training, where a language model learns general linguistic patterns from diverse, unlabeled text corpora, and fine-tuning, where the model is further trained on task-specific labeled data. The fine-tuning stage allows for efficient adaptation of pre-trained models to targeted tasks, leveraging the previously acquired linguistic knowledge. The increasing availability of open-source pre-trained models enables scholars to achieve data efficiency, training efficiency, and performance enhancement via the process of fine-tuning without developing classifiers from scratch.

In the next section, I explain the measurement task of text classification and provide a brief overview of current methods used in the field of political science. I show that while there are a few papers that use the PFP approach (Bestvater and Monroe, 2022; Wang, 2023*a*,*b*), the vast majority of published works do not. To motivate the PFP, I then outline the limitations of the most widely used methods for political science domain-specific text analysis. The following section is dedicated to a comprehensive overview of the PFP emphasizing the attention mechanism, while the subsequent section walks through the process of using the PFP in details with an application to identify toxic language in Twitter conversation threads. The article concludes with a discussion on the limitations inherent in the PFP and suggests potential avenues for future research in this domain.

## Current Text Classification Approaches

To contextualize the discussion, imagine we are interested in measuring toxicity from some text corpus that contains the two example sentences below.[2]

Example 1 (*E1*): Stop the evil *yellow* invasion from the Chinese.

Example 2 (*E2*): The flowers are *yellow*.

The simplest approach for identifying toxic language is to use a standard dictionary method. The procedure entails examining the input text for the presence of specific derogatory terms that have been pre-defined by scholars (Jacobs et al., 2021; Osmundsen et al., 2021; Osnabrügge, Hobolt and Rodon, 2021; Siegel and Badaan, 2020; Wasow, 2020). Should these specific terms be detected within the input text, it is classified as toxic. For instance, Siegel and Badaan (2020) used a dictionary containing anti-Shia slurs to measure sectarian hate speech on Twitter. I use a similar application to detect toxic language on Twitter to demonstrate the effectiveness of the PFP in the application section.

However, there are several limitations of using dictionaries for classifying toxic language. First, dictionaries rely solely on explicitly pre-defined terms, which may overlook more sophisticated forms of toxicity, such as coded language, or manipulative speech. Second, due to the over-reliance on explicit terms, dictionaries are inadequate in addressing the complexity of context-dependent terms, which might be deemed toxic or non-toxic based on their usage within a given context, such as the word *"yellow"* in *E1* and *E2*. Consequently, the inclusion or exclusion of *"yellow"* in the dictionary leads to misclassification of one of the two sentences.

[2]In the application section, I measured toxicity in conversation threads from tweets by U.S. Senators, comparing PFP models with traditional approaches discussed in this section, and demonstrated the superior performance of PFP models.

Beyond dictionaries, a popular approach in the field is to use bag-of-words (BOW), which generates numerical vectors (i.e., text embeddings) by counting the occurrences of words or tokens from the input text, regardless of the token order (Barberá et al., 2019; Beltran et al., 2021; Cocco and Monechi, 2022; Grimmer and Stewart, 2013; Vries, Schoonvelde and Schumacher, 2018). For example, Cocco and Monechi (2022) used BOW along with a random forest classifier to measure populist content in party manifestos. Beltran et al. (2021) transformed Spanish tweets using BOW with logistic regression to examine the differences in how male and female politicians communicate with the public on social media.

Similar to dictionaries, BOW is unable to capture multiple meanings of words due to its lack of context sensitivity. The word *"yellow"* from *E1* and *E2* would again be treated as the same using a BOW model. Moreover, word order is not factored in when generating numerical vectors, which makes BOW struggle to capture negative meanings. *E3* would be numerically equivalent to *E4* when using uni-grams in BOW. Such inability becomes particularly problematic in applications like toxicity detection.

Example 3 (*E3*): I do like Kamala and *not* Trump.

Example 4 (*E4*): I do *not* like Kamala and Trump.

A more advanced method of text representation is to use static word embeddings (Alrababa'h et al., 2021; D'Sa, Illina and Fohr, 2020; Esberg and Siegel, 2023). To generate numerical representations in a continuous vector space and take advantage of the surrounding words, word2vec is introduced with the intuition that a word is defined by its neighboring words (Mikolov et al., 2013). For example, Esberg and Siegel (2023) converted tweets into numerical vectors using word2vec to identify words that are semantically similar to their seed words in the data. Using word2vec with a naive Bayes classifier, Alrababa'h et al. (2021) identified anti-Muslim content in 15 millions tweets from British soccer fans.

The key limitation of word2vec and related models is that they generate static numerical representations. In other words, the numerical vector representing a word is fixed during

training. Therefore, it is unable to handle the complexity of context-dependent words like *"yellow"* in *E1* and *E2* when their context is different from training. The context provided by neighboring words is also limited by a window of words or tokens on the left or right, making it inadequate to connect two words that are further away from each other.[3]

In response to the constraints observed in prior models for text representation, transformer-based language models are introduced to effectively manage the intricacies inherent in linguistic nuances (Vaswani et al., 2017). First, multiple meanings of words are handled with dynamic embeddings, which ensure the numerical representation of a word is contextually enriched by the presence of all other words within the same sequence. For instance, the word *"yellow"* in *E2* may adopt meanings influenced by words such as *"flowers"*. Similarly, the connotation of *"yellow"* can also be shaped by its association with words such as *"Chinese"*, *"evil"*, and *"invasion"* in *E1*.

Second, transformers incorporate word order into their numerical vectors through the inclusion of position embeddings. Position embeddings are numerical vectors that are added to the word embeddings to encode the position of each word in the input sequence. By combining word embeddings and position embeddings, transformers are able to capture both the semantic meaning of the words and their sequential order, which is crucial for language understanding tasks such as text classification as shown in *E3* and *E4*.

Third, compared to static word embedding approaches like word2vec, transformers allow us to consider a much broader context window that is theoretically unlimited.[4] The fixed-size content window of traditional word embedding approaches can limit the model's

[3]Researchers can increase the context window to capture all tokens in an input sequence. Recurrent neural networks addresses this, but still face the vanishing gradient problem.

[4]In practice, a transformer's context window is limited by computational and memory constraints, with the actual size depending on the model's token limit. For example, Bidirectional Encoder Representations from Transformers (BERT) has a 512-token limit, GPT4-Turbo allows 128K tokens, and Gemini-1.5-Pro supports up to 1 million tokens.

ability to capture broader semantic relationship between words. In contrast, transformers use an attention-based mechanism that allows the model to dynamically attend to and consider the entire input sequence when computing the representation of a given word.

Fourth, given the large context window of transformers, the attention mechanism facilitates a more refined and context-rich representation of words (or tokens), thereby enabling the model to produce more expressive token embeddings at the sentence level.[5] This capacity extends to addressing irregularities such as typographical errors in the input text, commonly seen in social media data (Shawky, ElKaffas and Guirguis, 2024), showcasing the model's adaptability and sophistication in handling linguistic variability.

Despite the exceptional efficacy of transformer-based language models in text analysis (Karl and Scherp, 2023; Park, Vyas and Shah, 2022), their adoption within the field of political science remains rare. From a dataset collected by Park and Montgomery (2023), where they searched the 2020-2022 volumes of the American Political Science Review (APSR), the American Journal of Political Science (AJPS), and the Journal of Politics (JoP) for all articles that used machine learning of texts to create measures of latent concepts, 10 papers used topic modeling (Berliner et al., 2021; Feierherd, 2022; Magaloni and Rodriguez, 2020; Manekin and Mitts, 2022; Motolinia, 2021; Nielsen, 2020; Roberts, Stewart and Nielsen, 2020; Rossiter, 2022; Saraceno, 2020; Yoder, 2020), 12 papers used dictionary-based methods (Bridgman et al., 2021; Crabtree et al., 2020; Djourelova and Durante, 2022; Jacobs et al., 2021; Jung, 2020; Lajevardi, 2021; Magaloni, Franco-Vivanco and Melo, 2020; Osmundsen et al., 2021; Osnabrügge, Hobolt and Rodon, 2021; Todd et al., 2021; Wasow, 2020), and 18 papers used some type of supervised learning for text analysis (Alrababa'h et al., 2021; Anastasopoulos and Bertelli, 2020; Casas, Denny and Wilkerson, 2020; Fowler et al., 2021; Gohdes, 2020; Guess, 2021; Hager and Hilbig, 2020; Park, Greene and Colaresi, 2020; Schub, 2022; Stier et al., 2022; Wahman, Frantzeskakis

[5]Models like BERT average all token embeddings for sentence embeddings, while others like GPT use contrastive learning for more semantically meaningful sentence-level embeddings.

and Yildirim, 2021; Zubek, Dasgupta and Doyle, 2021). Among the 18 papers that used supervised learning, only one paper leveraged a transformer-based language model to measure issue areas of parliamentary speeches (Wahman, Frantzeskakis and Yildirim, 2021). This scarcity may partly be attributed to a lack of explication of the models and misconceptions about the amount of labeled data and computational resources these methods require. In the following section, I offer a comprehensive elucidation of the PFP. For reproducibility, all computation in this article was conducted on Google Colab on an A100 GPU. All PFP models used in this article can be found on HuggingFace Hub.[6] Python code are provided in the replication archive.

# The Pretrain-Finetune Paradigm

In this section, I first offer a detailed overview of the transformer encoder[7] with an emphasis on the attention mechanism. Then, I discuss the pre-training stage and the fine-tuning stage of the PFP. Finally, I provide guidelines for scholars to use the PFP with different levels of computational resources or available data at hand.

## The Transformer Encoder

A transformer encoder, as shown in Figure 1, takes text embeddings as the input and outputs contextualized embeddings of the same length as the input sequence. Unlike

[6]https://huggingface.co/AnonymousCS

[7]There are three types of transformer models: encoder-only models (i.e. auto-encoding models), like BERT, process input data and generate a meaningful numerical representation of the input data; decoder-only models (i.e. auto-regressive models), like GPT-3 and GPT-4, specialize in language generation and are often used with prompt engineering, though they can be fine-tuned for classification; and encoder-decoder models, such as BART and T5, are designed for sequence-to-sequence tasks, where one text string is converted into another, like machine translation.

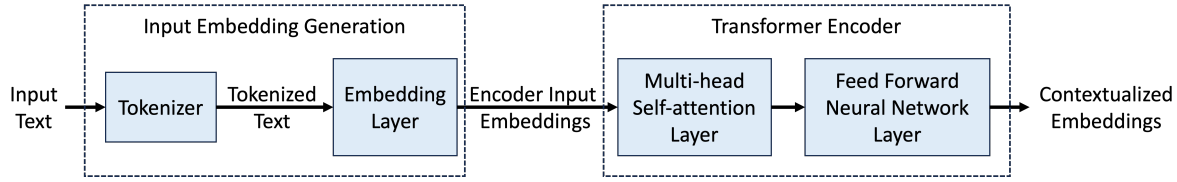



Figure 1: Transformer Encoder Architecture.

the input embeddings, each output embedding contains not only information about the individual token, but also its relationship with every other token in the same sequence. The following subsections examine each part in Figure 1 in details.

### Transformer Encoder Input Embedding Generation

Consider *E1* again. As shown in Figure 1 and Figure 2,[8] I first tokenize the input sequence into individual tokens. Then I map the tokens into input IDs according to their position in the vocabulary[9] containing all possible words in the training corpus. At this stage, both *"the"*s have the same input ID because they occupy the same position in the vocabulary. Then, I take the input IDs and map them into vectors of size 512[10] through an embedding layer which acts as a look-up table to get the initialized input embeddings (Vaswani et al., 2017). These corresponding embeddings can be loaded from some pre-trained models or initialized randomly,[11] and the embedding layer is often the first trainable layer in a

[8]Tokenization can be at word level or sub-word level. Different tokenizers have different design choices. Most transformer-based language models use sub-word level tokenization in practice. In this example, I use a word-level tokenizer for illustration purposes.

[9]In practice, the vocabulary size of language models range from 30K to 50K.

[10]512 is the dimensionality of the input and output vectors in each encoder layer. It determines the size of the input embeddings, the size of the feed forward neural networks, and the size of the output embeddings. Common values are 512 and 1024 or higher.

[11]Embeddings can be randomly initialized or from a trained corpus, a design choice for different models. The embedding layer's main function is to convert tokens into numerical vectors of desired dimensions,

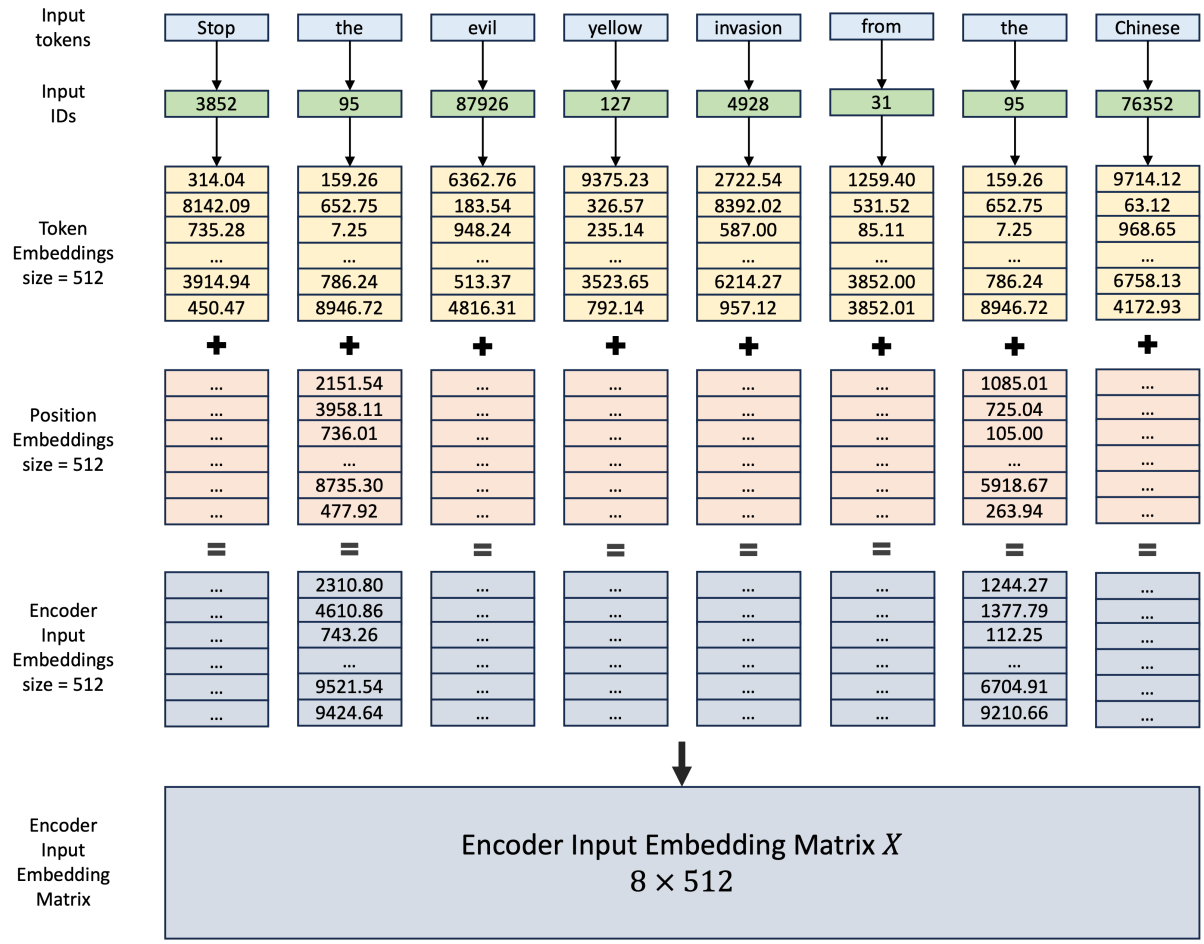


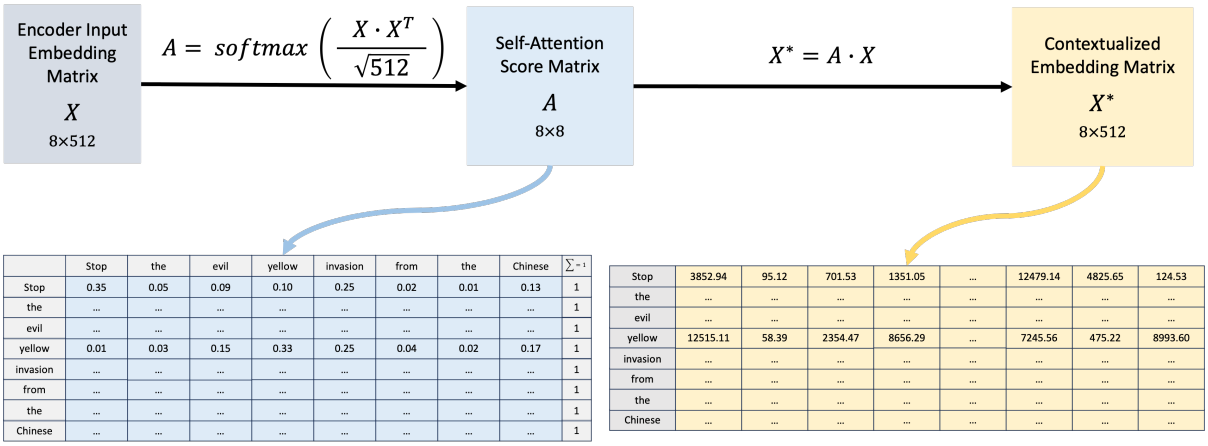

Figure 2: Input Embedding Generation to the Transformer Encoder. Numbers are for illustration purposes only.

transformer-based language model. Notice that at this stage, the same word is always mapped to the same embedding without any contextual information from surrounding words, similar to static word embedding models.

The next part of the embedding layer is the calculation of position embeddings, which are also of length 512 for each token and used by the model to keep track of their positions in the sequence and how distant they are from each other.[12] The token embeddings shown

clustering similar tokens together. Its weights are trainable parameters, updated along with other model parameters discussed in the next section.

[12]Position embeddings are usually only computed once and reused for every sentence during training and inference based on Vaswani et al. (2017), but they can also be learnable parameters as a design




| | Stop | the | evil | yellow | invasion | from | the | Chinese | $\sum=1$ |
|---|---|---|---|---|---|---|---|---|---|
| Stop | 0.35 | 0.05 | 0.09 | 0.10 | 0.25 | 0.02 | 0.01 | 0.13 | 1 |
| the | ... | ... | ... | ... | ... | ... | ... | ... | 1 |
| evil | ... | ... | ... | ... | ... | ... | ... | ... | 1 |
| yellow | 0.01 | 0.03 | 0.15 | 0.33 | 0.25 | 0.04 | 0.02 | 0.17 | 1 |
| invasion | ... | ... | ... | ... | ... | ... | ... | ... | 1 |
| from | ... | ... | ... | ... | ... | ... | ... | ... | 1 |
| the | ... | ... | ... | ... | ... | ... | ... | ... | 1 |
| Chinese | ... | ... | ... | ... | ... | ... | ... | ... | 1 |

| Stop | 3852.94 | 95.12 | 701.53 | 1351.05 | ... | 12479.14 | 4825.65 | 124.53 |
|---|---|---|---|---|---|---|---|---|
| the | ... | ... | ... | ... | ... | ... | ... | ... |
| evil | ... | ... | ... | ... | ... | ... | ... | ... |
| yellow | 12515.11 | 58.39 | 2354.47 | 8656.29 | ... | 7245.56 | 475.22 | 8993.60 |
| invasion | ... | ... | ... | ... | ... | ... | ... | ... |
| from | ... | ... | ... | ... | ... | ... | ... | ... |
| the | ... | ... | ... | ... | ... | ... | ... | ... |
| Chinese | ... | ... | ... | ... | ... | ... | ... | ... |

Figure 3: Calculation of the Self-attention Mechanism. Numbers are for illustration purposes only.

on the top in Figure 2 and position embeddings shown at the bottom are summed together to construct the input embeddings to a transformer encoder illustrated in Figure 1.

### The Self-Attention Mechanism

The core of the transformer encoder is the multi-head self-attention layer. Before covering multi-head self-attention, I first discuss the (single-head scaled dot-product) self-attention mechanism.

Consider *E1* and *E2* again. The mere syntactic arrangement fails to capture the polysemous nature of the word *"yellow"*. For a nuanced and context-rich representation of the word in *E1*, it is necessary to establish a semantic linkage between *"yellow"* and contextual markers such as *"evil"* or *"Chinese"*. In the parlance of transformers, the word *"yellow"* should pay *attention* to other surrounding words, creating connections that vary in strength based on context. The key advantage of transformers is their ability to not only estimate word positions in a semantic space, but also the relative importance of a given word in shaping the meaning of other words (Vaswani et al., 2017). This is achieved through the incorporation of the scaled dot-product self-attention mechanism.

To disentangle the self-attention mechanism mathematically, imagine I want to calculate

choice.

the contextualized embeddings incorporating self-attention scores for *E1*, as illustrated in Figure 3. The self-attention mechanism takes in an input embedding matrix $X$ (i.e., the encoder input embedding matrix shown in Figure 2 and in the middle of Figure 1) and outputs a contextualized embedding matrix $X^*$. $X^*$ is generated using the self-attention[13] score matrix $A$ through:

$$X^* = \text{Self-Attention Score}(X) \cdot X = A \cdot X = softmax(\frac{X \cdot X^T}{\sqrt{d}}) \cdot X \tag{1}$$

where:

$X$ = the encoder input embedding matrix,

$d$ = dimensionality of the model (see footnote 10.)

For *E1*, the input sequence length is 8 and the dimension $d$ of the model and input embedding is 512,[14] which is the dimension of the encoder in the original paper that introduced transformers (Vaswani et al., 2017). As shown in Figure 3, I first conduct dot product matrix multiplication of $X$ and the transpose of $X$, and divide it by a scaling factor, usually the square root of the input embedding dimension. The resulting matrix is then passed through a softmax function[15] for normalization, resulting in an 8 by 8 matrix where all the rows sum up to 1. The values in this resulting matrix, as shown in the blue matrix $A$ in Figure 3, are the self-attention scores, where they represent the strengths of

[13]There are different types of attention mechanisms, such as self-attention, cross-attention, and causal-attention. Each is a slight variation of the original attention mechanism. For self-attention, the Query, Key and Value matrices are replications of the same input embedding matrix, i.e. $Q = K = V =$ the input embedding matrix$X$.

[14]In this example, I use $d = d_{model} = d_k = 512$. The specific number can change with different transformer architectures.

[15]The softmax function, or multi-nomial logistic regression, is a generalization of logistic regression to the case where we want to handle multiple classes.

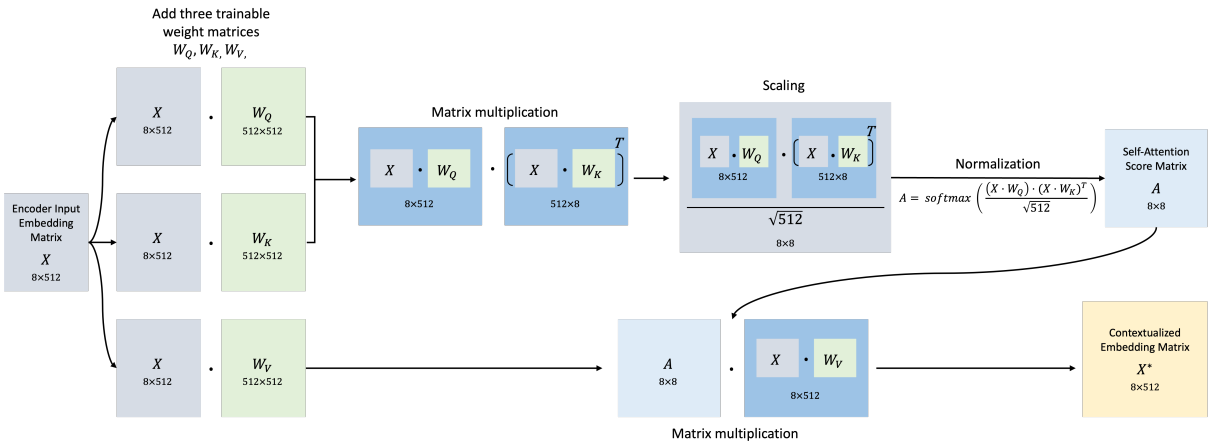



Figure 4: Self-attention with Trainable Weight Matrices $W_Q$, $W_K$, and $W_V$.

the relationships between one token and another. If a particular token is not relevant to the understanding of the target token, the self-attention mechanism may assign a very low or zero score to that position, effectively ignoring its contribution. Higher self-attention scores correspond to stronger relationships between two tokens. Intuitively, I can think of the word *"yellow"* consisting of 0.01 *Stop* + 0.03 *the* + 0.15 *evil* + 0.33 *yellow* + 0.25 *invasion* + 0.04 *from* + 0.02 *the* + 0.17 *Chinese*, which sums to 1. Different from the input embedding generation, the same token *"the"* can have different self-attention scores when they appear multiple times in the input sequence.

To obtain the contextualized embedding matrix $X^*$ (i.e., the yellow matrix in Figure 3) from the self-attention mechanism, I conduct another dot product matrix multiplication of $A$ and $X$. Each row in the resulting matrix $X^*$ contains the contextualized embedding of the initial input sequence, where the self-attention scores and position embeddings have been incorporated into the resulting embeddings.[16]

At this stage, the calculation of self-attention requires no trainable parameters. Up to now the interaction between tokens has been driven by their own embeddings. To add trainable parameters to the model, I can add updatable weight matrices. Although in the

[16]Values along the diagonal of the self-attention matrix are expected to be highest due to the dot-product calculation, so each token pays the most attention to itself.

calculation of self-attention based on Equation 1 I use the same input embedding matrix $X$ three times, I can multiply each $X$ by different weight matrices. I conduct the same mathematical operation with the trainable matrices $W_Q$, $W_K$, and $W_V$, as illustrated in Figure 4. This operation yields the contextualized embedding matrix $X^*$ as the output.[17]

### The Multi-Head Self-Attention Layer

The entire process described so far is everything that happens within a module called an attention head. When using only one set of weight matrices $W_Q$, $W_K$, and $W_V$, it is called single-head attention. Often times, single-head attention is not enough to capture complex relationships among tokens in the same sequence. Therefore, multi-head attention is used to make the model attentive to different aspects of the tokens.

The only change for multi-head attention is that I have one set of weights for each head, and concatenate the results from all heads afterwards. I continue to use *E1* for illustration, where the encoder input embedding matrix is of size $8 \times 512$. To better illustrate multi-head attention, consider a transformer encoder with two attention heads.[18] Figure 5 shows an overview of multi-head self-attention with two attention heads.

As illustrated in Figure 5, notice that the encoder input embedding matrix is split into two smaller matrices $X_1$ and $X_2$ to use as the input embedding matrix for each attention head. The input embedding is typically split across the embedding dimension (columns) instead of the sequence length dimension (rows).[19] Such approach allows separate sections

[17]The notations used in this article are slightly different from Vaswani et al. (2017) and the computations are simplified to omit unnecessary details for illustration purposes only.

[18]In practice, most language models have more than two attention heads. For example, BERT has 12 attention heads per encoder layer, and GPT models have over hundreds of attention heads per layer.

[19]Note that this is a logical split only. The $W_Q$, $W_K$, and $W_V$ matrices are not physically split into separate matrices in implementation. A single matrix is used for $W_Q$, $W_K$, and $W_V$ respectively with logically separate sections of them for each attention head.

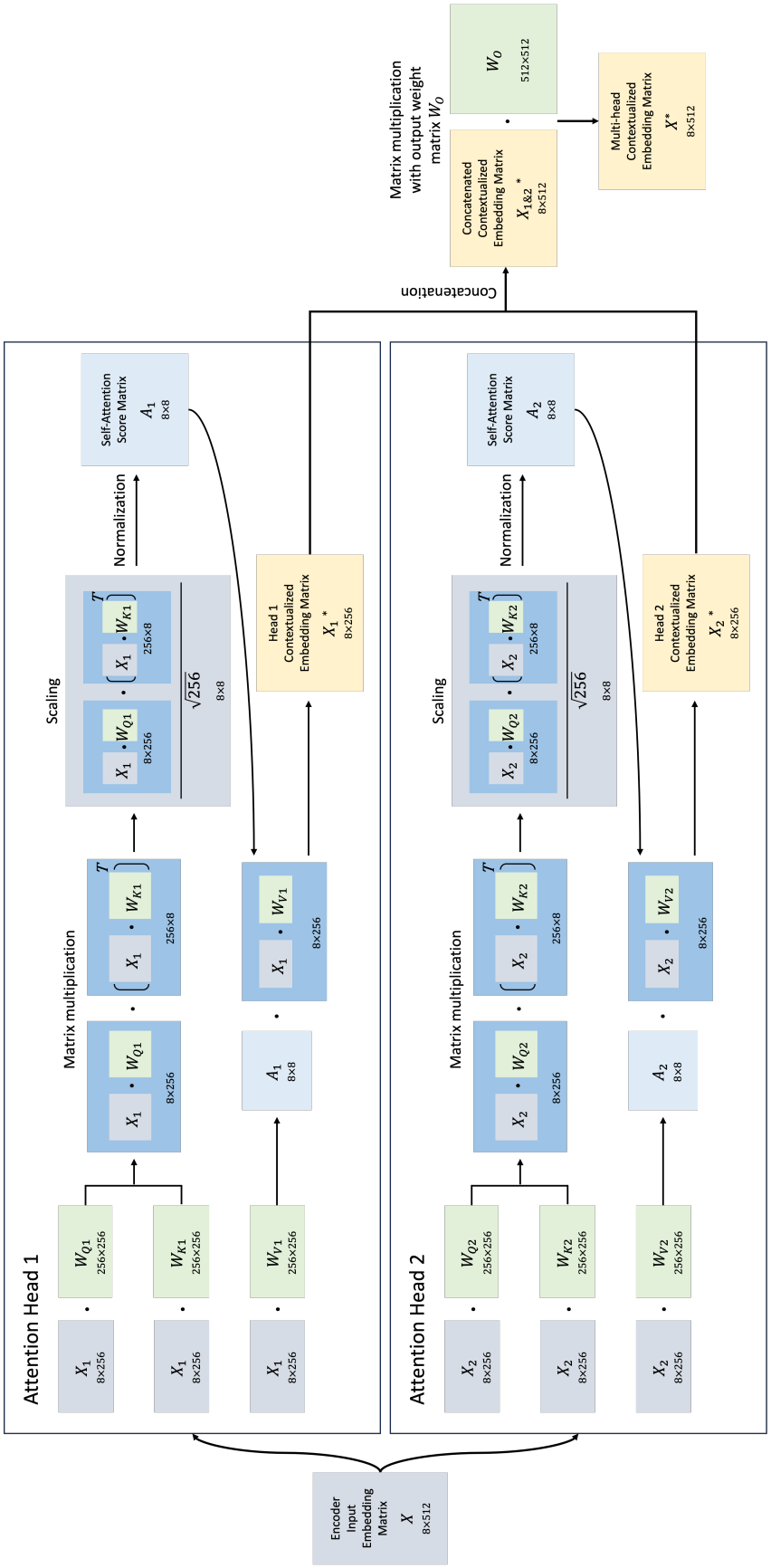



Figure 5: Calculation of the Multi-head Self-attention with Two Attention Heads.

of the input embedding to learn different aspects of the meanings of each token as it relates to other tokens in the sequence, which allows the transformer encoder to capture richer interpretations of the sequence.[20] With the smaller matrices, I conduct the same calculation of self-attention within each attention head in the same way as the single-head self-attention as shown in Figure 3. Then I concatenate the output matrices, i.e., the contextualized embedding matrices, from each attention head to obtain $X_{1\&2}{}^*$. To learn the optimal way to combine different output matrices from all attention heads, I add another trainable weight matrix $W_O$ before obtaining the final multi-head embedding matrix $X^*$ which contains the contextualized embedding of the input text after the multi-head self-attention mechanism. Once training is completed, the model saves all the weight matrices (i.e., trained parameters) for later use.

### Feed-forward Neural Network Layer

The multi-head self-attention mechanism is the core innovation of the transformer encoder. Following Figure 1, the next component[21] in the transformer encoder is a feed-forward neural network (FNN) (Lin and Lucas, 2024).[22] The FNN[23] receives the output from the multi-head self-attention layer, denoted as $X^*$ in Figure 5, and processes it through

[20]For instance, one section might capture the "gender-ness" of a noun while another might capture the "cardinality" of a noun. This example may not be realistic, but it could help to build intuition.

[21]There are also other components in the transformer encoder that assist the model, such as the residual connections and layer normalization, that will not be covered in detail in this paper. Please refer to the original paper for a more detailed discussion (Vaswani et al., 2017).

[22]For a comprehensive introduction to FNNs, Lin and Lucas (2024) provide an accessible guide for social scientists, explaining neural networks by building on concepts of logistic regression.

[23]FNNs in the transformer encoder often consist of two fully-connected layers with a non-linear activation function, such as ReLU, applied in between them. The number of layers in the FNN is configurable and typically repeated across multiple encoders stacked together.

dimensionality expansion and reduction to produce the encoder's final output (Lin and Lucas, 2024).

While the multi-head self-attention mechanism enables the model to handle long-range dependencies within the data, the FNN enhances the model's capacity to learn complex and non-linear relationships.[24] This capacity is crucial for classification tasks requiring language comprehension, where the meaning of a word often relies on its context within a sentence or even across a paragraph.

## Pre-training and Fine-tuning

In the following section, I will examine the PFP of encoder-only language models,[25] which typically comprise multiple transformer encoders, resulting in millions or billions of trainable parameters. Such large-scale models require extensive datasets to ensure diverse and ample examples for effective learning. However, these expansive datasets are often not accessible to applied scholars. Transfer learning offers a solution to this challenge, allowing models pre-trained on large datasets by research institutions and corporations to be fine-tuned on task-specific datasets with minimal resources.

During the pre-training phase, the parameters of the transformer encoders are learned using self-supervised learning on a vast corpus of unlabeled text, such as the Common Crawl. The objective of pre-training is to enable the model to acquire a general understanding of linguistic, semantic, and syntactic patterns in the language. This phase effectively initializes the model's parameters in a contextually informed manner, as opposed to random initialization, which lacks relevance to specific downstream tasks (Ruder et al., 2019).

In the fine-tuning stage, the pre-trained model is further adapted to address a specialized task, such as toxic language classification, by training on a smaller, labeled dataset

[24]The non-linearity is introduced in the FNN through the usage of non-linear activation functions.

[25]Types of language models are discussed in footnote 7.

(e.g., a corpus of annotated tweets). Unlike pre-training, fine-tuning is generally not resource-intensive, and can yield substantial performance improvements for tasks with limited annotated data. In this article, I propose the PFP approach due to its potential to deliver high performance in text classification tasks, while giving researchers greater control over the data that the model learns from.

The PFP has gained widespread adoption in natural language processing due to its effectiveness (Beltagy, Lo and Cohan, 2019; Caselli et al., 2021; Devlin et al., 2019; Nguyen, Vu and Nguyen, 2020; Qu et al., 2019; Yang et al., 2020). Two recent papers have also fine-tuned a pre-trained language model on political science-related downstream tasks. The first paper by Bestvater and Monroe (2022) uses a fine-tuned BERT model to study stance detection for political text analysis and demonstrated that it outperformed the standard dictionary-based approach. ELECTRA, an extended version of the classic BERT, has also been used to measure emotional appeals in German political discourse (Widmann and Wich, 2022). However, these papers lack detailed justification for the adoption of transformer-based language models and provide limited explanation of the PFP. Before turning to my own application, I will further explore the processes involved in pre-training and fine-tuning, and explain the necessity of both stages within the PFP.

### Pre-training Stage

To contextualize the following discussion, I use the classic BERT[26] as an example of encoder-only models (Devlin et al., 2019). Encoder-only models exploit transfer learning

[26]BERT is an encoder-only model that uses a vocabulary of 30,522 tokens. Input tokens are converted to 768-dimensional word embeddings and passed through 12 transformer encoders. Each contains a self-attention mechanism with 12 attention heads. The $W_Q i$, $W_K i$, and $W_V i$ weight matrices are of dimension $768 \times 64$ for each head $i$. The total number of parameters is approximately 110 million. When BERT was introduced, it was considered large, but it is far smaller than the state-of-the-art models. I use BERT to refer to the BERT-base model. For BERT-large, input tokens are converted to 1024-dimensional word embeddings and passed through 24 transformer encoders, where each contains a self-attention

(Zhuang et al., 2021).[27] In the context of classification tasks, such as toxic language detection, researchers usually start to train neural networks by randomly initializing the weights from a specified random seed. Intuitively, as the training process begins, these weights are continually updated to perform the task with fewer errors, a process often referred to as optimization. During the pre-training stage, the model is trained using self-supervised learning, which allows the use of enormous amounts of data without the need for manual labels. For example, for BERT, the self-supervised pre-training task consists of predicting missing words from sentences (i.e., Masked Language Modeling) from a large internet corpus of unlabelled text including the entire Wikipedia and Book Corpus.[28]

The pre-training process of a BERT-like encoder-only model is shown in Part 1 and Part 2 in Figure 6. The input tokens (along with a special `<CLS>` token denoting the start of the sequence specific for BERT-like models) are converted to token embeddings through the embedding layer as described in the previous subsection on Transformer Encoder Input Embedding Generation and Figure 2. These embeddings are passed through a series of transformer encoders to create a set of output embeddings as illustrated in the subsection on the Multi-head Self-attention Layer and Figure 5. The first encoder receives the output from the embedding layer as the input, while the subsequent encoders receive the output from the previous encoder as their input. A small fraction of the input tokens is randomly replaced with a generic `<mask>` token.[29] In pre-training, the goal is to predict the missing

mechanism with 16 heads. The total number of parameters is approximately 340 million.

[27]Transfer learning refers to the method of acquiring knowledge from one task or domain and then applying it or transferring it to solve a new task. Pre-training refers to training a neural network on some dataset in advance of any downstream tasks.

[28]BERT also uses a secondary task that predicts whether two sentences were originally adjacent in the text or not, but this only marginally improves performance (Liu et al., 2019).

[29]The implementation is a little more complicated. In BERT, 15% of the input tokens in a training

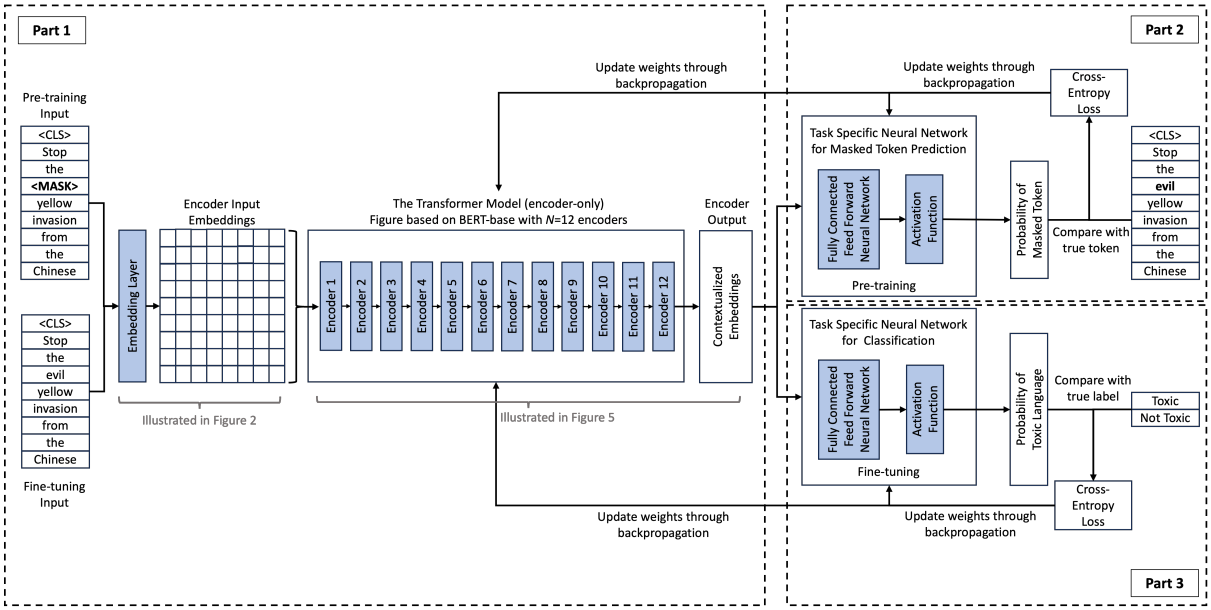



Figure 6: Pre-training and Fine-tuning for BERT-like Encoder-only Models. Shaded boxes contain trainable parameters.

token from the associated output embedding.[30] We then calculate the loss[31] from each of the model's predictions to update the weights for all learnable layers as highlighted in shaded boxes in Figure 6 through back-propagation by calculating the gradients or derivatives of the loss with respect to the weights (Devlin et al., 2019).

In the PFP, researchers can optionally *refine* a pre-trained model to improve classifi-

sequence are sampled for learning. Among them, 80% are replaced with `<mask>`, 10% are replaced with randomly selected tokens, and the remaining 10% are left unchanged. Note that all of the input tokens play a role in the self-attention process, but only the sampled tokens are used for learning.

[30]This task has the advantage that it uses both the left and right context to predict the missing word to assist language understanding, comparing to decoder-only models like GPT that only rely on single-sided context.

[31]Cross-entropy loss is used for most encoder-only models to measure the error between two probability distributions. For instance, for a binary classification task with two classes 0 and 1, the cross-entropy loss is $L = -\frac{1}{N}\left[\sum_{i=1}^{N}[t_i log(p_i) + (1-t_i) log(1-p_i)]\right]$ for $N$ data points where $t_i$ is the truth value taking a value 0 or 1 and $p_i$ is the softmax probability for the $i^{th}$ data point.

cation performance by repeating the pre-training process, such as training with masked language modeling on domain-specific data, to update trainable parameters in both Part 1 and Part 2 shown in Figure 6. Part 1 is then saved as the refined model to use later for fine-tuning tasks with Part 3. However, for researchers with limited computational resources, fine-tuning on a good quality labeled dataset could also achieve similar performance as discussed in the following subsection.

**Fine-tuning Stage**

Considering the substantial parameterization inherent to language models,[32] pre-training or refining a pre-trained model can incur significant computational expenses and necessitate extensive datasets. Rather than conducting the pre-training ourselves, political science researchers can leverage pre-trained models previously developed and made available by companies or research institutions worldwide,[33] thereby providing a viable starting point for ourselves.

Transformer-based pre-trained language models like BERT perform well on general-purpose language tasks but may struggle with domain-specific tasks like toxic language classification. Subsequently, scholars can fine-tune the pre-trained models to accommodate diverse downstream tasks using task-specific data, thereby capitalizing on the general linguistic knowledge established during the pre-training stage (Barbieri et al., 2020; Caselli et al., 2021; Chalkidis et al., 2020; Lee et al., 2020). In the fine-tuning stage, the model parameters are updated to specialize the model to a particular task.

As illustrated in Part 1 and Part 3 in Figure 6, to fine-tune a pre-trained model like BERT, scholars can append a task specific neural network on top of the encoder-only

[32]Models often have millions or billions of parameters. It is usually not advised to perform pre-training on a regular laptop.

[33]Platforms like the HuggingFace Hub has over 350K open-sourced models that are publicly available for anyone to use. https://huggingface.co/models

model and train the entire model with the a small annotated dataset. Instead of comparing the predicted token with the original masked token in pre-training, during fine-tuning for classification, the model compares the predicted label with the true label to compute the loss and update the weights through back-propagation as shown in Part 3 in Figure 6. Learned linguistic features such as syntactic and semantic roles from the pre-trained model are largely preserved in the fine-tuned models (Merchant et al., 2020).

Compared to pre-training, fine-tuning is a computationally inexpensive approach that requires considerably smaller volumes of data while affording adaptability to specific tasks. In the section on Application to Toxic Language Classification, I use the PFP approach on a toxic language classification task to demonstrate its data efficiency, training efficiency, and performance enhancement capabilities. I show that models trained with PFP outperform the traditional text classification methods reviewed in the section on Current Text Classification Approaches using an annotated sample Twitter dataset of conversation threads following tweets from US Senators one month before and after the 2020 Presidential Election.

## Guidelines of the PFP

There are several ways for scholars to use the PFP with different levels of data and compute availability. The entire text classification workflow is shown in Figure 7.

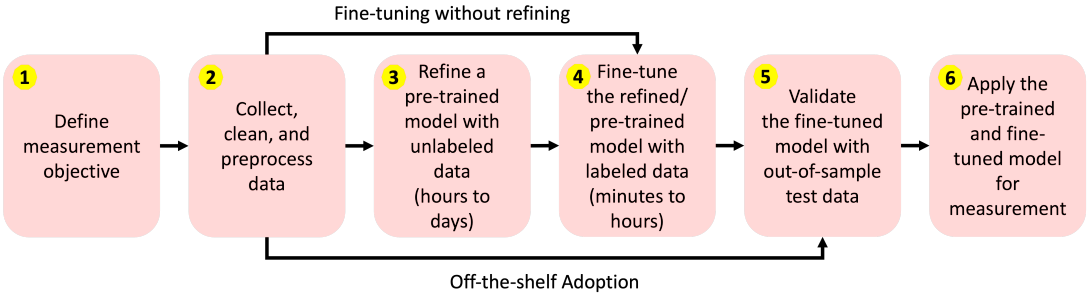



Figure 7: Text Classification Workflow Following the Pretrain-Finetune Paradigm. Intermediate steps can be skipped when working with limited computational resources and data.

If researchers have no annotated data, the simplest way to adopt the PFP is to download

publicly available classifiers already trained using the PFP and apply them directly datasets at hand. This approach offers the key advantage of minimal computational cost, as the model has already undergone the resource-intensive pre-training and fine-tuning stages. However, a potential drawback of this direct application of models is the difficulty in finding an existing model that aligns perfectly with the specific needs of the task at hand.

When deciding whether to bypass steps outlined in Figure 7, researchers may initially employ off-the-shelf models and validate their performance using labeled data. In the following section, I show that while off-the-shelf models can be valuable in certain scenarios, they may also prove ineffective in others. Consequently, rigorous validation is essential when utilizing models with lower levels of customization.

For researchers with some annotated data and limited compute resources, they can download a pre-trained model and use the generated embeddings to train a classifier on top of the model, where only Part 3 in Figure 6 is updated.

For researchers with more compute resources, a more tailored approach is to start with a pre-trained model that has been trained on a relevant corpus, and then fine-tune it using a small annotated dataset, such as a few hundreds labeled tweets, where both Part 1 and Part 3 in Figure 6 are updated accordingly. This hybrid strategy of leveraging transfer learning combined with targeted fine-tuning is the most recommended approach for applied scholars in the PFP, as it balances the computational and data efficiency of using a pre-existing model with the ability to customize the model to their specific research requirements.

For scholars with access to more computational resources, another option is to engage in the entire pre-training and fine-tuning process from steps 1 through 6, allowing for maximum customization of the language model to their preferences and the task at hand. They can download a pre-trained model and refine the model by re-doing the pre-training tasks to update Part 1 and Part 2 in Figure 6 and save Part 1 as the refined model. Then conduct fine-tuning to update the Part 1 and Part 3 accordingly to obtain the final classification model. This approach, while more computationally intensive, provides the

greatest flexibility in tailoring the model to the unique needs of scholars.

In the next section, I demonstrate the power of the PFP through a real-world application on toxic language classification. To illustrate the potential of the PFP in non-English content, I also replicate a paper by Chang and Masterson (2020) where they applied Long Short-term Memory (i.e., LSTM) along with word2vec to classify over ten thousand Chinese Weibo posts into political versus non-political categories. More details are discussed in Appendix A (p. 1).

# Application to Toxic Language Classification

This section provides a step-by-step walk-through of the guidelines illustrated in Figure 7, with an application to toxic language classification. The objective is to systematically compare the performance of various PFP models against each other and against baseline models to assess their effectiveness. Table 1 provides a complete list of all models used, which will be discussed in greater detail in the subsequent sections.

## Data Preparation

The proliferation of toxic language has intensified with the widespread use of social media (Agarwal et al., 2021; Chandrasekharan et al., 2017; Massaro and Stryker, 2012; Matamoros-Fernández and Farkas, 2021; Rheault, Rayment and Musulan, 2019; Tamara et al., N.d.). Accurately measuring toxicity within large datasets remains challenging due to the absence of an unambiguous and universally accepted definition. Appendix B (p. 2) offers a more detailed discussion of the definitional debate on toxicity. While resolving the definitional debate goes beyond the scope of this article, to demonstrate the effectiveness of the PFP, I adopted a comprehensive and operationalizable definition of toxicity offered by Poletto et al. (2021), where toxicity is defined as *"any impolite, rude or hurtful language that can show a debasement of someone or something, or show intense emotion, including hate speech, derogatory language and also profanity"* (Poletto et al.,

2021).

To apply this framework, I collected a dataset comprising 20,072 tweets[34] posted by U.S. Senators from the $116^{th}$ U.S. Congress between October 1 and November 30, 2020, using the Twitter Academic API. Additionally, conversation threads[35] generated in response to each of these tweets over the following year were gathered, yielding a total of 1,652,973 tweets. From this corpus, I randomly sampled 3,000 tweets to be labeled by human annotators recruited via Amazon Mechanical Turk. All annotators were required to complete a training module and pass a qualification test to ensure consistency and accuracy in annotation. Each tweet was annotated by two independent coders, achieving an inter-coder reliability rate of 93%. Detailed information on the data annotation process is available in Appendix C (p. 3-11).

For consistency in evaluation, a common test set (consisting of a random 10% sample from the annotated dataset) was used to assess all models listed in Table 1. Examples of tweets from the dataset are presented in Table 2. I assess the balanced accuracy scores of each model alongside their computational costs in terms of training time. Results indicate that all fine-tuned PFP models outperform the baseline models for this text classification task, regardless of refining. Notably, even selected off-the-shelf PFP models yield significantly improved performance compared to the baseline models.

## Four Approaches to Apply the PFP

The first approach I explore is using off-the-shelf models trained for toxicity detection by other researchers. In this approach, I simply downloaded Models 1-6[36] shown in Table 1

[34]Entire tweet histories were collected, including retweets and quote tweets.

[35]The threads include publicly available direct replies to the parent tweets and replies of replies to these tweets.

[36]Models 1-3 were pre-trained using the RAL-E dataset that contains English Reddit comments from banned communities and then fine-tuned (Caselli et al., 2021). Models 4-6 were fine-tuned based on the

Table 1: List of All Models Used in the Section on Application to Toxic Language Classification

| Model Number | Model Name | Category | Training Time (min:sec) |
|---|---|---|---|
| 1 | HateBERT-abuseval | off-the-shelf PFP | 00:00 |
| 2 | HateBERT-offenseval | off-the-shelf PFP | 00:00 |
| 3 | HateBERT-hateval | off-the-shelf PFP | 00:00 |
| 4 | BERT-abuseval | off-the-shelf PFP | 00:00 |
| 5 | BERT-offenseval | off-the-shelf PFP | 00:00 |
| 6 | BERT-hateval | off-the-shelf PFP | 00:00 |
| 7 | HateBERT-Twitter | fine-tuned PFP | 01:14 |
| 8 | BERT-base-uncased-Twitter | fine-tuned PFP | 01:14 |
| 9 | BERT-base-cased-Twitter | fine-tuned PFP | 01:21 |
| 10 | BERT-large-uncased-Twitter | fine-tuned PFP | 03:30 |
| 11 | BERT-large-cased-Twitter | fine-tuned PFP | 03:46 |
| 12 | freeze-BERT-base-uncased-Twitter | only trained classifier PFP | 00:32 |
| 13 | refined-BERT-base-uncased-Twitter | refined and fine-tuned PFP | 46 hrs + 01:19 |
| 14 | Dictionary-based | baseline | 00:00 |
| 15 | BOW + Naive Bayes | baseline | 00:19 |
| 16 | fastText + linear classifier | baseline | 00:22 |
| 17 | Google's Perspective API | baseline | 00:00 |

and applied them directly to classify the test set without any further customization.[37] This approach offers the distinct advantage of convenience and efficiency, as it eliminates the need for extensive computational resources and time required for additional pre-training and fine-tuning. However, a significant limitation of using off-the-shelf models is the lack of task-specific customization. Since these models were not explicitly designed for the toxicity detection task in tweets, their performance may be suboptimal compared to models that are further fine-tuned. Additionally, reliance on models developed by

*BERT-base-uncased* model. The fine-tuning datasets used by Caselli et al. (2021) were three annotated and publicly available toxic language benchmark datasets: *AbusEval*, *OffensEval*, and *HatEval*.

[37]Simple code examples are provided in Appendix D (p. 12-14) for all four approaches mentioned in this section. For reproducibility, all computation in this article is conducted on Google Colab on an A100 GPU. All PFP models used in this article can be found on HuggingFace Hub https://huggingface.co/AnonymousCS. Full Python code are provided in the replication archive.

other researchers can introduce unknown biases embedded within the training data of the original researchers, potentially impacting the validity of toxicity detection in political contexts. Thus, while off-the-shelf models provide a practical starting point, they may lack the precision and adaptability achieved through fine-tuning tailored to the specific needs of the current study.

The second approach is to fine-tune a pre-trained language model with a toxic language classification task using my annotated dataset described above, where both Parts 1 and 3 are updated in Figure 6. This method offers the advantage of enhanced model adaptability, as fine-tuning enables the model to learn features and patterns that are more directly relevant to the specific task and dataset, potentially leading to improved classification accuracy. However, a drawback is the increased computational cost and time associated with the fine-tuning process, particularly for large models with numerous parameters. In this study, I fine-tuned four versions of widely used pre-trained BERT models (i.e., *bert-base-uncased*, *bert-base-cased*, *bert-large-uncased*, and *bert-large-cased*) as well as a BERT model refined on toxicity-related dataset (i.e., *HateBERT*) to generate Models 7-11 (Caselli et al., 2021).

To reduce computational cost, a third option is to create a model where only the task-specific layer (shown as Part 3 in Figure 6) is updated, while keeping the parameters of the encoders (shown as Part 1 in Figure 6) fixed. This is implemented in Model 12, where I updated only the classification layer with *BERT-base-uncased* using the labeled dataset, leaving the parameters of the encoder layers unchanged. The primary advantage of this approach is its efficiency: by not adjusting the large number of parameters in the encoders, the computational requirements and time necessary for fine-tuning are significantly reduced. However, a notable disadvantage is the potential reduction in model performance, as the fixed encoder parameters may limit the model's ability to learn nuances specific to the task and dataset. Consequently, while this approach can provide a cost-effective solution, it may fail to achieve the same level of accuracy as models that undergo full fine-tuning, particularly in cases where task-specific adaptation is critical for

optimal performance.

The final approach involves re-training the entire model, as illustrated in Figure 6, using both labeled and unlabeled data. This comprehensive re-training procedure first uses unlabeled data to refine the pre-trained language model (shown as Part 1 in Figure 6) via training with a masked token prediction task (shown as Part 2 in Figure 6). Then using the labeled data to fine-tune the refined model for a classification task to further update Part 1 alongside Part 3 shown in Figure 6. The principal advantage of this method is that it enables the model to develop a more nuanced understanding of both general language patterns and the unique characteristics of the specific task, potentially enhancing performance on complex tasks that require deep contextual understanding. However, this approach also has a significant disadvantage: it is computationally intensive, requiring considerable time and processing resources, which may not be feasible for resource-limited scholars.

In this study, I refined the *BERT-base-uncased* model by re-training it on a corpus of over 1.65 million unlabeled tweets using a masked token prediction task. Following re-training, I incorporated the annotated tweets to further fine-tune the model on the toxic language classification task in a manner consistent with Models 7-11. This refined-and-finetuned model is denoted as Model 13 in Table 1. The training process, executed on Google Colab with an A100 GPU, required approximately 46 hours to complete.

Additionally, four baseline models (listed as Models 14-17 in Table 1) using traditional classification methods discussed in the Current Text Classification Approaches section are included for comparison. Detailed descriptions of these baseline models are available in Appendix E (p. 15-17).

## Model Performance Comparison

In this section, I first compare the out-of-sample performance of various PFP models listed in Table 1 against baseline models, focusing on balanced accuracy scores. Second, I assess the face validity of the measure of toxicity for the best-performing PFP model against the

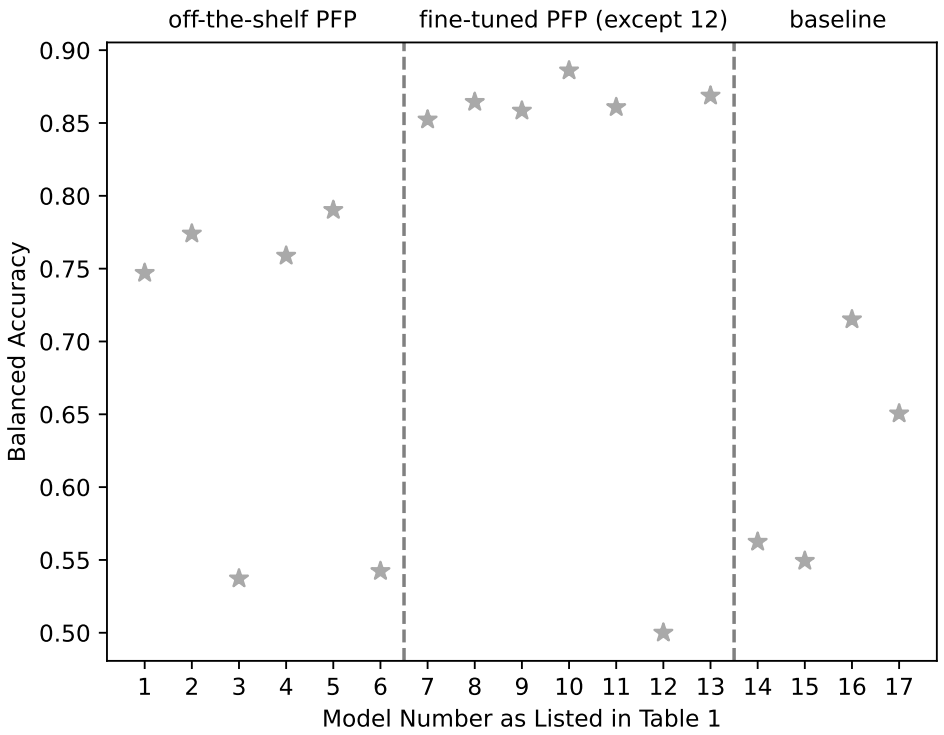



Figure 8: Balanced Accuracy Scores of All Models.

baseline models. This involves examining example classifications produced by the models to verify that identified toxic content aligns with both theoretical definitions and human intuition. Finally, I apply the validated measure to detect toxicity in a real-world dataset. Specifically, I analyze responses to tweets from U.S. Senators to explore the exacerbation of toxicity in public discourse online. Through these stages, I aim to establish both the robustness and applicability of the PFP in analyzing politically relevant data.

Figure 8 illustrates the balanced accuracy scores on the out-of-sample test dataset for all models listed in Table 1. Detailed evaluation results for each model category (i.e., off-the-shelf models, fine-tuned PFP models, and baseline models) are presented in Tables 1, 2, and 3 in Appendix E (p. 15-17). Across all metrics, the fine-tuned PFP models exhibited significantly improved performance relative to the baseline models consistently. Among these, Model 10 achieved the highest balanced accuracy score of 0.8860, highlighting its effectiveness for toxic language classification.

As shown in Table 1, fine-tuning times for the models varied, with most models

requiring only a few minutes. The longest fine-tuning duration was recorded by Model 11, taking 3 minutes and 46 seconds. In contrast, Model 12, which involved updating only the task-specific layer rather than performing full fine-tuning, required just 32 seconds for training, but displayed a notable reduction in performance. This underscores the importance of updating the parameters of the encoders during fine-tuning to maximize model performance for a given task. Even without additional training, certain off-the-shelf models (Models 1, 2, 4, and 5) achieved balanced accuracy scores exceeding that of the best-performing baseline model (Model 16). The high performance among off-the-shelf models may, in part, be due to the alignment between toxicity definitions in the annotated *OffensEval* and *AbusEval* dataset, which were used to fine-tune these models, and the definition applied within my annotated Twitter dataset. This highlights the importance of validating off-the-shelf models to ensure they align well with the specific requirements of the target dataset. Interestingly, the process of refining the pre-trained model prior to fine-tuning, which required a substantial 46 hours of computational time, did not produce any measurable improvement in classification accuracy compared to models that were simply fine-tuned. The result suggests that the fine-tuning approach is sufficient to achieve optimal results in this specific application. Consequently, these findings underscore that while refining can sometimes enhance model performance, the gains may be marginal considering the high computational cost.

To enhance interpretability of the models' classification decisions, I employed the integrated gradients algorithm to calculate attribution scores for each token when classifying the input tweet as toxic or not (Sundararajan, Taly and Yan, 2017). Figure 3 in Appendix F (p. 18-19) shows example tweets where the attribution scores for each token are visualized. This interpretability approach allows for an intuitive understanding of the models' decision-making process by quantifying each token's influence on the predicted toxicity score. More comprehensive details regarding the implementation and interpretive value of integrated gradients in this context are provided in Appendix F (p. 18-19).

Table 2: Example of Disagreed Tweets in the Test Set.

| | | PFP | Baseline | | | |
|---|---|---|---|---|---|---|
| **Text** | **True Label** | **Model 10** | **Model 14** | **Model 15** | **Model 16** | **Model 17** |
| "@SteveDaines @JohnCornyn You Republicans have done enough damage-not one of you had the guts to say to tell Trump this needed to be dealt with at the National level. Pathetic a d Disgusting." | 1 | 1 (0.9958) | 0 | 1 | 1 | 1 |
| "@SenTomCotton @nytimes Racist white boy!" | 1 | 1 (0.9944) | 1 | 0 | 0 | 1 |
| "@KamalaHarris Thanks for the reminder .... I went and voted to keep your skanky body out of the White House ..... LOL" | 1 | 1 (0.9942) | 1 | 1 | 0 | 0 |
| "@ChrisVanHollen @WesMoore @MarysCenter Boo Hoo your whining and crying again VanHollen" | 1 | 1 (0.9907) | 0 | 0 | 0 | 0 |
| "@marcorubio Titus 1:16 _ They profess to know God, but they deny him by their works. They are detestable, disobedient, unfit for any good work. #evilrubio" | 1 | 1 (0.9827) | 0 | 1 | 0 | 0 |

### Assessment of Face Validity

Beyond evaluating classification metrics, I also assess the face validity of PFP models in comparison to the baseline models. To this end, I examine instances where Model 10, the highest-performing PFP model, classified tweets as toxic with highest probabilities, while baseline models yielded divergent classifications. Table 2 highlights examples of such classifications.[38] In many cases, the baseline models failed to detect toxicity in tweets where neutral words, such as "crying" were used with negative or derogatory intent. This finding suggests that the baseline models struggle with subtle contextual cues that indicate toxicity, while Model 10's more nuanced understanding of context allows for improved identification.

I further investigate the aggregate classification outcomes, focusing on the total number of tweets labeled as toxic by Model 10 compared to baseline models. Despite aligning closely in ranking Senators based on the volume of toxic tweets in their conversation threads, substantial discrepancies arise in the actual counts. Specifically, Model 15 identifies a considerably higher number of toxic tweets, exceeding Model 10's total count by an average of 55.9% across all Senators, whereas Models 14 and 16 report fewer toxic tweets, with reductions of 20.9% and 4.5% respectively.

This discrepancy remains consistent when disaggregating results by gender and party

[38]I also examined tweets where Model 10 labeled as non-toxic, but one or more baseline models classified as toxic. In several cases, the true label was incorrect and missed certain toxic content.

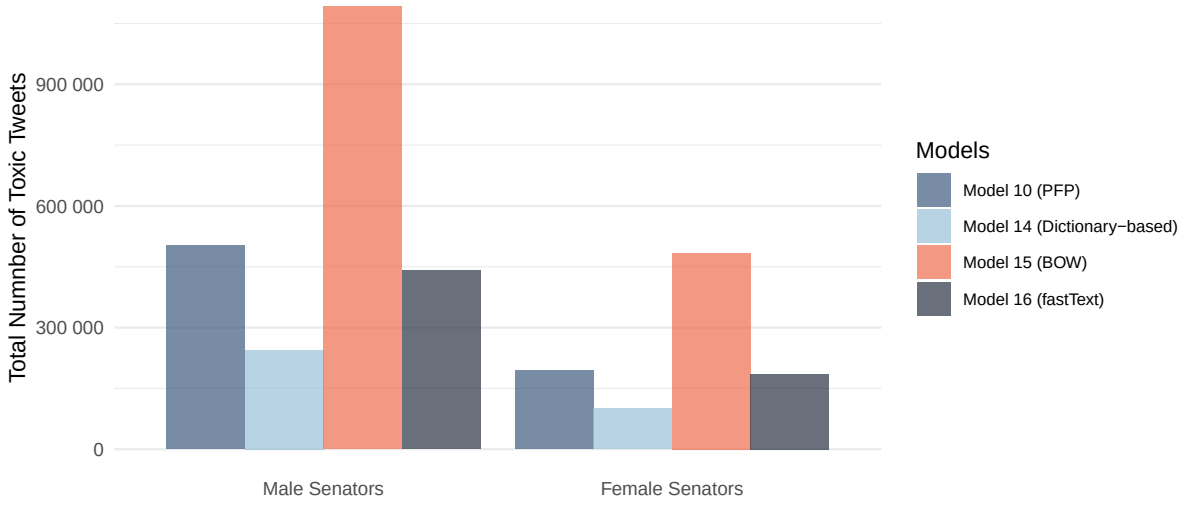



Figure 9: Bar-plots of Total Number of Toxic Tweets Labeled by Different Models by Gender

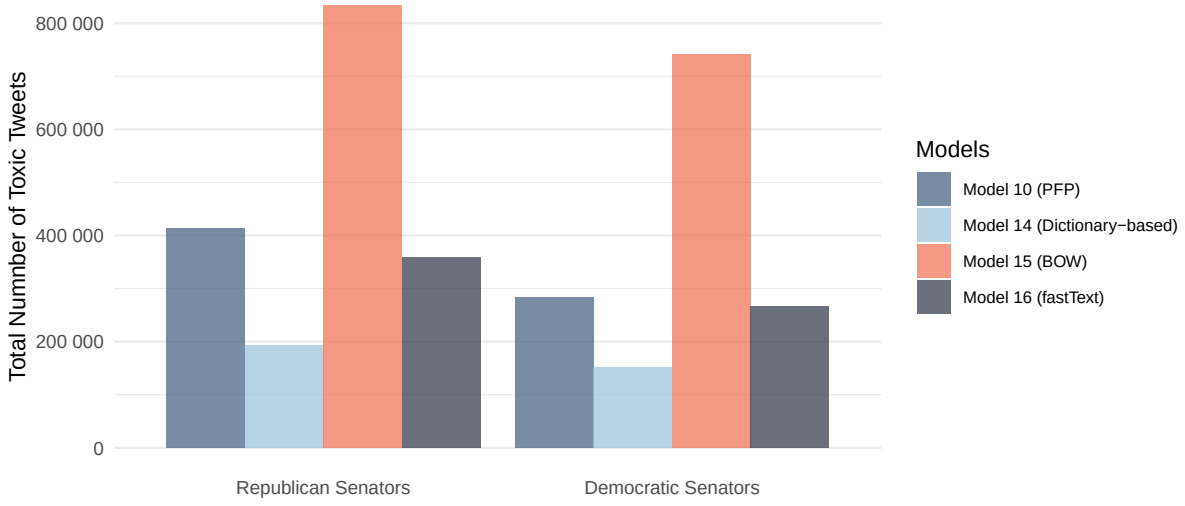



Figure 10: Bar-plots of Total Number of Toxic Tweets Labeled by Different Models by Party

affiliation. For female Senators, as shown in Figure 9, Model 15 registers a 58.3% increase in toxic tweet counts compared to Model 10, while Models 14 and 16 show reductions of 19.2% and 2.9% respectively. Among male Senators, Model 15 reports 55.0% more toxic tweets than Model 10, with Models 14 and 16 reporting reductions of 21.5% and 5.0% respectively.

The trend is also evident when comparing toxic tweet counts across political party lines

as illustrated in Figure 10. For Republican Senators, Model 15 classifies 50.7% more toxic tweets than Model 10, whereas Models 14 and 16 show reductions of 24.5% and 5.5%. For Democratic Senators, Model 15 registers a 61.4% increase in toxic tweet counts relative to Model 10, while Models 14 and 16 report decreases of 17.1% and 3.4% respectively. Among the baseline models, Model 16 exhibits the closest alignment with Model 10 in terms of aggregate classification counts. In contrast, other baseline models exhibit considerable misalignment, often skewing their distributions significantly toward either over- or under-counting of toxic content relative to Model 10. This analysis highlights Model 10's superior performance in maintaining both contextual sensitivity and reliability in toxicity detection.

**Application to Twitter Conversation Threads**

I demonstrate the potential usefulness of the PFP through an empirical application of toxicity detection using the dataset described in the Data Preparation section. Specifically, I empirically analyze how Twitter user-base responded to tweets from Senators of the $116^{th}$ U.S. Congress and explore whether escalates within conversation threads and identify factors that may contribute to this phenomenon.

In recent years, the U.S. political landscape has become increasingly polarized, leading to a heightened level of incivility in political discourse, especially in online spaces (Abramowitz and Webster, 2018; Druckman, Peterson and Slothuus, 2013; Iyengar et al., 2019; Iyengar and Westwood, 2015; Rogowski and Sutherland, 2016). Research on partisan polarization suggests that political divisions between Republicans and Democrats have intensified, with individuals becoming more entrenched in their ideological camps (Iyengar and Westwood, 2015). Negative partisanship has fueled hostile rhetoric, particularly against high-profile political figures (Abramowitz and Webster, 2018). When politicians post toxic content, it can provoke out-partisan responses that mirror or escalate such toxicity. Social media platforms further amplify negative content through engagement-driven algorithms, where posts expressing toxicity or hostility are often more widely circulated

(Huszár et al., 2022; Rathje, Van Bavel and van der Linden, 2021). As a result, some scholars have argued that toxic tweets by politicians are more likely to generate toxic responses and be displayed widely, creating a self-reinforcing cycle where toxicity is both rewarded and amplified (Huszár et al., 2022; Munger, 2017; Rathje, Van Bavel and van der Linden, 2021). Therefore, I propose the following hypothesis:

> *Toxic parent tweets are associated with higher levels of toxicity in the following conversation threads.*

To test this theory, each tweet is classified as either containing toxic content (= 1) or not (= 0). Using these classification results, I calculate a *Toxicity* score for each parent tweet, representing the proportion of toxic tweets within the conversation threads following that tweet. The *Toxicity* score ranges from 0 to 1, where 0 indicates the absence of toxic tweets in the threads and 1 indicates that all tweets in the threads are toxic. The main explanatory variable *ParentToxicity* measures whether the parent tweet is toxic or not, where 1 indicates toxicity and 0 otherwise. To test this hypothesis, I additionally control for whether the parent tweet mentions the two presidential candidates (Donald Trump or Joe Biden), specific policy issues: (economy, COVID, abortion, crime or climate), and whether the parent tweet references other Senators from either the same or opposing party. I also control for Senator-specific fixed effects.

To assess the effect of toxic parent tweets on toxicity levels in subsequent conversation threads, I implement a fixed effects model. For each Senator $i$:

$$\begin{aligned} Toxicity_i = \ & \alpha_i + \beta_1 ParentToxicity_i \\ & + \beta_2 MentionTrump_i + \beta_3 MentionBiden_i + \beta_4 MentionEconomy_i \\ & + \beta_5 MentionCovid_i + \beta_6 MentionAbortion_i + \beta_7 MentionCrime_i \\ & + \beta_8 MentionClimate_i + \beta_9 MentionCopartisan_i + \beta_{10} MentionOutpartisan_i \\ & + \epsilon_i \end{aligned} \tag{2}$$

Table 3: Results from Fixed Effects Model with Senator-clustered Standard Errors.

| | *Dependent variable: Toxicity* | | | |
|---|---|---|---|---|
| | Model 10<br>BERT-large-uncased-Twitter<br>(1) | Model 14<br>Dictionary-based<br>(2) | Model 15<br>BOW + Naive Bayes<br>(3) | Model 16<br>fastText + linear classifier<br>(4) |
| ParentToxicity | 0.0767*** | 0.0338*** | 0.3002*** | 0.0058 |
| | (0.0121) | (0.0032) | (0.0419) | (0.0052) |
| Observations | 20,072 | 20,072 | 20,072 | 20,072 |
| $R^2$ | 0.0763 | 0.0531 | 0.1149 | 0.0636 |
| Adjusted $R^2$ | 0.0713 | 0.0479 | 0.1101 | 0.0586 |
| F Statistic (df = 10; 19963) | 164.9973*** | 111.8771*** | 259.1146*** | 135.6425*** |
| *Note:* | | | *p<0.1; **p<0.05; ***p<0.01 | |

Full regression table including other control variables is provided in Appendix G (p. 20). Additionally, parent tweets that mention Trump, the economy, and COVID are associated with increased toxicity in conversation threads, whereas tweets mentioning co-partisan Senators correspond with lower toxicity levels in the conversation threads.

with Senator-specific intercept $\alpha_i$ and error term $\epsilon_i$. The regression analysis was implemented in R using the `plm` package.

The results from Model 10, summarized in Table 3, show a positive and statistically significant coefficient for the main explanatory variable *ParentToxicity* at 0.0767, suggesting that, for a given Senator, posting toxic tweets correlates with increased probability of toxic content for each subsequent tweet in the conversation threads. Importantly, while the direction of this relationship is consistent across models, the magnitude varies substantially. Model 15, for instance, reports a considerably higher coefficient of 0.3002, leading to massive differences in interpretation using BOW. Such high coefficient could be attributed to its systematic over-count of toxicity, as evidenced in the previous section. Conversely, classification results from Model 14 produced a lower coefficient, which may reflect its observed tendency to under-count toxicity levels.

Despite the closest alignment of Model 16 to Model 10 on an aggregate level, Model 16 fails to capture a significant relationship between *ParentToxicity* and toxicity in subsequent threads. Collectively, these findings underscore the robustness and reliability of PFP models in measuring toxicity and capturing meaningful relationships in downstream analysis, particularly for nuanced patterns in toxic language within political discourse.

# Conclusion

This article makes an important contribution to political science by introducing and elucidating the PFP as a robust, accessible approach for text classification. Addressing a gap in the field, I explain the PFP in detail, demystifying how transformer-based language models can be effectively adapted for political science research. With step-by-step practical guidelines, I clarify the processes involved in pre-training and fine-tuning, making these models accessible to scholars who may not have extensive computational resources or technical expertise. By walking through each component of the transformer encoder architecture and detailing how it overcomes the limitations of traditional models, such as BOW or dictionary-based approaches, I provide scholars with a comprehensive yet approachable framework to adopt the PFP for nuanced textual measurement tasks.

I also demonstrate the PFP's impressive effectiveness through an empirical application focused on toxic language classification, which showcases its power in capturing complex, context-dependent meanings often missed by traditional methods. This application highlights how the PFP can accurately interpret subtle cues and dynamic language patterns in social media data. By fine-tuning pre-trained models on a relatively small, annotated dataset, the PFP allows researchers to harness the models' rich linguistic understanding, enhancing accuracy in detecting politically relevant language, such as toxicity in online conversation threads following Senators' tweets. I find that Senators who post toxic content on Twitter encounter higher levels of toxicity in the subsequent conversation threads. This empirical demonstration establishes the PFP as a highly efficient and effective approach for creating context-sensitive text classifiers that respond to real-world research needs in political science.

Looking ahead, there are several promising future directions for the PFP. Firstly, as multilingual models evolve, the PFP approach could be expanded to accommodate political science research in non-English languages or low-resource languages, thereby broadening the scope of comparative studies and cross-cultural research. For instance,

Appendix A (p. 1) shows an application to classify Chinese Weibo posts. Additionally, the PFP framework's flexibility could be extended to address multi-modal data, applying transformer-based models to images, audio, and text in combination, which could yield new insights into multi-modal political messages or campaigns. Finally, as large language models continue to improve, future research could focus on refining the interpretability of these models within the PFP framework. Techniques such as integrated gradients described in details in Appendix F (p. 18-19), which I use to highlight model decision-making, could become more sophisticated, allowing researchers to gain deeper insights into how and why these models interpret complex political language as they do.

Ultimately, by breaking down the complexity of transformer-based models and demonstrating their superior performance in text classification, this paper establishes the PFP as a valuable tool for political scientists for its superior performance, data efficiency, and computational accessibility. As scholars face increasingly large and complex datasets, the PFP offers an efficient and effective way to enhance measurement accuracy and analytical rigor. This article not only lays the foundation for adopting the PFP but also encourages innovative applications that will advance empirical research across diverse domains in political science.

# References


Abramowitz, Alan I. and Steven W. Webster. 2018. "Negative Partisanship: Why Americans Dislike Parties But Behave Like Rabid Partisans." *Political Psychology* 39(S1):119–135.

Agarwal, Pushkal, Oliver Hawkins, Margarita Amaxopoulou, Noel Dempsey, Nishanth Sastry and Edward Wood. 2021. "Hate Speech in Political Discourse: A Case Study of UK MPs on Twitter." *Proceedings of the 32nd ACM Conference on Hypertext and Social Media* pp. 5–16.

Alrababa'h, Ala', William Marble, Salma Mousa and Alexandra A. Siegel. 2021. "Can Exposure to Celebrities Reduce Prejudice? The Effect of Mohamed Salah on Islamophobic Behaviors and Attitudes." *American Political Science Review* 115(4):1111–1128.

Anastasopoulos, L. Jason and Anthony M. Bertelli. 2020. "Understanding Delegation Through Machine Learning: A Method and Application to the European Union." *American Political Science Review* 114(1):291–301.

Barberá, Pablo, Andreu Casas, Jonathan Nagler, Patrick J. Egan, Richard Bonneau, John T. Jost and Joshua A. Tucker. 2019. "Who Leads? Who Follows? Measuring Issue Attention and Agenda Setting by Legislators and the Mass Public Using Social Media Data." *American Political Science Review* 113(4):883–901.

Barbieri, Francesco, Jose Camacho-Collados, Leonardo Neves and Luis Espinosa-Anke. 2020. "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification.".

Beltagy, Iz, Kyle Lo and Arman Cohan. 2019. "SciBERT: A Pretrained Language Model for Scientific Text.".

Beltran, Javier, Aina Gallego, Alba Huidobro, Enrique Romero and Lluís Padró. 2021.

"Male and Female Politicians on Twitter: A Machine Learning Approach." *European Journal of Political Research* 60(1):239–251.

Berliner, Daniel, Benjamin E. Bagozzi, Brian Palmer-Rubin and Aaron Erlich. 2021. "The Political Logic of Government Disclosure: Evidence from Information Requests in Mexico." *The Journal of Politics* 83(1):229–245.

Bestvater, Samuel E. and Burt L. Monroe. 2022. "Sentiment is Not Stance: Target-Aware Opinion Classification for Political Text Analysis." *Political Analysis* pp. 1–22.

Bosley, Mitchell, Saki Kuzushima, Ted Enamorado and Yuki Shiraito. 2022. "Improving Probabilistic Models in Text Classification via Active Learning.".

Bridgman, Aengus, Costin Ciobanu, Aaron Erlich, Danielle Bohonos and Christopher Ross. 2021. "Unveiling: An Unexpected Mid-campaign Court Ruling's Consequences and the Limits of Following the Leader." *The Journal of Politics* 83(3):1024–1029.

Casas, Andreu, Matthew J. Denny and John Wilkerson. 2020. "More Effective Than We Thought: Accounting for Legislative Hitchhikers Reveals a More Inclusive and Productive Lawmaking Process." *American Journal of Political Science* 64(1):5–18.

Caselli, Tommaso, Valerio Basile, Jelena Mitrović and Michael Granitzer. 2021. "HateBERT: Retraining BERT for Abusive Language Detection in English.".

Chalkidis, Ilias, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras and Ion Androutsopoulos. 2020. "LEGAL-BERT: The Muppets Straight Out of Law School.".

Chandrasekharan, Eshwar, Umashanthi Pavalanathan, Anirudh Srinivasan, Adam Glynn, Jacob Eisenstein and Eric Gilbert. 2017. "You Can't Stay Here: The Efficacy of Reddit's 2015 Ban Examined Through Hate Speech." *Proceedings of the ACM on Human-Computer Interaction* 1(CSCW):31:1–31:22.

Chang, Charles and Michael Masterson. 2020. "Using Word Order in Political Text Classification with Long Short-term Memory Models." *Political Analysis* 28(3):395–411.

Cocco, Jessica Di and Bernardo Monechi. 2022. "How Populist are Parties? Measuring Degrees of Populism in Party Manifestos Using Supervised Machine Learning." *Political Analysis* 30(3):311–327.

Collobert, Ronan and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th international conference on Machine learning*. ICML '08 pp. 160–167.

Crabtree, Charles, Matt Golder, Thomas Gschwend and Indrii H. Indriason. 2020. "It Is Not Only What You Say, It Is Also How You Say It: The Strategic Use of Campaign Sentiment." *The Journal of Politics* 82(3):1044–1060.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.". arXiv:1810.04805.

Djourelova, Milena and Ruben Durante. 2022. "Media Attention and Strategic Timing in Politics: Evidence from U.S. Presidential Executive Orders." *American Journal of Political Science* 66(4):813–834.

Druckman, James N., Erik Peterson and Rune Slothuus. 2013. "How Elite Partisan Polarization Affects Public Opinion Formation." *American Political Science Review* 107(1):57–79.

D'Sa, Ashwin Geet, Irina Illina and Dominique Fohr. 2020. BERT and fastText Embeddings for Automatic Detection of Toxic Speech. In *2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)*. pp. 1–5.

Esberg, Jane and Alexandra A. Siegel. 2023. "How Exile Shapes Online Opposition: Evidence from Venezuela." *American Political Science Review* 117(4):1361–1378.

Feierherd, Germán. 2022. "Courting Informal Workers: Exclusion, Forbearance, and the Left." *American Journal of Political Science* 66(2):418–433.

Fortuna, Paula, Juan Soler and Leo Wanner. 2020. Toxic, Hateful, Offensive or Abusive? What Are We Really Classifying? An Empirical Analysis of Hate Speech Datasets. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association pp. 6786–6794.

Fowler, Erika Franklin, Michael M. Franz, Gregory J. Martin, Zachary Peskowitz and Travis N. Ridout. 2021. "Political Advertising Online and Offline." *American Political Science Review* 115(1):130–149.

Goet, Niels D. 2019. "Measuring Polarization with Text Analysis: Evidence from the UK House of Commons, 1811–2015." *Political Analysis* 27(4):518–539.

Gohdes, Anita R. 2020. "Repression Technology: Internet Accessibility and State Violence." *American Journal of Political Science* 64(3):488–503.

Grimmer, Justin and Brandon M. Stewart. 2013. "Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts." *Political Analysis* 21(3):267–297.

Guess, Andrew M. 2021. "(Almost) Everything in Moderation: New Evidence on Americans' Online Media Diets." *American Journal of Political Science* 65(4):1007–1022.

Hager, Anselm and Hanno Hilbig. 2020. "Does Public Opinion Affect Political Speech?" *American Journal of Political Science* 64(4):921–937.

Hartvigsen, Thomas, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray and Ece Kamar. 2022. "ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection.".

Huszár, Ferenc, Sofia Ira Ktena, Conor O'Brien, Luca Belli, Andrew Schlaikjer and Moritz Hardt. 2022. "Algorithmic amplification of politics on Twitter." *Proceedings of the National Academy of Sciences* 119(1):e2025334119.

Iyengar, Shanto and Sean J. Westwood. 2015. "Fear and Loathing across Party Lines: New Evidence on Group Polarization." *American Journal of Political Science* 59(3):690–707.

Iyengar, Shanto, Yphtach Lelkes, Matthew Levendusky, Neil Malhotra and Sean J. Westwood. 2019. "The Origins and Consequences of Affective Polarization in the United States." *Annual Review of Political Science* 22(Volume 22, 2019):129–146.

Jacobs, Alan M., J. Scott Matthews, Timothy Hicks and Eric Merkley. 2021. "Whose News? Class-Biased Economic Reporting in the United States." *American Political Science Review* 115(3):1016–1033.

Jung, Jae-Hee. 2020. "The Mobilizing Effect of Parties' Moral Rhetoric." *American Journal of Political Science* 64(2):341–355.

Karl, Fabian and Ansgar Scherp. 2023. "Transformers are Short Text Classifiers: A Study of Inductive Short Text Classifiers on Benchmarks and Real-world Datasets.". arXiv:2211.16878.

Lajevardi, Nazita. 2021. "The Media Matters: Muslim American Portrayals and the Effects on Mass Attitudes." *The Journal of Politics* 83(3):1060–1079.

Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So and Jaewoo Kang. 2020. "BioBERT: a Pre-trained Biomedical Language Representation Model for Biomedical Text Mining." *Bioinformatics* 36(4):1234–1240.

Lin, Gechun and Christopher Lucas. 2024. An Introduction to Neural Networks for the Social Sciences. In *Oxford Handbook of Engaged Methodological Pluralism in Political Science (Vol 1)*, ed. Janet M. Box-Steffensmeier, Dino P. Christenson and Valeria Sinclair-Chapman. Oxford University Press.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer and Veselin Stoyanov. 2019. "RoBERTa: A Robustly Optimized BERT Pretraining Approach.". arXiv:1907.11692.

Lucas, Christopher, Richard A. Nielsen, Margaret E. Roberts, Brandon M. Stewart, Alex Storer and Dustin Tingley. 2015. "Computer-Assisted Text Analysis for Comparative Politics." *Political Analysis* 23(2):254–277.

Luis von Ahn's, Research Group. 2022. "Useful Resources: Offensive/Profane Word List.". **URL:** *https://www.cs.cmu.edu/ biglou/resources/bad-words.txt*

Magaloni, Beatriz, Edgar Franco-Vivanco and Vanessa Melo. 2020. "Killing in the Slums: Social Order, Criminal Governance, and Police Violence in Rio de Janeiro." *American Political Science Review* 114(2):552–572.

Magaloni, Beatriz and Luis Rodriguez. 2020. "Institutionalized Police Brutality: Torture, the Militarization of Security, and the Reform of Inquisitorial Criminal Justice in Mexico." *American Political Science Review* 114(4):1013–1034.

Manekin, Devorah and Tamar Mitts. 2022. "Effective for Whom? Ethnic Identity and Nonviolent Resistance." *American Political Science Review* 116(1):161–180.

Massaro, Toni M. and Robin Stryker. 2012. "Freedom of Speech, Liberal Democracy, and Emerging Evidence on Civility and Effective Democratic Engagement Symposium: Political Discourse, Civility, and Harm." *Arizona Law Review* 54(2):375–442.

Matamoros-Fernández, Ariadna and Johan Farkas. 2021. "Racism, Hate Speech, and Social Media: A Systematic Review and Critique." *Television & New Media* 22(2):205–224.

Merchant, Amil, Elahe Rahimtoroghi, Ellie Pavlick and Ian Tenney. 2020. "What Happens To BERT Embeddings During Fine-tuning?". arXiv:2004.14448.

Mikolov, Tomas, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space.". arXiv:1301.3781.

Motolinia, Lucia. 2021. "Electoral Accountability and Particularistic Legislation: Evidence from an Electoral Reform in Mexico." *American Political Science Review* 115(1):97–113.

Munger, Kevin. 2017. "Tweetment Effects on the Tweeted: Experimentally Reducing Racist Harassment." *Political Behavior* 39(3):629–649.

Nguyen, Dat Quoc, Thanh Vu and Anh Tuan Nguyen. 2020. "BERTweet: A Pre-trained Language Model for English Tweets.". arXiv:2005.10200.

Nielsen, Richard A. 2020. "Women's Authority in Patriarchal Social Movements: The Case of Female Salafi Preachers." *American Journal of Political Science* 64(1):52–66.

Osmundsen, Mathias, Alexander Bor, Peter Bjerregaard Vahlstrup, Anja Bechmann and Michael Bang Petersen. 2021. "Partisan Polarization Is the Primary Psychological Motivation behind Political Fake News Sharing on Twitter." *American Political Science Review* 115(3):999–1015.

Osnabrügge, Moritz, Sara B. Hobolt and Toni Rodon. 2021. "Playing to the Gallery: Emotive Rhetoric in Parliaments." *American Political Science Review* 115(3):885–899.

Parekh, Bhikhu. 2012. Is There a Case for Banning Hate Speech? In *The Content and Context of Hate Speech: Rethinking Regulation and Responses*, ed. Michael Herz and Peter Molnar. Cambridge: Cambridge University Press pp. 37–56.

Park, Baekkwan, Kevin Greene and Michael Colaresi. 2020. "Human Rights are (Increasingly) Plural: Learning the Changing Taxonomy of Human Rights from Large-scale Text Reveals Information Effects." *American Political Science Review* 114(3):888–910.

Park, Hyunji Hayley, Yogarshi Vyas and Kashif Shah. 2022. "Efficient Classification of Long Documents Using Transformers.". arXiv:2203.11258.

Park, Ju Yeon and Jacob Montgomery. 2023. "From Text to Measure: Creating Trustworthy Measures Using Supervised Machine Learning.". Unpublished manuscript.

Poletto, Fabio, Valerio Basile, Manuela Sanguinetti, Cristina Bosco and Viviana Patti. 2021. "Resources and Benchmark Corpora for Hate Speech Detection: A Systematic Review." *Language Resources and Evaluation* 55(2):477–523.

Qu, Chen, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang and Mohit Iyyer. 2019. "BERT with History Answer Embedding for Conversational Question Answering." *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval* pp. 1133–1136.

Rathje, Steve, Jay J. Van Bavel and Sander van der Linden. 2021. "Out-group Animosity Drives Engagement on Social Media." *Proceedings of the National Academy of Sciences* 118(26):e2024292118.

Rheault, Ludovic, Erica Rayment and Andreea Musulan. 2019. "Politicians in the Line of Fire: Incivility and the Treatment of Women on Social Media." *Research & Politics* 6(1).

Roberts, Margaret E. 2016. "Introduction to the Virtual Issue: Recent Innovations in Text Analysis for Social Science." *Political Analysis* 24(V10):1–5.

Roberts, Margaret E., Brandon M. Stewart and Richard A. Nielsen. 2020. "Adjusting for Confounding with Text Matching." *American Journal of Political Science* 64(4):887–903.

Rogowski, Jon C. and Joseph L. Sutherland. 2016. "How Ideology Fuels Affective Polarization." *Political Behavior* 38(2):485–508.

Rossiter, Erin L. 2022. "Measuring Agenda Setting in Interactive Political Communication." *American Journal of Political Science* 66(2):337–351.

Ruder, Sebastian, Matthew E. Peters, Swabha Swayamdipta and Thomas Wolf. 2019. "Transfer Learning in Natural Language Processing." *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials* pp. 15–18.

Saraceno, Joseph. 2020. "Disparities in a Flagship Political Science Journal? Analyzing Publication Patterns in the Journal of Politics, 1939–2019." *The Journal of Politics* 82(4):e45–e55.

Schub, Robert. 2022. "Informing the Leader: Bureaucracies and International Crises." *American Political Science Review* 116(4):1460–1476.

Sellars, Andrew. 2016. "Defining Hate Speech.". Berkman Klein Center Research Publication No. 2016-20.

Shawky, Peter E., Saleh Mesbah ElKaffas and Shawkat K. Guirguis. 2024. "Effect of Typos on Text Classification Accuracy in Word and Character Tokenization." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 40(2):152–162.

Siegel, Alexandra A., Evgenii Nikitin, Pablo Barberá, Joanna Sterling, Bethany Pullen, Richard Bonneau, Jonathan Nagler and Joshua A. Tucker. 2021. "Trumping Hate on Twitter? Online Hate Speech in the 2016 U.S. Election Campaign and its Aftermath." *Quarterly Journal of Political Science* 16(1):71–104.

Siegel, Alexandra A. and Vivienne Badaan. 2020. "#No2Sectarianism: Experimental Approaches to Reducing Sectarian Hate Speech Online." *American Political Science Review* 114(3):837–855.

Stier, Sebastian, Frank Mangold, Michael Scharkow and Johannes Breuer. 2022. "Post Post-Broadcast Democracy? News Exposure in the Age of Online Intermediaries." *American Political Science Review* 116(2):768–774.

Sundararajan, Mukund, Ankur Taly and Qiqi Yan. 2017. "Axiomatic Attribution for Deep Networks.". arXiv:1703.01365.

Tamara, Shepherd, Alison Harvey, Tim Jordon, Sam Srauy and Kate Miltner. N.d. "Histories of Hating - Tamara Shepherd, Alison Harvey, Tim Jordan, Sam Srauy, Kate Miltner, 2015.".

Theocharis, Yannis, Pablo Barberá, Zoltán Fazekas and Sebastian Adrian Popa. 2020. "The Dynamics of Political Incivility on Twitter." *SAGE Open* 10(2):2158244020919447.

Todd, Jason Douglas, Edmund J. Malesky, Anh Tran and Quoc Anh Le. 2021. "Testing Legislator Responsiveness to Citizens and Firms in Single-Party Regimes: A Field Experiment in the Vietnamese National Assembly." *The Journal of Politics* 83(4):1573–1588.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*. Vol. 30 Curran Associates, Inc.

Vries, Erik de, Martijn Schoonvelde and Gijs Schumacher. 2018. "No Longer Lost in Translation: Evidence that Google Translate Works for Comparative Bag-of-Words Text Applications." *Political Analysis* 26(4):417–430.

Wahman, Michael, Nikolaos Frantzeskakis and Tevfik Murat Yildirim. 2021. "From Thin to Thick Representation: How a Female President Shapes Female Parliamentary Behavior." *American Political Science Review* 115(2):360–378.

Wang, Yu. 2023*a*. "On Finetuning Large Language Models." *Political Analysis* pp. 1–5.

Wang, Yu. 2023*b*. "Topic Classification for Political Texts with Pretrained Language Models." *Political Analysis* 31(4):662–668.

Wasow, Omar. 2020. "Agenda Seeding: How 1960s Black Protests Moved Elites, Public Opinion and Voting." *American Political Science Review* 114(3):638–659.

Widmann, Tobias and Maximilian Wich. 2022. "Creating and Comparing Dictionary, Word Embedding, and Transformer-Based Models to Measure Discrete Emotions in German Political Text." *Political Analysis* pp. 1–16.

Yang, Jiacheng, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Weinan Zhang, Yong Yu and Lei Li. 2020. "Towards Making the Most of BERT in Neural Machine Translation." *Proceedings of the AAAI Conference on Artificial Intelligence* 34(05).

Yoder, Jesse. 2020. "Does Property Ownership Lead to Participation in Local Politics? Evidence from Property Records and Meeting Minutes." *American Political Science Review* 114(4):1213–1229.

Zhuang, Fuzhen, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong and Qing He. 2021. "A Comprehensive Survey on Transfer Learning." *Proceedings of the IEEE* 109(1):43–76.

Zubek, Radoslaw, Abhishek Dasgupta and David Doyle. 2021. "Measuring the Significance of Policy Outputs with Positive Unlabeled Learning." *American Political Science Review* 115(1):339–346.

# Appendices

# Appendix A: Application to Chinese Weibo Posts

To illustrate the potential of the PFP in non-English contexts, I replicated a study by Chang and Masterson (2020), who used an Long Short-term Memory along with word2vec embeddings to classify over 10,000 Chinese Weibo posts as political versus non-political categories. I employed *BERT-base-Chinese* as the pre-trained language model and fine-tuned it for the binary classification task using the original annotated dataset released by the authors, where each post was labeled as political or non-political.

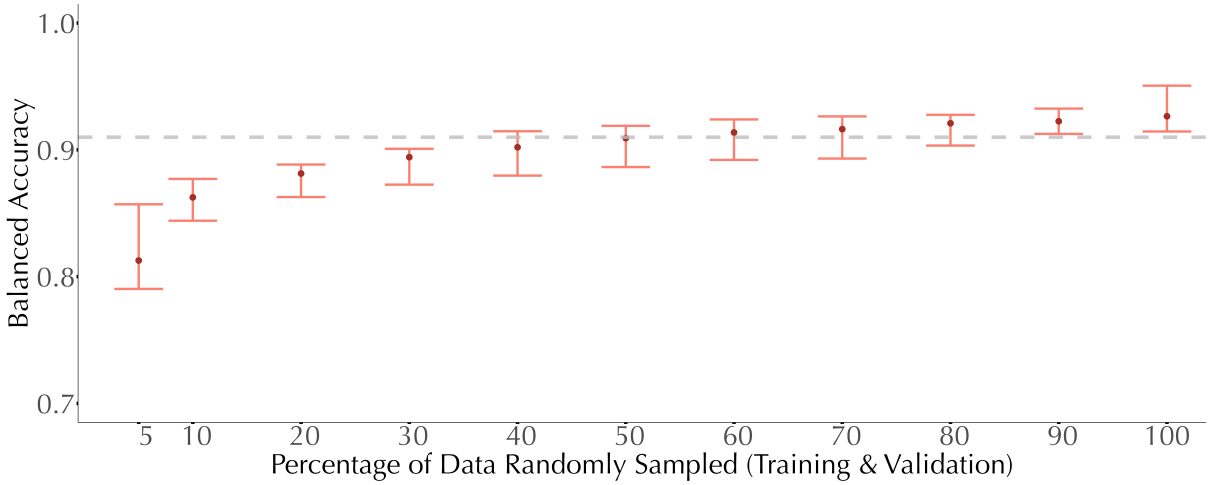



Figure 1: Replication Results from Chang and Masterson 2020

In Figure 1, the confidence intervals show the maximum and minimum balanced accuracy scores of five independent experiments. The dots represent the average balanced accuracy scores of the five experiments. The $X$-axis represents the percentage of data randomly sampled from the training and validation datasets. The gray dashed horizontal line marks the performance of the model proposed by the original paper using 100% of the dataset. To make the comparison fair, the size of the test dataset is maintained to be 10% of the entire dataset for all experiments following the approach of the original paper. Using only 50% of the data, the PFP model achieves higher balanced accuracy on average compared to the original approach. Remarkably, with just 5% of the training and validation data, the average balanced accuracy score of the fine-tuned model is already over 0.8, indicating the data efficiency of the fine-tuning process.

# Appendix B: Definitions of Toxicity

Measuring toxicity in large datasets remains challenging due to a lack of an unambiguous and agreed-upon definition. Scholars often do not clearly distinguish toxicity from incivility, hate speech, offensive speech, or abusive speech. There are two primary tendencies in the literature when defining toxicity. At one end of the spectrum are narrow and application-specific definitions, which often restrict toxicity to explicitly violence-inducing speech or derogatory language specifically targeting a group with common characteristics (Munger, 2017; Siegel et al., 2021). At the other end of the spectrum are broad and comprehensive definitions designed to identify all types of abusive content (Hartvigsen et al., 2022; Theocharis et al., 2020). To make the concept more feasible for human annotation, I followed the later approach and adopted a comprehensive and operationalizable definition of toxicity offered by Poletto et al. (2021), where toxicity is defined as *"any impolite, rude or hurtful language that can show a debasement of someone or something, or show intense emotion, including hate speech, derogatory language and also profanity"* (Poletto et al., 2021). Based on this definition, hate speech, incivility, and hostile language are all proper subsets of toxic language, as shown in the Venn diagram in Figure 2.

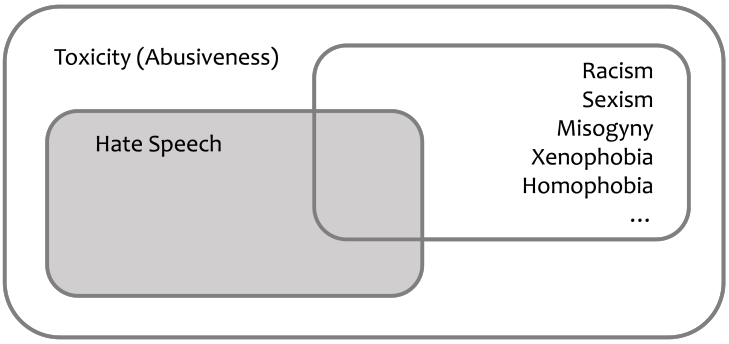



Figure 2: Toxic Language Definition. The figure is based on the definitions offered by Poletto et al. (2021). Toxicity is the most comprehensive category, which encompasses all different types of related concepts discussed in the literature.

# Appendix C: Data Annotation

I randomly selected 3000 tweets (approximately 0.2%) from the entire dataset of around 1.65 million tweets to be annotated by human coders hired through Amazon Mechanical Turk. The tweets were split into 30 batches of 100 tweets each. Within each batch, there were five gold-standard unambiguous tweets that were implemented as attention checks to filter bots and irresponsible annotators. The annotators must go through a training module where I provided definitions and examples of toxic and non-toxic tweets. The annotators must correctly code at least 10 out of 12 test tweets to be qualified for the actual annotation tasks. Each tweet is cross-coded by two coders. After the first round of annotation, I filtered the batches with cross-coder agreement levels lower than 85% and re-published them with the same instructions to get them re-coded. After the second round of annotation, the overall agreement level across batches increased to 93%, which resulted in 2790 cross-coded and agreed tweets to be used for the fine-tuning process. I split them randomly into training (80%), validation (10%), and test (10%) datasets. The training dataset is used to fine-tune the language model for classification, and the validation dataset is used to evaluate the model during the training process. The test dataset is then used to evaluate the model performance after the training process is fully complete. Although the fine-tuning process of the pre-trained language models typically requires a relatively smaller dataset than pre-training to achieve good performance, more annotated data is still desired in the future for validation purposes. Each tweet is coded as either containing toxic language or not containing any toxic language (i.e., 1 or 0). The annotated dataset with 2790 crosscoded and agreed tweets has 1641 (58.8%) nontoxic tweets and 1149 (41.2%) toxic tweets. Since toxic tweets may not have slurs and toxic keywords do not necessarily make the tweets toxic, the annotation codebook considers both the context and target of a tweet.

Below is a condensed version of the training module used for Amazon Mechanical Turk. The human annotators must go through the training program and score at least

ten points on the test to be able to qualify for the later classification tasks. The purpose of the implemented training module prior to the actual annotation task is to reduce the human error rate in the labeled dataset. Manual data annotations may inevitably contain human errors to some extent, which in the end would propagate to the training process of the classifier during the fine-tuning stage.

**Completing this training module qualifies you to complete later HITs**

**WARNING: This task may contain offensive and abusive content.**

**Introduction:**

If you finish this training module with a passing score, you will be qualified to complete HITs posted with the title Toxic Language Detection in Tweets.

We are researching toxic content on Twitter. Your job will be to look at a tweet and determine whether it contains toxic content as we define it below. To do this accurately, you will need to carefully read the definition of toxicity we provide. We will also provide you with some examples to help you understand how to apply this definition accurately. Each task will be structured the same. You will read a tweet and indicate it toxic or not toxic.

**Instructions:**

You should code a tweet as toxic if the tweet is: 1. strongly impolite, 2. rude, or 3. contains hurtful language.

Toxic langauge will often include attacking the character, dignity, or importance of someone, something, or some groups of people (e.g., "you are a liar" or "Democrats are all sleazy"). It is also common to see it associated with intense (usually negative) emotions (e.g. "I fucking hate these people!").

Toxic language includes a wide spectrum of speech ranging from derogatory languguage and profanity (e.g., "what an idiot") to hate speach (e.g., "Serve your country, burn down a mosque"). Examples include racism, misogny, and homophobia but toxic speech can also be personal, or attacks based on political affiliation, religion, and more. Even tweets

without any of these traits can be toxic if the message is conveyed in a strongly impolite, rude, or hurtful manner.

You will encounter some cases where the coding decision will not be clear. In those cases, use your best judgment. When in doubt, consider whether the tweet would seem impolite, rude, or hurtful if it was directed at you. If you think a tweet is just slightly rude or impolite, code it as toxic. If you think a tweet would be rude in-person during a conversation with someone you just met, code it as toxic. If you cannot confidently code a tweet as not toxic, it is usually toxic. When in doubt, code it toxic. For long tweets with multiple sentences, if one sentence is toxic and the rest are supporting it, the tweet is toxic.

**Additional Tips:**

1. The tweets we are studying are real tweets collected from the conversation threads following tweets from U.S. Senators. Politically relevant toxic language is often present in these tweets. Specific examples include asking someone to get out of office, claiming someone has done no work, advocating for someone to lose in elections, calling someone a liar, claiming someone is ignorant of their job, saying someone has failed the nation, etc.

2. When toxic hashtags are present, and the tweet is supporting the hashtag or using the hashtag as a reference, the tweet is considered as toxic. Examples include #CrookedJoeBiden, #RadicalLeft, #SuperSpreader Trump, etc.

3. When emojis are used in a toxic way or as a toxic reference, the tweet is considered as toxic. For example, the middle-finger emoji, the vomiting face, can be toxic in certain contexts.

4. When the @USER appears in the tweet, it means the tweet is either replying to, commenting on, or mentioning the user. Toxicity can often be directed towards the @USER.

5. URLs might appear in the tweets, you can simply ignore them for this coding task.

6. Typos may appear in tweets. If you encounter them, try your best to infer the meaning.

**Attention:**

For each HIT, your task is to read each tweet and classify whether it is toxic or not toxic. Be aware that these are all replies to tweets, and sometimes the context is missing. Please try your best to infer the context then do the classification. Skimming or reading quickly will result in low-quality evaluations, and you may not be invited to participate in our future HITS.

This training module has two parts. In Part 1, we will provide 10 practice HITs followed by instructions about how the tweet should be coded. In Part 2, we will give you 12 test HITs to complete. To receive the qualification for future HITs, you must complete 10 out of 12 of these test HITs correctly.

In the module below, we have overemphasized toxic tweets for training purposes. In the real HITs that you will be coding, toxic tweets are much rarer.

**Part 1:**

The followings are practice HITs. Your answer will not be scored or count for or against your qualification. We will provide you with the correct answers.

**Example 1:**

@JustinTSheehy@ChrisMurphyCT Video proof of Donald trump comdeming white supremacy and racism from 2016 on... Proof that Biden and harris LIED to your faces right on that stage so what else can they lie about!! But you leftist snowflakes dont care because your brainwashed, no brain, non thinkers https://t.co/ztZnNgbTtl

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.** The tweet contains strongly impolite and rude language directed at Biden, Harris and people supporting left-leaning political ideologies. Keywords include "LIED," "leftist snowflakes," "brainwahed, no brain, non thinkers".

**Example 2:**

@Sen ThomTillis @senjudiciary I look forward to you being voted out of office and then try to obtain healthcare. #Disgraceful #notforamericans

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.** The tweet contains strongly impolite, rude and hurtful language directed towards the @USERs. It also uses toxic hashtags. Keywords include "voted out of office," "#Disgraceful," "#notforamericans."

**Example 3:**

@senrobportman Fuck you, Rob. Approve the goddamned plan you've had since May, you rich prick. #3MillionDollarsFromNRAtoPortman

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.** The tweet contains profanity and rude language directed at "Rob." Keywords include "Fuck you" and "rich prick."

**Example 4:**

@hova 1066 @amyklobuchar @deanbphillips What in the world does this have to do with the fact that you are a woman?

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Not Toxic.** The tweet does not contain strongly impolite, rude, or hurtful language.

**Example 5:**

@SenWarren Talk about a scam!!! Look in the mirror

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.** Althought the tweet does not contain explicitly rude language, the implied meaning is that @USER is a scam. The meaning conveyed is strongly impolite, rude, and hurtful, so the tweet is toxic.

**Example 6:**

@misiaszek42 @ashlynoriginal @SenWarren Half of all fertilized eggs don't implant and are lost. By your logic, since life begins at conception, then almost everyone who has had regular unprotected sex is a murderer (including most married couples).

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Not Toxic.** The tweet is simply presenting an argument and does not contain strongly impolite, rude, or hurtful language.

**Example 7:**

@lisamurkowski Vote Out @lisamurkowski in two years. She's destroying Alaska for Trump!

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.** The tweet contains rude language directed at @USER. Keywords include "Vote Out," "destroying Alaska for Trump."

**Example 8:**

@JohnCornyn Looks like you have only two speeds: enabler and obstructionist.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.** Although the tweet does not contain explicitly rude language, the words "obstructionis" and "enabler" are used with a derogatory connotation. The meaning conveyed is strongly impolite, rude, and hurtful, so the tweet is toxic.

**Example 9:**

@KamalaHarris @JoeBiden Promises as empty as you and Joe.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.** The tweet contains rude and hurtful language directed at @USERs. Keywords include "promises as empty as you and Joe."

**Example 10:**

@drewwilder_1@thealmightymeh@LegendaryScoop @SenWarren It's an embryo. It can't feel, it has no awareness, it does not suffer. The pendant person does. Stop applying your emotions about babies to embryos.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Not Toxic.** The tweet does not contain strongly impolite, rude, or hurtful language.

**Part 2:**

The next 12 questions are your test HITs. Please read the tweet carefully and label whether it contains toxicity. Your answers will be scored. You must answer at least 10 of

the test HITs correctly to receive the qualification.

**Test 1:**

@maziehirono As usual, you made a fool out of yourself at the hearings yesterday, that question should have been addressed to Biden...

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.** (Answers of test tweets are not shown in real training module.)

**Test 2:**

@scottnr0331 @CatPoacher @SenWarren You do realize you are not the only person in this conversation who served this country, don't you? I gave 32 years to its service and I have more than earned the right to hold both you and Trump in the contempt you have richly earned. I used to be a Republican too. No more.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.**

**Test 3:**

@sendavidperdue I'm a Jew! Know why 87% of Jews vote Dem? Because we've seen it before. Hitler did what Trump and Perdue are doing. Suppress the media. Suppress science. Lock up opposition. Call elections rigged. Is this the man to help minorities??!! Vote DEMOCRAT ACROSS THE BOARD

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.**

**Test 4:**

@SenatorCollins Yippee, we get 6 more years of you being McConnell's lapdog. Your lack of integrity and political courage are a burden on Maine.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.**

**Test 5:**

@drewwilder_1@thealmightymeh @LegendaryScoop @SenWarren So go ahead. Tell me I should have left my two small children without a mother. Tell me I should have suffered

and died for the sake of an embryo that was never in any pain. Tell me that choosing not to risk my life is a choice I shouldn't get to make for myself.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Not Toxic.**

**Test 6:**

@SenJohnKennedy I am Thankful for Sen. Kennedy. The best Kennedy that ever served our nation.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Not Toxic.**

**Test 7:**

@marcorubio This is really something. What about the Hero's Act?? You can't be this dumb

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.**

**Test 8:**

@SenatorBraun @SecAzar 9 of 51 (R)Senators called for ́transition̈to Pres Elect Biden. Your not one of them, you are failing to do the job you swore to do. Step up, call for the transition & also for the Cares Act to help citizens you represent. It is ur job! How long ru going hold out?? Do the job

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.**

**Test 9:**

@JohnCornyn @politico It's not her religion, John. It's the offshoot cult she belongs to.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.**

**Test 10:**

@LindseyGrahamSC @seanhannity @senjudiciary Learn from Obama. Do the transi-

tion. Don't be naive... Enough is enough dude. U lose. Period. Priority save American life not your seat. Republican.. All your integrity on table. Don't disgrace your family name and your children will be shameful in future. Wake up. Jesus christ

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.**

**Test 11:**

@SenDuckworth You are an inspiration to millions of Americans who thank you for your service and sacrifice.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Not Toxic.**

**Test 12:**

@SenJeffMerkley I thought you told us the current administration didn't have a vaccine distribution plan? Do you ever recant your hyper-political BS? It's not helpful.

Is the tweet Toxic or Not? ☐ Toxic ☐ Not Toxic

**Answer: Toxic.**

**Before you submit your answers:**

You will only have 1 chance to take this test. Make sure that you are satisfied with all of your answers above before submitting.

If you become qualified to participate in the HITs, please continue to fully read each future HIT and provide your best guess of the correct answer. Your performance will be monitored as you complete more HITs. If you provide poor quality answers, you may be blocked from continued participation in this study (and future studies).

# Appendix D: Simple Code Example for Using PFP

## D.1. Off-the-shelf Adoption

Below is a code example to apply off-the-shelf models directly to our data stored in *test.csv* containing three columns: *id*, *text*, and *label*.

```
from transformers import pipeline
import pandas as pd
test_df = pd.read_csv("test.csv")
classifier = pipeline(model="path_to_your_model_or_HuggingFace_model",
            device = 'cuda')
predicted_labels = [classifier(x)[0] for x in test_df['text']]
```

The variable `predicted_labels` contains the predicted labels for all text inputs along with the probabilities. For reproducibility, all computation in this article is conducted on Google Colab on an A100 GPU. All PFP models used in this article can be found on HuggingFace Hub.[39] Python code are provided in the replication archive.

## D.2. Task-specific Supervised Training

Below is a code example to freeze the bert encoders and only update the task-specific classification layer as shown in Figure 6.

```
for name, param in model.named_parameters():
    if 'classifier' not in name: # classifier layer
        param.requires_grad = False
```

[39] https://huggingface.co/AnonymousCS

## D.3. Fine-tuning with Task-specific Supervised Training

Below is a code example for fine-tuning language models using *bert-base-uncased* as an example.[40] The full executable python script is provided in the replication archive.

```
model = AutoModelForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels = 2) # increase for multi-class tasks
training_args = TrainingArguments(
    output_dir="toxicity_Twitter",
    learning_rate=2e-5,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    num_train_epochs=4,
    weight_decay=0.01,
    logging_steps=100,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    push_to_hub=True,
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_data["train"],
    eval_dataset=tokenized_data["val"],
```

[40] I used the same set of hyperparameters for all models trained in this article for demonstration purposes. For more experienced scholars, hyperparameter tuning is highly encouraged to achieve optimal performance of models.

```
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)
trainer.train()
trainer.push_to_hub() # to push fine-tuned model to HuggingFace Hub
```

## D.4. Unsupervised Refining and Fine-tuning with Task-specific Supervised Training

In this application, I refined the *BERT-base-uncased* model by re-training the model with my Twitter dataset of over 1.65 million tweets with a masked token prediction task. The training process took 46 hours on Google Colab on an A100 GPU. The refined model was then fine-tuned following the same steps as other models in Section 4.4. Below is a code example to execute *run_mlm.py* in Google Colab, which contains the code for refining *bert-base-uncased* model with a masked token prediction task. *run_mlm.py* is provided in the replication archive.

```
!python run_mlm.py \
    --model_name_or_path google-bert/bert-base-uncased \
    --train_file training_text.txt \
    --validation_file test_text.txt \
    --per_device_train_batch_size 32 \
    --per_device_eval_batch_size 32 \
    --do_train \
    --do_eval \
    --output_dir ./twitter_hatebert \
    --pad_to_max_length \
    --line_by_line
```

# Appendix E: Model Evaluation Results

To compare PFP models with traditional models, I used Google's Perspective API, fastText, BOW, and a dictionary-based approach[41] as the baseline models. Google's Perspective API is a close-sourced API that detects toxicity online, where the model was also fine-tuned using a crowd-sourced labeled dataset (Fortuna, Soler and Wanner, 2020). The exact training process is not transparent from its online documentation. I also performed text classification using the open-source library fastText, which allows researchers to train their own static embeddings using custom dataset and then fastText uses a linear classifier to obtain the final classification results. I also use a BOW text representation model with a naive Bayes classifier. For classification with the dictionary, I use a naive approach: if an input contains any words or phrases from the dictionary, it is considered toxic. Table 1, Table 2, and Table 3 show the out-of-sample performance metrics for the baseline models, off-the-shelf models, and PFP fine-tuned models respectively.

[41]I adopted a dictionary of abusive words from Luis von Ahn's Research Group (Luis von Ahn's, 2022).

Table 1: Out-of-Sample Evaluation Results on Twitter Politician Threads Test Data for Off-the-shelf Models

| Model | | Precision | Recall | F1-Score | Accuracy | Balanced Accuracy |
|---|---|---|---|---|---|---|
| **HateBERT-abuseval** | Label-0 | 0.7421 | 0.9939 | 0.8497 | 0.7921 | 0.7470 |
| Model 1 | Label-1 | 0.9828 | 0.5000 | 0.6628 | | |
| | Macro Avg | 0.8624 | 0.7470 | 0.7563 | | |
| | Weighted Avg | 0.8404 | 0.7921 | 0.7734 | | |
| **HateBERT-offenseval** | Label-0 | 0.7734 | 0.9515 | 0.8533 | 0.8065 | 0.7740 |
| Model 2 | Label-1 | 0.8947 | 0.5965 | 0.7158 | | |
| | Macro Avg | 0.8341 | 0.7740 | 0.7845 | | |
| | Weighted Avg | 0.8230 | 0.8065 | 0.7971 | | |
| **HateBERT-hateval** | Label-0 | 0.6109 | 0.9515 | 0.7441 | 0.6129 | 0.5372 |
| Model 3 | Label-1 | 0.6364 | 0.1228 | 0.2059 | | |
| | Macro Avg | 0.6236 | 0.5372 | 0.4750 | | |
| | Weighted Avg | 0.6213 | 0.6129 | 0.5242 | | |
| **BERT-abuseval** | Label-0 | 0.7500 | 1.0000 | 0.8571 | 0.8029 | 0.7588 |
| Model 4 | Label-1 | 1.0000 | 0.5175 | 0.6821 | | |
| | Macro Avg | 0.8750 | 0.7588 | 0.7696 | | |
| | Weighted Avg | 0.8522 | 0.8029 | 0.7856 | | |
| **BERT-offenseval** | Label-0 | 0.7861 | 0.9576 | 0.8634 | **0.8208** | **0.7902** |
| Model 5 | Label-1 | 0.9103 | 0.6228 | 0.7396 | | |
| | Macro Avg | 0.8482 | 0.7902 | 0.8015 | | |
| | Weighted Avg | 0.8368 | 0.8208 | 0.8128 | | |
| **BERT-hateval** | Label-0 | 0.6148 | 0.9091 | 0.7335 | 0.6093 | 0.5423 |
| Model 6 | Label-1 | 0.5714 | 0.1754 | 0.2685 | | |
| | Macro Avg | 0.5931 | 0.5423 | 0.5010 | | |
| | Weighted Avg | 0.5971 | 0.6093 | 0.5435 | | |

Table 2: Out-of-Sample Evaluation Results on Twitter Politician Threads Test Data for Fine-tuned Models

| Model | | Precision | Recall | F1-Score | Accuracy | Balanced Accuracy |
|---|---|---|---|---|---|---|
| **HateBERT-Twitter** | Label-0 | 0.8629 | 0.9152 | 0.8882 | 0.8638 | 0.8523 |
| Model 7 | Label-1 | 0.8654 | 0.7895 | 0.8257 | | |
| Training Time in min:sec | Macro Avg | 0.8641 | 0.8523 | 0.8570 | | |
| 01:14 | Weighted Avg | 0.8639 | 0.8638 | 0.8627 | | |
| **BERT-base-uncased-Twitter** | Label-0 | 0.8659 | 0.9394 | 0.9012 | 0.8781 | 0.8644 |
| Model 8 | Label-1 | 0.9000 | 0.7895 | 0.8411 | | |
| 01:14 | Macro Avg | 0.8830 | 0.8644 | 0.8711 | | |
| | Weighted Avg | 0.8798 | 0.8781 | 0.8766 | | |
| **BERT-base-cased-Twitter** | Label-0 | 0.8644 | 0.9273 | 0.8947 | 0.8710 | 0.8584 |
| Model 9 | Label-1 | 0.8824 | 0.7895 | 0.8333 | | |
| 01:21 | Macro Avg | 0.8734 | 0.8584 | 0.8640 | | |
| | Weighted Avg | 0.8717 | 0.8710 | 0.8696 | | |
| **BERT-large-uncased-Twitter** | Label-0 | 0.8994 | 0.9212 | 0.9102 | **0.8925** | **0.8860** |
| Model 10 | Label-1 | 0.8818 | 0.8509 | 0.8661 | | |
| 03:30 | Macro Avg | 0.8906 | 0.8860 | 0.8881 | | |
| | Weighted Avg | 0.8922 | 0.8925 | 0.8922 | | |
| **BERT-large-cased-Twitter** | Label-0 | 0.8810 | 0.8970 | 0.8889 | 0.8674 | 0.8608 |
| Model 11 | Label-1 | 0.8468 | 0.8246 | 0.8356 | | |
| 03:46 | Macro Avg | 0.8639 | 0.8608 | 0.8622 | | |
| | Weighted Avg | 0.8670 | 0.8674 | 0.8671 | | |
| **freeze-BERT-base-uncased-Twitter** | Label-0 | 0.5914 | 1.0000 | 0.7432 | 0.5914 | 0.5000 |
| Model 12 | Label-1 | 0.0000 | 0.0000 | 0.0000 | | |
| 00:32 | Macro Avg | 0.2957 | 0.5000 | 0.3716 | | |
| | Weighted Avg | 0.3498 | 0.5914 | 0.4396 | | |
| **refined-BERT-base-uncased-Twitter** | Label-0 | 0.8571 | 0.9455 | 0.8991 | 0.8746 | 0.8587 |
| Model 13 | Label-1 | 0.9072 | 0.7719 | 0.8341 | | |
| 01:19 | Macro Avg | 0.8822 | 0.8587 | 0.8666 | | |
| | Weighted Avg | 0.8776 | 0.8746 | 0.8726 | | |

Table 3: Out-of-Sample Evaluation Results on Twitter Politician Threads Dataset for Baseline Models

| Model | | Precision | Recall | F1-Score | Accuracy | Balanced Accuracy |
|---|---|---|---|---|---|---|
| **Dictionary-based** | Label-0 | 0.7083 | 0.3091 | 0.4304 | 0.5161 | 0.5624 |
| Model 14 | Label-1 | 0.4493 | 0.8158 | 0.5794 | | |
| | Macro Avg | 0.5788 | 0.5624 | 0.5049 | | |
| | Weighted Avg | 0.6025 | 0.5161 | 0.4913 | | |
| **BOW + Naive Bayes** | Label-0 | 0.6800 | 0.3091 | 0.4250 | 0.5054 | 0.5493 |
| Model 15 | Label-1 | 0.4412 | 0.7895 | 0.5660 | | |
| | Macro Avg | 0.5606 | 0.5493 | 0.4955 | | |
| | Weighted Avg | 0.5824 | 0.5054 | 0.4826 | | |
| **fastText + linear classifier** | Label-0 | 0.7473 | 0.8424 | 0.7920 | **0.7384** | **0.7151** |
| Model 16 | Label-1 | 0.7204 | 0.5877 | 0.6473 | | |
| | Macro Avg | 0.7339 | 0.7151 | 0.7197 | | |
| | Weighted Avg | 0.7363 | 0.7384 | 0.7329 | | |
| **Google's Perspective API** | Label-0 | 0.6749 | 0.9939 | 0.8039 | 0.7133 | 0.6505 |
| Model 17 | Label-1 | 0.9722 | 0.3070 | 0.4667 | | |
| | Macro Avg | 0.8236 | 0.6505 | 0.6353 | | |
| | Weighted Avg | 0.7964 | 0.7133 | 0.6661 | | |

# Appendix F: Integrated Gradients

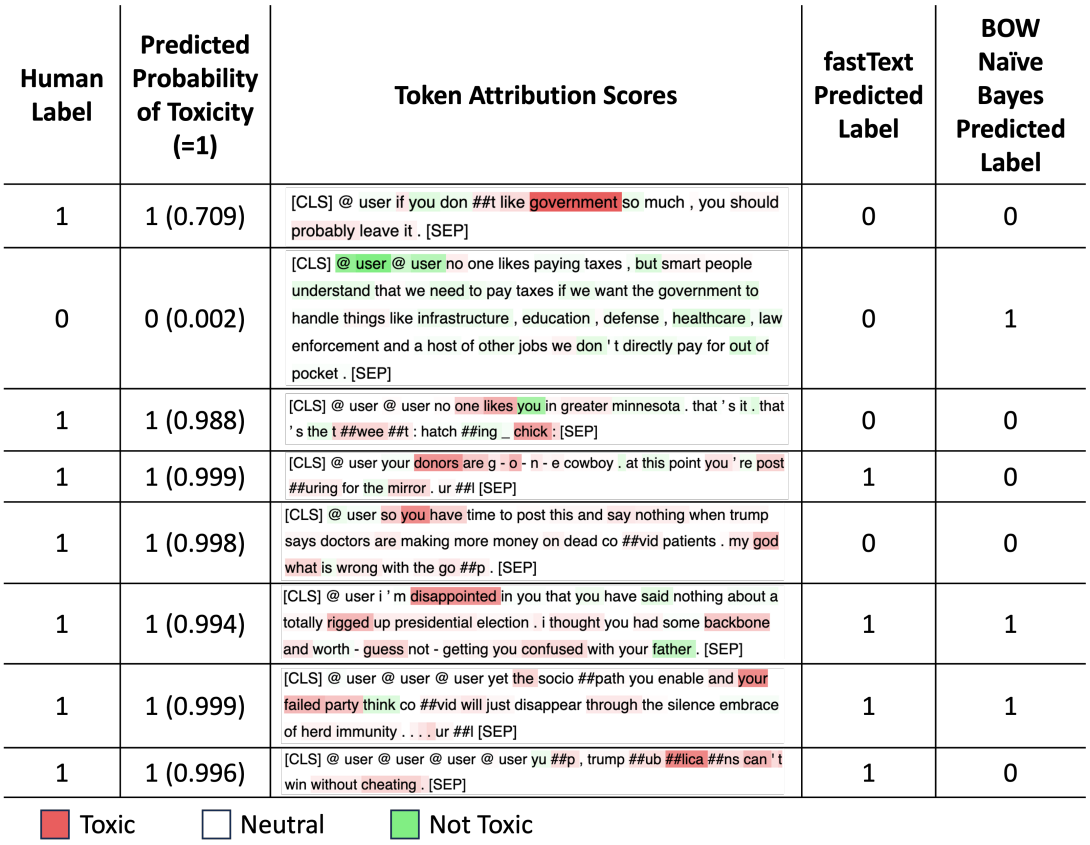



Figure 3: Example Tweets with Attribution Scores using Integrated Gradients Algorithm.

Integrated gradients is a feature attribution method originally proposed by Sundararajan, Taly and Yan (2017) that aims to attribute an importance value to each input feature of a machine learning model based on the gradients of the model output with respect to the input. In particular, integrated gradients define an attribution value for each feature by considering the integral of the gradients taken along a straight path from a baseline instance $x'$ to the input instance $x$.

For classification models, the gradient usually refers to the output corresponding to the true class label or to the class label predicted by the model. Let us consider an input instance $x$, a baseline instance $x'$ and a model $M : X \to Y$ which acts on the feature space $X$ and produces an output $y$ in the output space $Y$. We can define the function $F$ as $F(x) = M_k(x)$ with the index $k$ denoting the $k^{th}$ element of $M(x)$. The attributions $A_i(x, x')$ for each feature $x_i$ with respect to the corresponding feature $x'_i$ in the baseline

are calculated as:

$$A_i(x, x') = (x_i - x'_i) \int_0^1 \frac{\delta F(x' + \alpha(x - x'))}{\delta x_i} \, d\alpha \tag{3}$$

where the integral is taken along a straight path from the baseline $x'$ to the instance $x$ parameterized by the parameter $\alpha$.

The integrated gradients algorithm has three important axioms:

- Sensitivity axiom: if we consider a baseline $x'$ which differs from the input instance $x$ only for the value of one feature $x_i$ and yields different predictions, the attribution given to the feature $x_i$ must be non-zero.

- Implementation invariance axiom: an attribution method should be such that the attributions do not depend on the particular implementation of the model.

- Completeness axiom: the sum over all features attributions should be equal to the difference between the model output at the instance $x$ and the model output at the baseline $x'$.

$$\sum_i A_i(x, x') = F(x) - F(x') \tag{4}$$

The proofs are provided in the original paper by Sundararajan, Taly and Yan (2017).

To offer some interpretability of the classification process of the toxicity detection model, I use the integrated gradients algorithm to generate attribution scores for each token when classifying the input tweet as toxic or not (Sundararajan, Taly and Yan, 2017). Each token is considered an input feature to the model. The baseline used in this article is a numerical vector with all values set to zero. First, I interpolate the baseline input by adding the text tokens to resemble the actual text input token-by-token. Then I calculate the gradients with each additional token, measuring the change in the model's output (i.e., predicted label) with respect to the change in the input tokens. After all that, I accumulate the gradients using Riemann sums as the approximation method, which basically sums the gradients and divides them by the total number of steps. Figure 3 here shows example tweets where the attribution scores for each token are visualized. Red indicates toxicity, and green indicates no toxicity.

# Appendix G: Regression Results

Table 4: Results from Fixed Effects Model with Senator-clustered Standard Errors.

| | *Dependent variable: Toxicity* | | | |
|---|---|---|---|---|
| | Model 10<br>BERT-large-uncased-Twitter<br>(1) | Model 14<br>Dictionary-based<br>(2) | Model 15<br>BOW + Naive Bayes<br>(3) | Model 16<br>fastText + linear classifier<br>(4) |
| ParentToxicity | 0.0767*** | 0.0338*** | 0.3002*** | 0.0058 |
| | (0.0121) | (0.0032) | (0.0419) | (0.0052) |
| MentionTrump | 0.0729*** | 0.0199*** | 0.0761*** | 0.0585*** |
| | (0.0067) | (0.0034) | (0.0111) | (0.0062) |
| MentionBiden | -0.0122 | -0.0145 | -0.0672 | -0.0073 |
| | (0.0239) | (0.0138) | (0.0612) | (0.0247) |
| MentionEconomy | 0.0238** | 0.0194*** | 0.0785*** | 0.0434*** |
| | (0.0100) | (0.0070) | (0.0187) | (0.0100) |
| MentionCovid | 0.0318*** | 0.0204*** | 0.0611*** | 0.0376*** |
| | (0.0051) | (0.0034) | (0.0125) | (0.0047) |
| MentionAbortion | -0.0221 | 0.0233 | -0.0882 | -0.0211 |
| | (0.0447) | (0.0328) | (0.0815) | (0.0364) |
| MentionCrime | 0.0001 | 0.0347 | -0.0427 | -0.0038 |
| | (0.0386) | (0.0270) | (0.0837) | (0.0357) |
| MentionClimate | 0.0081 | 0.0005 | 0.0089 | 0.0257* |
| | (0.0149) | (0.0101) | (0.0266) | (0.0136) |
| MentionCopartisan | -0.1667*** | -0.0860*** | -0.3713*** | -0.1496*** |
| | (0.0102) | (0.0065) | (0.0270) | (0.0088) |
| MentionOutpartisan | -0.0153 | -0.0114 | 0.0282 | 0.0119 |
| | (0.0166) | (0.0077) | (0.0313) | (0.0240) |
| Observations | 20,072 | 20,072 | 20,072 | 20,072 |
| $R^2$ | 0.0763 | 0.0531 | 0.1149 | 0.0636 |
| Adjusted $R^2$ | 0.0713 | 0.0479 | 0.1101 | 0.0586 |
| F Statistic (df = 10; 19963) | 164.9973*** | 111.8771*** | 259.1146*** | 135.6425*** |
| *Note:* | | | *p<0.1; **p<0.05; ***p<0.01 | |