Political Science Research and Methods (2026), 1–27
doi:10.1017/psrm.2026.10120

ORIGINAL ARTICLE

High accuracy with low costs: the pretrain-finetune paradigm for classification with transformer-based language models

Cecilia Y. Sui

Political Science, Washington University in St. Louis, St. Louis, MO, USA
Email: c.sui@wustl.edu

(Received 8 June 2025 UTC; revised 11 November 2025 UTC; accepted 25 January 2026 UTC)

Abstract

Political science increasingly uses text classification to gauge subtle concepts such as toxicity or anger. Traditional methods treat words in isolation, overlooking the contextual dynamics where meaning resides. Transformer-based language models address these limitations but remain underutilized, especially among applied scholars, partly due to misconceptions about their mechanisms and computational requirements. This article bridges this gap by offering an accessible explication of the pretrain-finetune paradigm, focusing on underlying mechanisms and their potential to improve political text analysis by harnessing models trained on extensive datasets while requiring modest labeled data and computational resources. I demonstrate the approach by identifying toxic language in conversation threads following U.S. Senators' tweets and provide an online tutorial to support nontechnical scholars in adopting the methodology.

Keywords: domain-adaptive pre-training; language models; natural language processing; pretrain-finetune paradigm; text analysis; text classification; toxicity detection; transformer-based language models; transfer learning

1. Introduction

Text classification is now a standard tool for measurement tasks in political science across substantive domains (Justin and Stewart, 2013; Lucas *et al.*, 2015; Roberts, 2016; Goet, 2019; Bestvater and Monroe, 2023; Bosley *et al.*, 2025). 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 (Siegel and Badaan, 2020; Wasow, 2020; Jacobs *et al.*, 2021; Osmundsen *et al.*, 2021; Osnabrügge *et al.*, 2021). 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, 2023). 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.

[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).

© The Author(s), 2026. Published by Cambridge University Press on behalf of EPS Academic Ltd. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

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 (Vaswani *et al.*, 2017). These models leverage transfer learning, enabling efficient adaptation of pre-trained language representations to new tasks with minimal labeled data. Transfer learning, as implemented in transformer-based models, enables efficient knowledge transfer by first training on vast, unlabeled text corpora before fine-tuning on domain-specific labeled datasets. Such approach significantly enhances model performance while reducing data and computational requirements, making advanced natural language processing (NLP) techniques more accessible to applied scholars.

Although prior studies have applied transformer-based models to political science tasks, they primarily focused on specific applications or implementations rather than elucidating the underlying mechanisms. For instance, Bestvater and Monroe (2023) used BERT for stance detection, Widmann (2024) employed ELECTRA to analyze emotional appeals in German political discourse, Milliff (2024) used MuRIL to measure appraisals in oral violence-exposure histories, and Licht *et al.* (2025) measured anti-elite rhetoric across languages. However, these studies offered limited justification for the adoption of transformer-based language models and provided minimal discussion of the classification pipeline. On the other hand, Laurer *et al.* (2024) presented transfer learning in the context of BERT-NLI, which can be applicable to classification tasks but requires additional reformulation of the hypotheses to pair with the text and match the labels. Meanwhile, Timoneda and Vallejo Vera (2025) provided a valuable comparison of performance across different transformer architectures including BERT, RoBERTa, DeBERTa, and XLM-R, but did not delve into how these models actually work internally. Importantly, all the articles mentioned above omitted the explanation of the functional integration of the attention mechanism and pre-trained embeddings in facilitating transfer learning which contributes to the effectiveness of these models. Given the relatively rare uptake of this method in the field despite their clear advantages, what is needed is a detailed explanation of transformer-based language models for a standard classification tasks with a simple application for applied scholars to build from.

To fill this gap, this article provides a conceptual overview of the pretrain-finetune paradigm (PFP), offering an accessible explication of its methods, applications, and limitations. Through a step-by-step breakdown of how these models work internally, I make this technique more accessible to applied researchers without extensive technical backgrounds. I further demonstrate the practical implementation of the PFP through an application to toxicity detection in social media discourse, showcasing how these models can effectively capture nuanced concepts that traditional approaches often miss. The accompanying tutorial and code examples[2] further bridge the gap between theoretical understandings and practical implementation, enabling researchers to adopt these powerful tools with greater confidence and clarity about their inner workings.

In Section 2, I provide an overview of current methods used in the field of political science and discuss their limitations. I show that while there are a few papers that used the PFP approach (Bestvater and Monroe, 2023; Wang, 2024, 2023b; Burnham, 2025; Laurer *et al.*, 2024; Timoneda and Vallejo Vera, 2025), many published works using supervised learning still do not. To motivate the PFP, I then outline the limitations of the most widely used methods for political science domain-specific text analysis. Section 3 is dedicated to a comprehensive overview of the PFP focusing on the inner workings of the attention mechanism, while Section 4 walks through the process of using the PFP in details with an application to identify toxic language in Twitter conversation threads. The article

[2] https://github.com/CeciliaYSui/PFP_Tutorial/blob/main/Online_Tutorial.ipynb.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

concludes with a discussion on the limitations inherent in the PFP and suggests potential avenues for future research in this domain.

2. Current text classification approaches

Over the past two decades, quantitative text analysis in political science has evolved substantially—from early dictionary-based and topic-modeling approaches to the contemporary use of contextualized embedding models. This development reflects a broader methodological trajectory: initial efforts to identify political concepts relied on word counts and dictionary-based rules (Siegel and Badaan, 2020; Osnabrügge *et al.*, 2021), followed by probabilistic topic models (Catalinac, 2016; Roberts *et al.*, 2019), and later by neural feature extraction models such as word embeddings and, more recently, transformer-based architectures. The release of accessible software like quanteda (Benoit *et al.*, 2018) further democratized text-as-data research by lowering technical barriers for applied researchers and enabling large-scale, reproducible text measurement.

To contextualize the discussion, imagine we are interested in measuring toxicity from some text corpus that contains the two example sentences below.[3]

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

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

The simplest approach 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 (Siegel and Badaan, 2020; Wasow, 2020; Jacobs *et al.*, 2021; Osmundsen *et al.*, 2021; Osnabrügge *et al.*, 2021). 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 Section 4.

While dictionary-based methods offer speed, transparency, and ease of implementation, they also have important limitations 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 (Barberá *et al.*, 2021), 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.

A step forward came with bag-of-words (BOW) representations, which convert text into numerical vectors (i.e., text embeddings) by counting the occurrences of words or tokens from the input text, regardless of the token order (Justin and Stewart, 2013; Vries *et al.*, 2018; Barberá *et al.*, 2019; Beltran *et al.*, 2021; Cocco and Monechi, 2022). 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, despite its speed and straightforward implementation, 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.

[3]In Section 4, I measured toxicity in conversation threads from tweets by U.S. Senators, comparing PFP models with traditional approaches discussed in Section 2, and demonstrated the superior performance of PFP models.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

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 Harris and* not *Trump.*

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

A more advanced method of text representation is to use static word embeddings (D'Sa *et al.*, 2020; Alrababa'h *et al.*, 2021; Esberg and Siegel, 2023). To generate numerical representations in a continuous vector space and take advantage of the surrounding words, word2vec was 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 were 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 tokens is also limited by a window of tokens on the left or right, making it inadequate to connect two tokens that are further away from each other.

In response to the constraints in prior models, transformer-based language models were introduced to effectively manage the intricacies inherent in linguistic nuances (Vaswani *et al.*, 2017). They address three key challenges in prior text representation methods.

First, they produce dynamic embeddings, allowing the numerical representation of each word to depend on all other words in the same sequence. This enables the model to differentiate context-dependent meanings—for example, *'yellow'* in *E2* is associated with *'flowers,'* while in *E1* it acquires a negative connotation through its connection with *'Chinese,' 'evil'* and *'invasion.'*

Second, transformers incorporate word order through positional encodings—numerical vectors added to each token embedding that represent its position in the sequence. By combining semantic and positional information, transformers capture both meaning and structure, distinguishing between sentences such as *E3* and *E4*.

Third, unlike static word embeddings such as word2vec, which rely on a fixed local context window, transformers can model global dependencies across an entire text sequence.[4] Earlier architectures such as recurrent neural networks (RNNs) and long short-term memory (LSTM) models partially addressed local-context limitations by processing tokens sequentially and retaining prior information. Bidirectional LSTMs extended this capability by incorporating context from both directions, improving sequence-labeling tasks like named entity recognition (NER) and part-of-speech (POS) tagging. However, because these models process tokens one at a time, they are prone to losing information over long sequences.

Transformer encoders overcome such constraint through multi-head self-attention, which allows each token to attend to all others in the sequence simultaneously. Each attention head learns a different type of linguistic relationship, producing context-specific embeddings that reflect both local and global meaning. This attention-based computation, rather than a larger context window alone, is what

[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, classic encoder models such as Bidirectional Encoder Representations from Transformers (BERT) support sequences of up to 512 tokens, whereas newer long-context generative models can accommodate hundreds of thousands or millions of tokens.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

gives transformers their flexibility and superior representational power. Moreover, transformers produce context-rich representations that are resilient to linguistic variability, including typographical errors and nonstandard spellings common in social media data (Shawky *et al.*, 2024).

However, while transformer-based models have become dominant in NLP, they may not be universally optimal. RNNs and LSTMs still perform competitively on certain tasks—particularly sequence-labeling tasks like POS tagging and NER—and are less computationally demanding. For instance, models such as Flair have outperformed some transformer encoders on NER benchmarks (Yamada *et al.*, 2020). Thus, model selection should depend on the specific research goal, data availability, and computational constraints. In this article, transformer encoders were chosen because the primary task is context-dependent text classification, where capturing long-range semantic relationships and interpretive nuance is more critical than token-level prediction.

Despite the exceptional efficacy of transformer-based language models in text analysis (Park *et al.*, 2022; Karl and Scherp, 2023), their adoption within the field of political science remains limited. With a search of 2020-2025 volumes of the *American Political Science Review*, the *American Journal of Political Science*, and the *Journal of Politics* for all articles that used machine learning of texts to create measures of latent concepts (Park and Montgomery, 2025), 22 papers used topic modeling, 12 papers used dictionary-based methods, and 33 papers used some type of supervised learning for text analysis.[5] Among the 33 papers that used supervised learning, only eight leveraged transformer-based language models with seven implementing some degree of fine-tuning and none engaging in domain-adaptive pre-training (DAPT) (Gururangan *et al.*, 2020). Such limitation may partly be attributed to a lack of explication of the mechanisms of these models and misconceptions about the amount of labeled data and computational resources these methods require. In Section 3, I offer a comprehensive elucidation of the PFP focusing on the attention mechanism. For reproducibility, all computation in this article was conducted on Google Colab with 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 and the online tutorial.[7]

3. The pretrain-finetune paradigm

In this section, I first offer a detailed overview of the transformer encoder[8] with an emphasis on the attention mechanism and its integration with embedding generation. Then, I discuss the pre-training phase, during which the model acquires general linguistic patterns, followed by the fine-tuning stage, where task-specific data further improves the model's classification performance. Finally, I provide practical guidelines for scholars to implement the PFP with different levels of computational resources or available data at hand.

3.1. 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 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. Sections 3.1.1 to 3.1.4 examine each part in Figure 1 in details.

[5]The full list of papers is provided in Appendix 1.

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

[7]https://github.com/CeciliaYSui/PFP_Tutorial/blob/main/Online_Tutorial.ipynb.

[8]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-3/4/5, 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.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

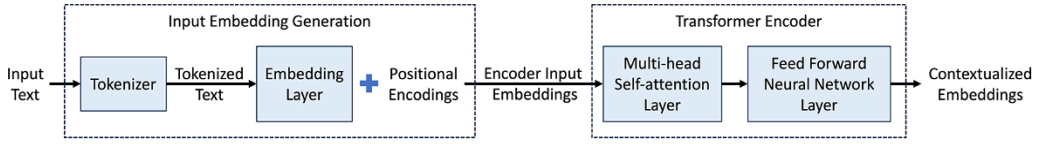


Figure 1. Transformer encoder architecture.

3.1.1. *Transformer encoder input embedding generation*

Consider *E1* again. As shown in Figures 1 and 2, the model first tokenizes[9] the input sequence into individual tokens. Then it maps the tokens into input IDs according to their position in the vocabulary containing all possible words in the training corpus. At this stage, both *'the*'s have the same input ID since they occupy the same position in the vocabulary. Then, the model takes the input IDs and maps 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 other previously pre-trained models or initialized randomly,[11] and the embedding layer is often the first trainable layer in a 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 positional encodings, 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 on the top in Figure 2 and positional encodings shown in the middle are summed together to construct the input embeddings to a transformer encoder illustrated in Figure 1.

3.1.2. *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 without introducing trainable parameters to build intuition.

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.

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

[10]512 is the dimensionality of the input and output representations in each encoder layer of the original transformer. It determines the size of the input embeddings and encoder outputs. Within each encoder layer, the feed-forward network expands these representations to an inner dimensionality of 2048 before projecting them back to 512 dimensions (Vaswani et al., 2017). BERT-base uses a corresponding model dimensionality (hidden size) of 768 with an inner feed-forward dimensionality of 3072, while BERT-large uses 1024 and 4096, respectively (Devlin et al., 2019).

[11]Token embeddings are generated by a learned embedding matrix that maps each token ID to a vector of a specified dimensionality. When training a model from scratch, the embedding matrix is typically randomly initialized; when fine-tuning a pre-trained model, it is initialized from the pre-trained model's learned parameters. The embedding weights are trainable and, unless frozen, are updated along with the model's other parameters during training.

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

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

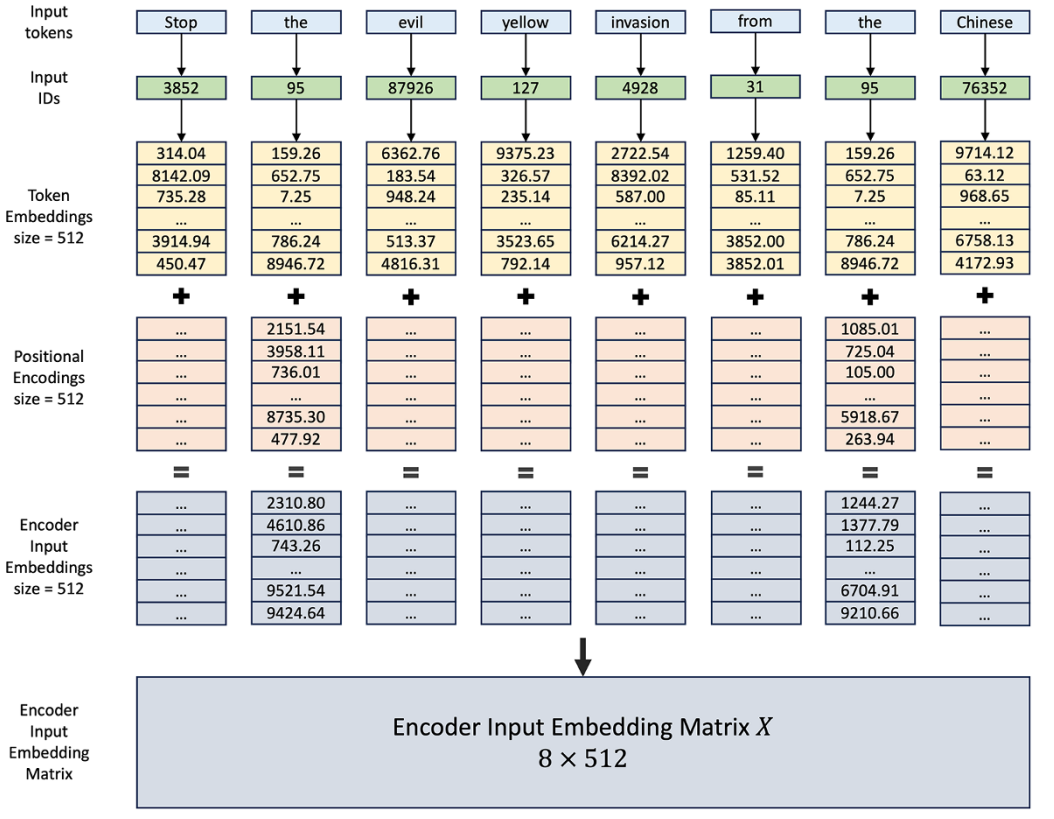


Figure 2. Input embedding generation to the transformer encoder. numbers are for illustration purposes only.

To disentangle the mechanism mathematically, imagine I want to calculate 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:[14]

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

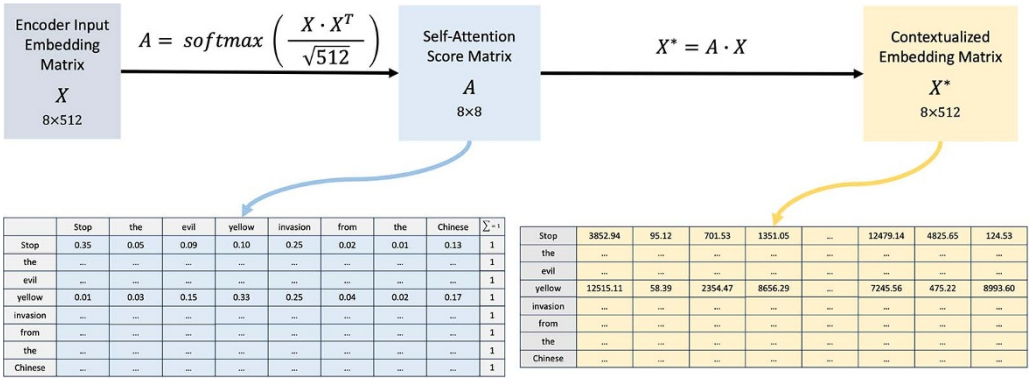
where:

X = the encoder input embedding matrix,
d = dimensionality of the model (see footnote 10.)

[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. In self-attention, the Query, Key, and Value matrices are all derived from the same input sequence, whereas other forms of attention differ in the source of these representations or in which positions may be attended to (Vaswani *et al.*, 2017; White, 2021).

[14]For simplicity, Equation 1 uses $X \cdot X^T$ to illustrate the intuition behind attention weighting. In practical transformer implementations (Vaswani *et al.*, 2017), self-attention is computed as $softmax\left(\frac{Q \cdot K^T}{\sqrt{d}}\right) \cdot V$, where Q, K, and V are learned linear projections of X (i.e., $Q = XW_Q$, $K = XW_K$, and $V = XW_V$). The simplified form corresponds to the special case where these projection matrices are identity transformations and is intended to convey the logic of attention weighting without introducing additional notations.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press



	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.

For *E1*, the input sequence length is 8 and the dimension d of the model and input embedding is 512,[15] which is the dimension of the encoder in the original paper that introduced transformers (Vaswani *et al.*, 2017). As shown in Figure 3, the model first conducts 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 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 matrix A in Figure 3, are the self-attention scores, where they represent the strengths of 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 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, the model conducts 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 positional encodings 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 add updatable weight matrices. Although in the simplified formulation of self-attention based on Equation 1 the model uses the same input embedding matrix X three times, in practice, each instance of X is multiplied by a distinct weight matrix W_Q, W_K, and W_V, as illustrated in Figure 4. These three weight matrices learn task-relevant linear projections of the same input sequence, each emphasizing different aspects of how tokens relate to one another. By separating the transformations this way, the model can capture multiple dimensions of contextual meaning and generate more expressive attention patterns. This operation yields the contextualized embedding matrix X^* as the

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

[16] Values along the diagonal of the self-attention matrix represent each token's dot product with itself and are often relatively large.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

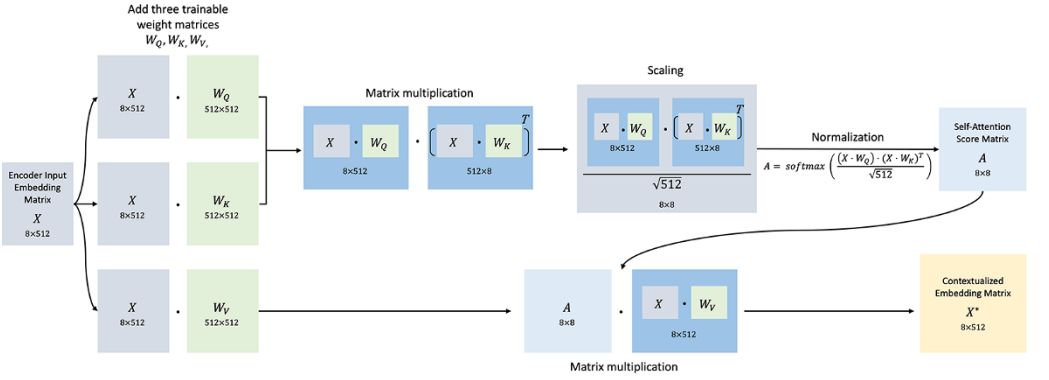


Figure 4. Self-attention with trainable weight matrices W_Q, W_K, and W_V.

output.[17] In practical transformer implementations, the self-attention mechanism therefore operates not directly on the original embeddings, but on these learned projections.[18]

3.1.3. 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 architectural difference for multi-head attention is that the model maintains one set of weight matrices for each attention head and then concatenates the resulting contextualized representations from all heads afterwards. Consider a transformer encoder with two attention heads[19] as shown in Figure 5. Notice that the encoder input embedding matrix is split into two smaller submatrices X_1 and X_2 to use as the input for each attention head. The input embedding is typically split across the embedding dimension (columns) instead of the sequence length dimension (rows).[20] Such approach allows separate sections 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.[21]

Within each attention head, the model performs the same self-attention operation as shown earlier in Figure 3, producing one contextualized embedding matrix per head. Then it concatenates 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, the model adds 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. The matrix W_O allows the model to capture more complex interactions across different heads.

[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, even the initial computation of attention scores involves learned transformations of the input embeddings.

[19] In practice, most language models have more than two attention heads. For example, BERT-base has 12 attention heads per encoder layer, BERT-large has 16, and the largest GPT-3 model uses 96 attention heads per layer.

[20] This split is typically logical rather than physical. In implementation, a single W_Q, W_K, and W_V can each produce the projections for all attention heads, which are then reshaped into separate heads. $W_Q W_K W_V$

[21] 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.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

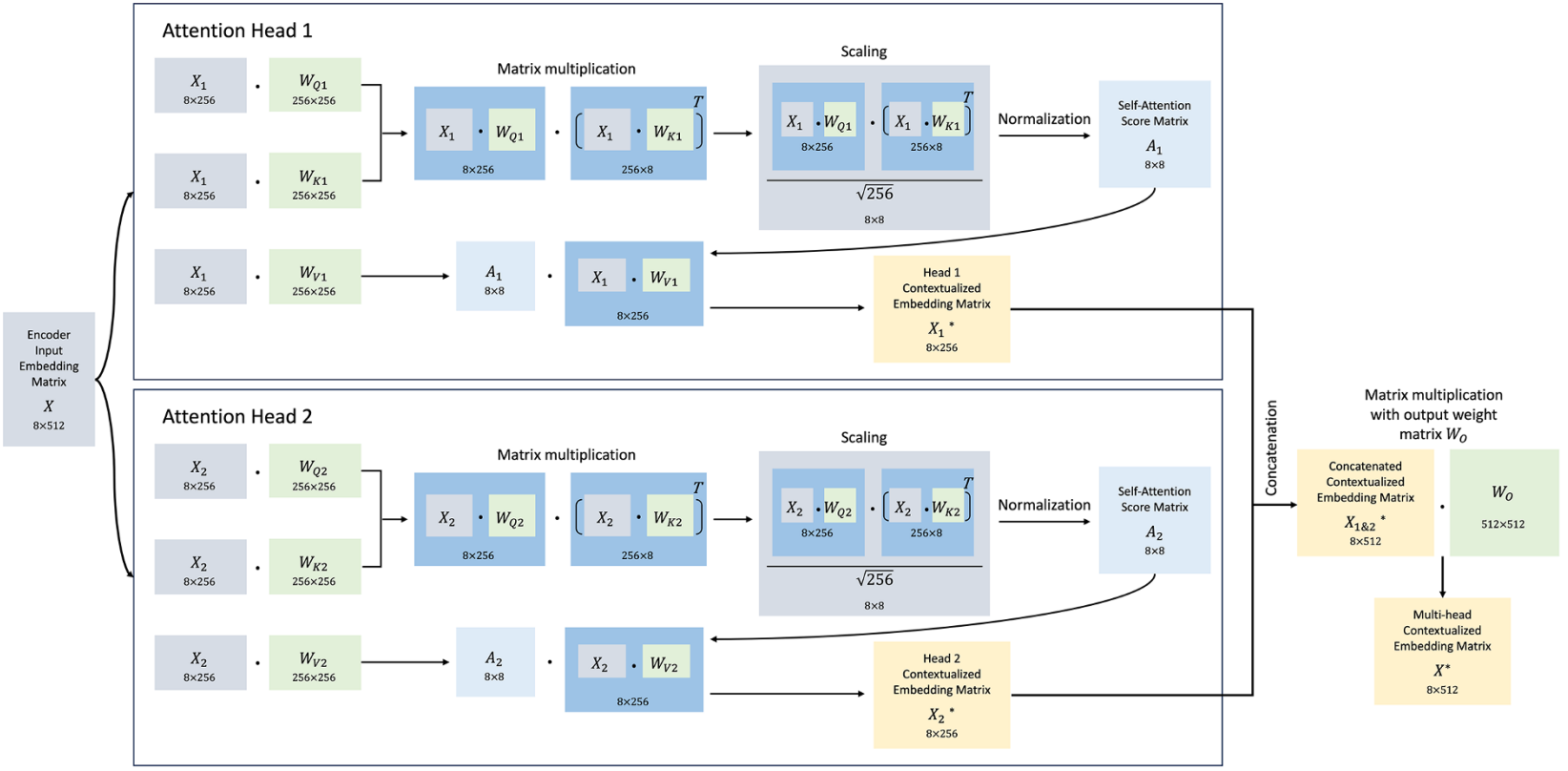


Figure 5. Calculation of the multi-head self-attention with two attention heads.

During training, all weight matrices are initialized with small random values and updated iteratively through backpropagation. In this process, the model first generates predictions based on its current parameters and compares them to the true labels[22] to compute a loss function that quantifies prediction error. The model then computes the gradient and uses this information to adjust the parameters in the direction that reduces the loss. This gradient-based optimization is repeated across many iterations and batches of data, gradually refining the weight matrices to minimize overall error. Once training converges, these learned parameters encode how strongly each token attends to others across multiple representational subspaces, and they are subsequently retained for downstream inference.

3.1.4. 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[23] is a feed-forward neural network (FNN).[24] The FNN[25] receives the output from the multi-head self-attention layer, denoted as X^* in Figure 5, and processes it through 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.[26] 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.

3.2. Pre-training and fine-tuning

In this section, I examine the PFP of encoder-only language models, which typically comprise of multiple transformer encoders stacked together, 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 (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

[22] During masked language modeling, the 'true labels' correspond to the original tokens that were masked out in the input sequence. The model predicts these tokens based on surrounding context, and backpropagation updates all parameters to reduce the difference between predicted and true tokens, typically using a cross-entropy loss function.

[23] 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).

[24] 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.

[25] 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.

[26] Non-linearity in the FNN is introduced through a non-linear activation function, such as ReLU in the original transformer (Vaswani et al., 2017).

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

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.

3.2.1. Domain definition

Before turning to the implementation of the PFP, it is useful to clarify what a domain is in this context. In NLP research, a domain refers to a collection of texts that share systematic linguistic regularities—topical, stylistic, or community-specific—that differ from the general corpora on which base models such as BERT are originally pre-trained. These differences may arise from topical specialization (e.g., legislative debates or policy documents that employ distinctive jargon), stylistic variation (e.g., the informal, sarcastic tone of social-media discourse), or community-specific lexicons that encode culture-bound meanings (e.g., ethnic slurs or political slogans).

Distinguishing among these forms of domain variation is important because they pose distinct representational challenges for language models. Polysemous variation, as in the word *'left,'* represents a linguistic-semantic challenge: the same surface form can express multiple, unrelated meanings. Resolving such ambiguity depends primarily on the model's ability to use local syntactic and semantic context—a strength already present in contextual encoders through mechanisms such as multi-head self-attention. In contrast, socio-cultural or community-specific variation, exemplified by the phrase *'yellow invasion,'* reflects a cultural-pragmatic challenge. Its pejorative sense depends on historically and socially embedded usage that may be under-represented in the general pre-training data. Addressing this type of variation requires additional exposure to in-domain texts in which such expressions appear, allowing the model to learn their contextual associations and pragmatic connotations.

Domain-adaptive pre-training (DAPT) directly addresses this latter problem by continuing pre-training on a large, unlabeled corpus drawn from the target domain before task-specific fine-tuning. Through this intermediate adaptation step, the model updates its internal representations to reflect domain-specific linguistic patterns without discarding the broad semantic knowledge acquired during initial pre-training. In short, DAPT allows the model to better align its representational space with the statistical properties of the target corpus, thereby improving downstream classification performance when the new data differ markedly from the general-domain text.

In later sections, I return to this distinction to show that when the target domain is only moderately distinct from the general pre-training corpus—as in the case of political discourse on Twitter—the empirical gains from DAPT remain limited, underscoring the importance of matching method to domain specificity.

3.2.2. Pre-training stage

To contextualize the following discussion, I use the classic BERT[27] as an example of encoder-only models (Devlin *et al.*, 2019). Encoder-only models exploit transfer learning (Zhuang *et al.*, 2021).[28] 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

[27] BERT(-base-uncased) 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 encoder layers. 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 768×64 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-uncased model. BERT-large consists of 24 encoder layers with a hidden size of 1024 and 16 attention heads per layer. It has approximately 340 million parameters (Devlin et al., 2019).

[28] 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.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

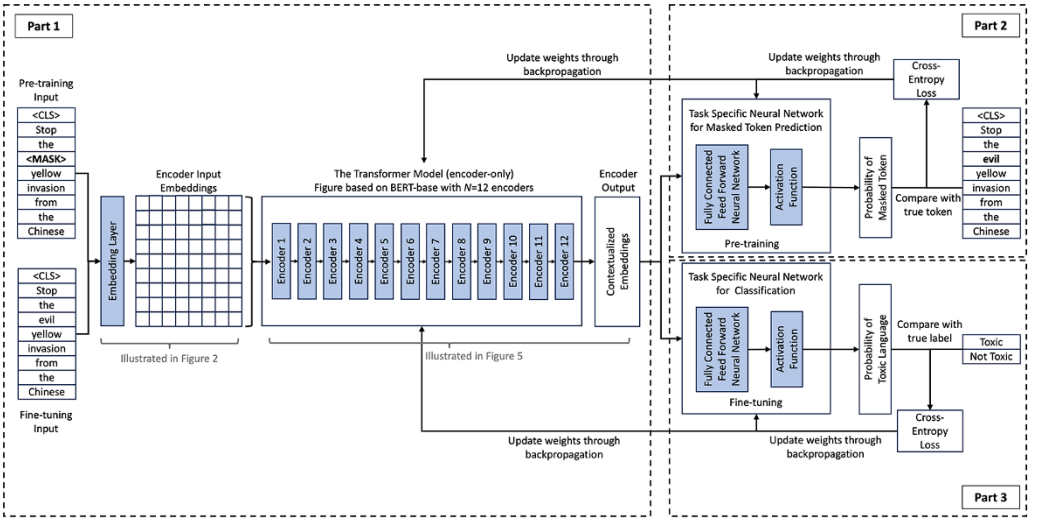


Figure 6. Pre-training and fine-tuning for BERT-like encoder-only models. Shaded boxes contain trainable parameters.

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.[29]

The pre-training process of a BERT-like encoder-only model is shown in Parts 1 and 2 in Figure 6. The input tokens (along with a special < CLS > token[30] denoting the start of the sequence specific for BERT-like models) are converted to token embeddings through the embedding layer as described in Section 3.1.1 and Figure 2. These embeddings are passed through a series of transformer encoders to create a set of output embeddings as illustrated in Section 3.1.3 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.[31] In pre-training, the goal is to predict the missing token from the associated output embedding.[32] The model then calculates the loss from each of its predictions to update the weights for all learnable layers as highlighted in shaded boxes in Figure 6 through backpropagation by calculating the gradients or derivatives of the loss with respect to the weights (Devlin *et al.*, 2019).

[29]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).

[30]In BERT-style encoder models, a special token < CLS >, short for classification, is automatically inserted at the start of every input sequence. During training and inference, the model learns a contextual embedding for this token that summarizes information from the entire sequence. This single vector is then passed to the task-specific classifier head for predicting labels such as toxicity. It does not appear in natural text, and is part of the model's vocabulary similar to < SEP > (added at the end of a sequence) and < PAD > (added for padding). When BERT was designed, researchers needed a way to summarize the entire input sequence into a single vector for downstream classification tasks to avoid having to average or pool over every token manually. The researchers added < CLS > and trained the model so that its final layer embedding for < CLS > carries a global representation of the whole sequence.

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

[32]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.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

In the PFP, researchers can optionally conduct DAPT from a publicly pre-trained model to improve classification performance by repeating the pre-training process (Gururangan *et al.*, 2020), such as training with masked language modeling on domain-specific data, to update trainable parameters in both Parts 1 and 2 shown in Figure 6. Part 1 is then saved as the domain-adapted 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 section.

3.2.3. Fine-tuning stage

Considering the substantial parameterization inherent to language models,[33] pre-training or DAPT 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.[34]

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; Chalkidis *et al.*, 2020; Lee *et al.*, 2020; Caselli *et al.*, 2021). In the fine-tuning stage, the model parameters are updated to specialize the model to a particular task.

As illustrated in Parts 1 and 3 in Figure 6, to fine-tune[35] a pre-trained model like BERT, scholars can append a task specific neural network on top of the encoder-only 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, and can take just a few minutes to complete. In Section 4, I apply the PFP approach to 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 Section 2 using an annotated sample Twitter dataset of conversation threads following tweets from US Senators one month before and after the 2020 Presidential Election.

3.3. Guidelines of the PFP

Before outlining practical implementation steps, it is useful to briefly consider why the PFP is particularly effective for political text analysis compared to earlier modeling frameworks. Earlier architectures—such as static embedding and LSTM-based models—often captured domain-specific

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

[34] Platforms like the HuggingFace Hub hosts over two million open-sourced models that are publicly available for anyone to use, thereby providing a viable starting point. For example, there are models for measuring emotions (j-hartmann/emotion-english-distilroberta-base), populism (AnonymousCS/populism_model334), or ideology (juan-glez29/ideology-FacebookAI-xlm-roberta-large).

[35] More specialized fine-tuning strategies—such as layer-wise learning rate decay (LLRD), discriminative fine-tuning, and Performance-based Adaptive Fine-tuning (PAF)—can yield marginal performance gains by adjusting how different layers of a pre-trained model are updated (Howard and Ruder, 2018; Sun *et al.*, 2019; Bu *et al.*, 2024). While these approaches can improve performance in some settings, they add hyperparameter-tuning and implementation complexity. In this article, I employ a simplified fine-tuning setup to prioritize clarity, accessibility, and replicability for applied researchers. The PFP thus demonstrates that stable, high-quality results can be achieved without relying on these more specialized layer-wise optimization techniques.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

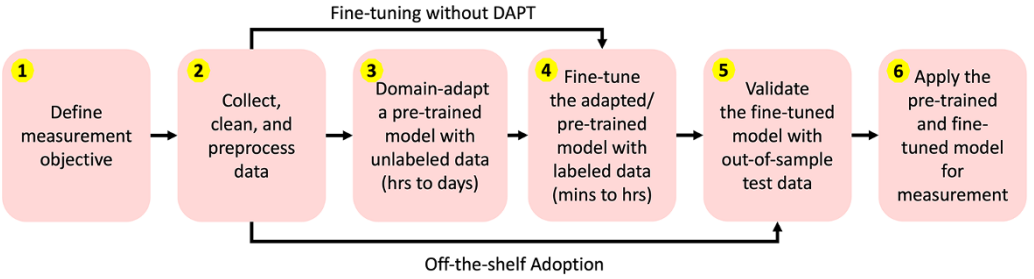


Figure 7. Text classification workflow following the PFP. Intermediate steps can be skipped when working with limited computational resources and data.

patterns but struggled to generalize across new datasets or topics without substantial retraining. In contrast, transformer encoders leverage large-scale pre-training to learn contextualized representations that transfer effectively across domains and time periods. Liu and Ritter (2023) demonstrated that transformers exhibit greater robustness and temporal generalization than earlier sequential models. These findings underscore the rationale for adopting the PFP: it capitalizes on the generalization strengths of pre-trained models while allowing applied researchers to adapt models efficiently to their own corpora and research questions.

There are several ways to use the PFP with different levels of data and compute availability. Figure 7 shows the entire text classification workflow.

When 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 to 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 such direct application 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 Section 4, 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 especially when utilizing models with lower levels of customization (Justin and Stewart, 2013; Park and Montgomery, 2025).

For researchers with some annotated data and limited compute resources, they can download a publicly pre-trained model and use the generated embeddings to train a classifier on top of the model (Licht *et al.*, 2025), where only Part 3 in Figure 6 is updated.

For researchers with more compute resources, a more tailored approach is to start with a domain-adapted model that has been pre-trained on a relevant corpus, and then fine-tune it using a small annotated dataset, such as a few hundreds labeled tweets, where both Parts 1 and 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 compute 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 publicly pre-trained model and conduct DAPT with the model by re-doing the pre-training tasks to update Parts 1 and 2 in Figure 6 and save Part 1 as the domain-adapted model. Then conduct fine-tuning to update Parts 1 and 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.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

Table 1. List of all models used in Section 4

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	froze encoders PFP	00:32
13	adapted-BERT-base-uncased-Twitter	DAPT 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

In the following 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 2.

4. Application to toxic language classification

In this section, I provide a step-by-step walk-through of the guidelines illustrated in Figure 7, with an application to toxic language classification to validate the PFP and serve as a template for scholars to build from. The objective is to systematically compare the performance of various PFP models against each other and against baseline models to assess their out-of-sample validity, face validity, and predictive validity through a real-world application. Table 1 provides a complete list of all models used, which are discussed with greater details in Appendix 6.

4.1. Data preparation

The proliferation of toxic language has intensified with the widespread use of social media (Massaro and Stryker, 2012; Chandrasekharan *et al.*, 2017; Rheault *et al.*, 2019; Agarwal *et al.*, 2021; Matamoros-Fernández and Farkas, 2021; Tamara *et al.*, n.d..). Accurately measuring toxicity within large datasets remains challenging due to the absence of an unambiguous and universally accepted definition. The PFP allows scholars to customize models based on their own theoretical definitions, instead of relying on models trained with definitions provided by others which are often not transparent. Appendix 3 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*

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

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[36] posted by U.S. Senators from the 116th U.S. Congress between October 1 and November 30, 2020, using the Twitter Academic API. Additionally, conversation threads[37] 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 intercoder reliability rate of 93%. After excluding cases with inconsistent annotations, the final dataset contained 2,790 tweets with full agreement, of which 1,149 (41.2%) were labeled as toxic and 1,641 (58.8%) as non-toxic. For model training, I randomly divided the annotated dataset into training (80%), validation (10%) and test (10%) subsets. Detailed information on the data annotation process is available in Appendix 4.

I compare five categories of models: off-the-shelf PFP models trained on existing toxicity benchmarks, PFP models fine-tuned on the annotated Twitter data, a frozen-encoder PFP model that updates only the task-specific classification layer, a domain-adapted and fine-tuned PFP model, and four baseline models based on dictionary methods, BOW with Naive Bayes, fastText with a linear classifier, and Google's Perspective API. For consistency in evaluation, the test set is used to assess all models in Table 1. Full model descriptions, implementation details, and additional evaluation tables are provided in Appendix 6.

4.2. Validation of the PFP

To validate the PFP, I first assess the out-of-sample performance of each model listed in Table 1 alongside their computational costs in terms of training time, focusing on balanced accuracy scores. Second, I examine the content validity of this measure by looking at example tweets classified by the models and whether the results align with theoretical definitions and human intuition. Then, I further examine the face validity of the measure by comparing the proportion of toxic tweets in the human annotated random sample and the classified full dataset. Finally, I investigate the predictive validity of the toxicity measure through an exploration of escalation of toxicity within conversation threads on Twitter. Results indicate that all fine-tuned PFP models outperform the baseline models for the toxicity classification task, regardless of DAPT.

4.2.1. Out-of-sample performance

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 A1, A2, and A3 in Appendix 6. Across all metrics, the fine-tuned PFP models exhibit significantly improved performance relative to the baseline models consistently. Among these, Model 10 achieves the highest balanced accuracy score of 0.8860, highlighting its effectiveness for toxic language classification. Figure 9 shows the confusion matrix for Model 10. Confusion matrices for Models 14-17 are provided in Appendix 6.

As shown in Table 1, fine-tuning times for the models varied, with most models requiring only a few minutes. Details are included in Appendix 6. 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. It is the worst-performing model, and should not be used for research in general. This underscores the importance of updating the parameters

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

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

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

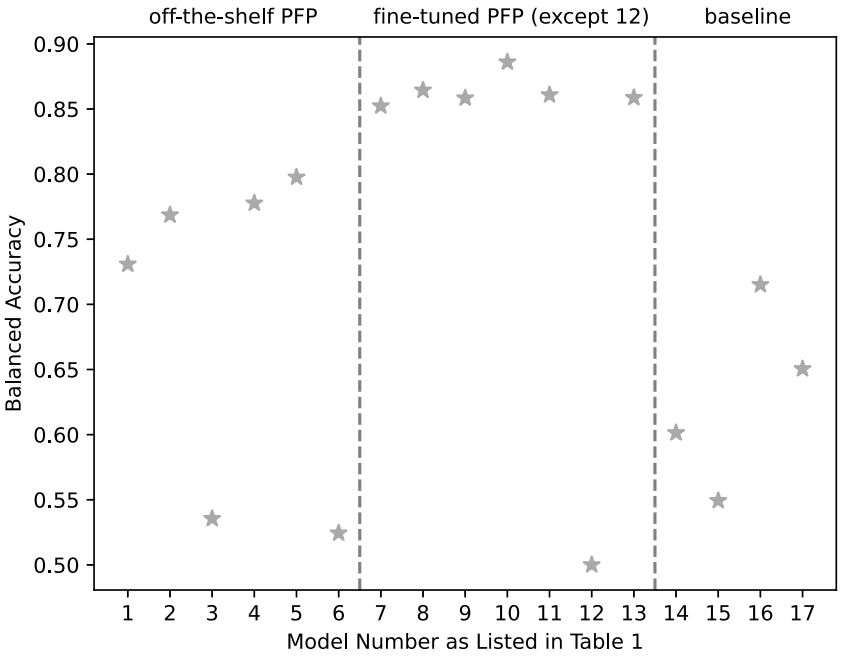


Figure 8. Balanced accuracy scores of all models.

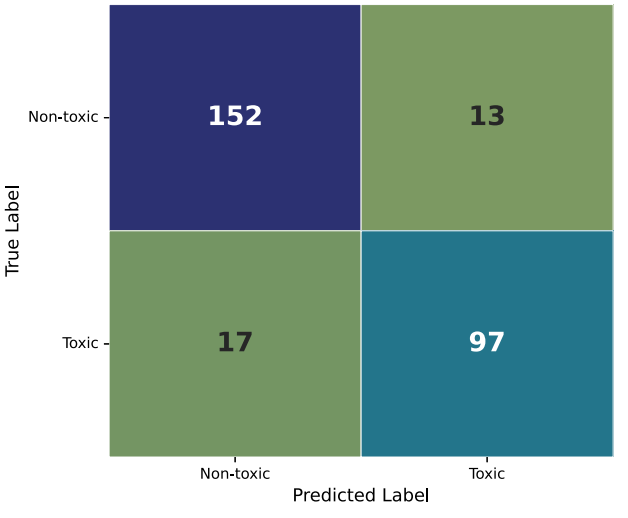


Figure 9. Confusion matrix for model 10.

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.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

Table 2. Examples of disagreed tweets in the test set. Probabilities are for the predicted label

		PFP	Baseline			
Text	True Label	Model 10	Model 14	Model 15	Model 16	Model 17
'@SenatorRomney Fucking mormon hypocrite'	1	1 (0.9962)	1	0	0	1
'@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
'@JohnCornyn Bi Partisan and non partisan constituents urge lawmakers to pass a comprehensive pandemic relief bill. https://t.co/rV3fOuXDQG'	0	0 (0.9882)	1	1	1	0
'@CoryBooker Tell us again why you want us to wear masks for the flu? https://t.co/OvJMEatUfb'	0	0 (0.9787)	0	1	1	0
'@Fire_In_Babylon @ChrisMurphyCT I mean gay marriage was a 5 − 4 decision at the Supreme Court abs could be overturned with ACB on the court. So maybe let's not say which issues are worse'	0	0 (0.9673)	1	1	0	0

Interestingly, the process of DAPT 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. It is important to note that the Twitter corpus used in this study represents a moderately domain-shifted context rather than a highly specialized linguistic environment. Political discourse on social media differs from the formal, encyclopedic style of the corpora on which BERT was originally pre-trained, yet it does not feature entirely novel or domain-exclusive vocabulary, such as medical or legal documents. Consequently, the limited additional benefit from DAPT observed in Model 13 is theoretically consistent with expectations. This finding is informative for applied researchers: it suggests that while DAPT can substantially improve performance in domains with distinct lexicons—such as medical or legal texts—it may offer only marginal returns for general political communication. The empirical illustration thus serves as a baseline case, demonstrating that DAPT is not always necessary and that task-specific fine-tuning alone can often achieve strong performance at much lower computational cost.

4.2.2. Content validity

Beyond evaluating classification metrics, I also assess the content validity of PFP models in comparison to the baseline models. To this end, I examine instances where Model 10, the best-performing PFP model, classified tweets as toxic with highest probabilities, while baseline models yielded divergent classifications. Table 2 highlights examples of such classifications. In many cases, the baseline models

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

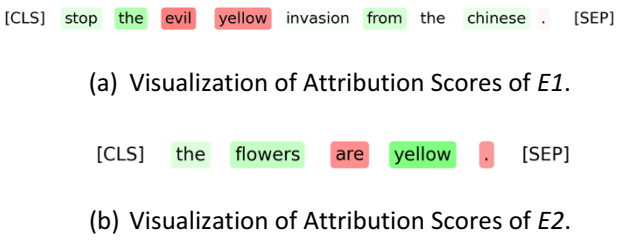


Figure 10. Token-level attribution scores with integrated gradients algorithm.

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.

To enhance interpretability of the models' classification decisions, I employed the integrated gradients (IG) algorithm to calculate attribution scores for each token when classifying the input tweet as toxic or not (Sundararajan *et al.*, 2017). The IG method quantifies how much each input token contributes to the model's prediction by computing the gradients of the output with respect to the token embeddings, and then integrating these gradients along a path from a neutral baseline input (e.g., an empty or zero vector) to the actual input. Tokens with higher positive attribution scores increase the model's confidence in the predicted class (e.g., toxicity), whereas tokens with negative scores suppress that prediction.

Figure 10 provides a representative example comparing *E1* and *E2* both containing the word *'yellow'*, showing how words such as *'evil'* contribute positively to toxicity detection, while neutral words receive near-zero or negative attribution. Table A4 in Appendix 7 visualizes token-level attributions for additional examples. This interpretability approach allows for an intuitive understanding of how the model weighs contextual cues when making predictions—revealing that transformer-based models capture nuanced linguistic relationships that simpler approaches cannot. More comprehensive details regarding the implementation and interpretive value of integrated gradients in this context are provided in Appendix 7.

4.2.3. Proportion validity

I further analyze the aggregate classification outcomes by comparing the proportion of tweets labeled as toxic and non-toxic by human annotators versus the models. Since the human-annotated dataset is a random sample from the full dataset, it provides a strong approximation of the true proportion of toxic and non-toxic tweets. While the models align closely in ranking senators based on the volume of toxic tweets in their conversation threads, notable discrepancies emerge in the absolute counts and proportions. Figure 11 presents the proportion of toxic and non-toxic tweets classified across different models, with the dotted line indicating the proportions identified in the human-annotated random sample. Notably, Model 15 classifies more than twice the proportion of toxic tweets compared to the human benchmark, whereas Model 14 substantially underestimates toxicity. Model 10 yields the closest approximation to the human-annotated proportion, followed by Model 16.

These discrepancies persist when disaggregating results by gender and party affiliation. As shown in Figure 12, Model 15 continues to classify a substantially higher proportion of tweets as toxic, whereas Model 14 reports significantly lower proportions compared to the human-annotated random sample. Among the baseline models, Model 16 most closely aligns with Model 10 and the

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

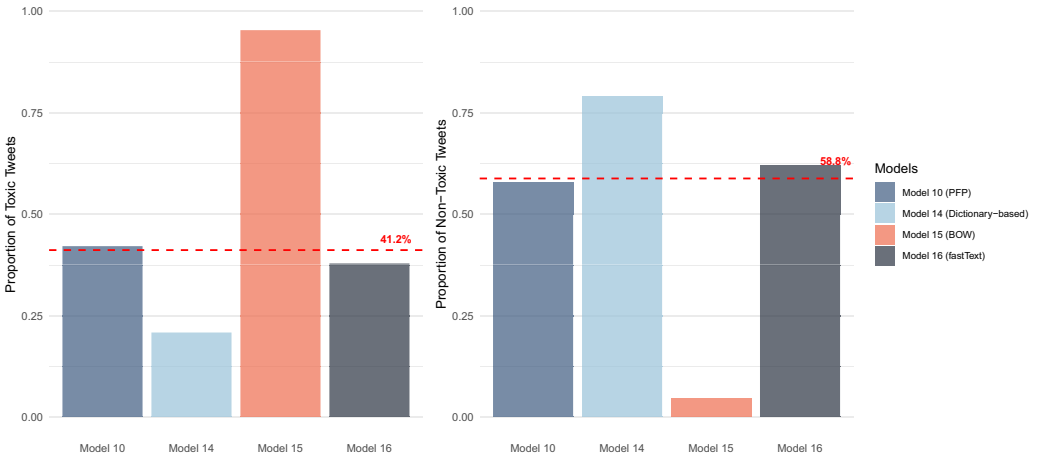


Figure 11. Bar-plots of Proportion of Toxic and Non-Toxic Tweets Labeled by Different Models.

human benchmark in overall classification proportions, where others demonstrate considerable misalignment. This analysis highlights Model 10's superior performance in maintaining both contextual sensitivity and reliability in toxicity detection.

4.2.4. Predictive validity

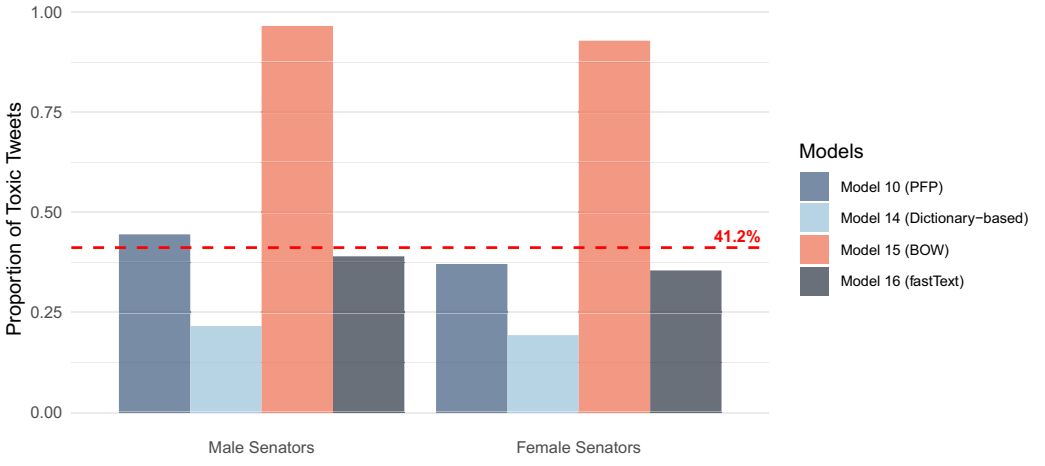
To demonstrate the potential usefulness of the PFP in political science and check predictive validity of the toxicity measure, I analyze how Twitter user-base responded to tweets from Senators of the 116^{th} U.S. Congress and explore whether factors correlate to toxicity within conversation threads in theoretically expected directions.

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 (Druckman *et al.*, 2013; Iyengar and Westwood, 2015; Rogowski and Sutherland, 2016; Abramowitz and Webster, 2018; Iyengar *et al.*, 2019). 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 (Steve *et al.*, 2021; Huszár *et al.*, 2022). 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 (Munger, 2017; Steve *et al.*, 2021; Huszár *et al.*, 2022). Therefore, I propose the following hypothesis:

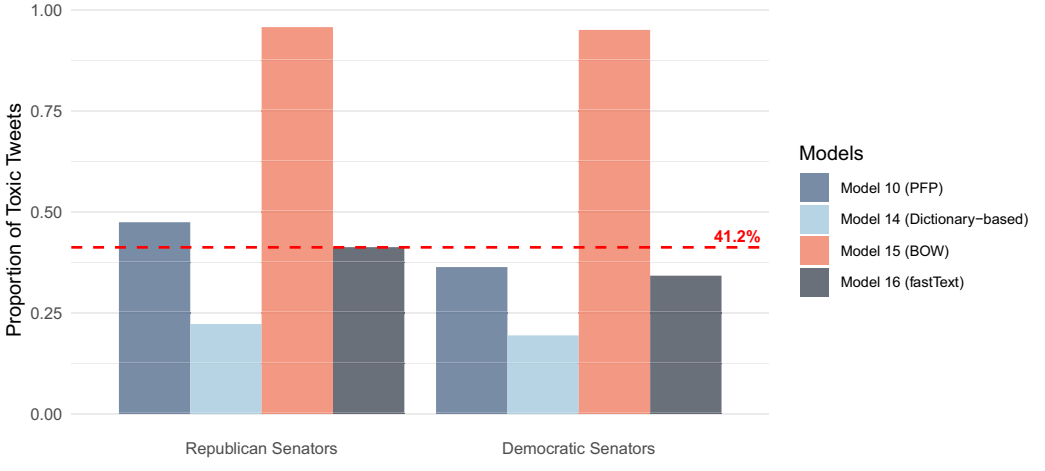
> Toxic parent tweets are associated with higher levels of toxicity in the following conversation threads.

To validate 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

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press



(a) Proportion of Toxic Tweets Labeled by Different Models by Gender.



(b) Proportion of Toxic Tweets Labeled by Different Models by Party.

Figure 12. Proportion of toxic tweets labeled by different models.

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.[38] I also control for Senator-specific fixed effects.

[38]The indicator variables for subjects, topics, or partisanship are not mutually exclusive. Individual posts may reference multiple subjects or parties simultaneously. For instance, a tweet in the dataset states: *'I commend the Trump Administration for its commitment to making sure that patients—particularly seniors and low-income patients—will be able to get COVID-19 vaccines administered at no cost and without the threat of receiving a surprise medical bill months later.'* This example illustrates that both a partisan actor (Trump) and a policy topic (COVID-19) can appear within the same observation, such that multiple dummy variables take the value of 1.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

Table 3. Results from fixed effects model with senator-clustered standard errors

	Dependent variable:			
	Toxicity			
	Model 10	**Model 14**	**Model 15**	**Model 16**
	BERT-large-uncased-Twitter	*Dictionary-based*	*BOW + Naive Bayes*	*fastText + linear classifier*
	(1)	(2)	(3)	(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: Full regression table including other control variables is provided in Appendix 8. 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.

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:

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

with Senator-specific intercept α_i and error term $\in_i$. 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-estimation of toxicity, as evidenced in Section 4.2.3. 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 shown in Section 4.2.3, Model 16 fails to capture a statistically significant relationship between *ParentToxicity* and toxicity in subsequent threads. For further details, Table A5 in Appendix 8 provides a complete table of regression coefficients. Additionally, parent tweets that mention Trump, the economy, and COVID are also associated with increased toxicity in conversation threads, whereas tweets mentioning co-partisan Senators correspond with lower toxicity levels in the conversation 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.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

5. Conclusion

This article makes an important contribution to political science by 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 and should 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 2 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 7, 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.

Acknowledgements. Significant thanks to Profs. Jacob Montgomery, Christopher Lucas, and Ted Enamorado for their role in advising, reading, and offering improvements to this paper. Thanks to Prof. Margit Tavits, Jeremy Siow, the WashU Political Science Research Workshop, and the WashU Political Data Science Lab for further feedback.

Data availability statement. Replication data and code can be found in Harvard Dataverse: https://doi.org/10.7910/DVN/STDKZ2.

Declaration of interests. None

Funding statement. None

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

Supplementary material. The supplementary material for this article can be found at https://doi.org/10.1017/psrm.2026.10120. To obtain replication material for this article, https://url.uk.m.mimecastprotect.com/s/3BAPC7L88smj5N4Ec8fXfopsNB?domain=doi.org

References

Abramowitz AI and Webster SW (2018) Negative partisanship: Why Americans dislike parties but behave like rabid partisans. *Political Psychology* **39**(S1), 119–135.

Agarwal P, Hawkins O, Amaxopoulou M, Dempsey N, Sastry N and Wood E (2021) Hate speech in political discourse: A case study of UK MPs on stwitter. *Proceedings of the 32nd ACM Conference on Hypertext and Social Media*, HT **21**, 5–16.

Alrababa'h A, Marble W, Mousa S and Siegel AA (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.

Barberá P, Boydstun AE, Linn S, McMahon R and Nagler J (2021) Automated text classification of news articles: A practical guide. *Political Analysis* **29**(1), 19–42.

Barberá P, Casas A, Nagler J, Egan PJ, Bonneau R, Jost JT and Tucker JA (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 F, Camacho-Collados J, Neves L and Espinosa-Snke L (2020) *TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification*.

Beltran J, Gallego A, Huidobro A, Romero E and Padró L (2021) Male and female politicians on twitter: A machine learning approach. *European Journal of Political Research* **60**(1), 239–251.

Benoit K, Watanabe K, Wang H, Nulty P, Obeng A, Müller S and Matsuo A (2018) Quanteda: An R package for the quantitative analysis of textual data. *Journal of Open Source Software* **3**(30), 774–74.

Bestvater SE and Monroe BL (2023) Sentiment is not stance: Target-aware opinion classification for political text analysis. *Political Analysis* **31**(2), 235–256.

Bosley M, Kuzushima S, Enamorado T and Shiraito Y (2025) Improving probabilistic models in text classification via active learning. *American Political Science Review* **119**(2), 985–1002.

Bu C, Liu Y, Huang M, Shao J, Ji S, Luo W and Wu X (2024) Layer-wise learning rate optimization for task-dependent fine-tuning of pre-trained models: An evolutionary approach. *ACM Transactions on Evolutionary Learning and Optimization* **4**(4), 1–23.

Burnham M (2025) Stance detection: A practical guide to classifying political beliefs in text. *Political Science Research and Methods* **13**(3), 611–628.

Caselli T, Basile V, Mitrović J and Granitzer M (2021) *HateBERT: Retraining BERT for Abusive Language Detection in English*.

Catalinac A (2016) From pork to policy: The rise of programmatic campaigning in Japanese elections. *The Journal of Politics* **78**(1), 1–18.

Chalkidis I, Fergadiotis M, Malakasiotis P, Aletras N and Androutsopoulos I (2020) LEGAL-BERT: The Muppets Straight Out of Law School. In *Findings of the association for computational linguistics: EMNLP 2020*, pp. 2898–2904.

Chandrasekharan E, Pavalanathan U, Srinivasan A, Glynn A, Eisenstein J and Gilbert E (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), 1–22.

Chang C and Masterson M (2020) Using word order in political text classification with long short-term memory models. *Political Analysis* **28**(3), 395–411.

Cocco JD and Monechi B (2022) How populist are parties? Measuring degrees of populism in party manifestos using supervised machine learning. *Political Analysis* **30**(3), 311–327.

Collobert R and Weston J (2008) A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th International Conference on Machine Learning* ICML **08**, 160–167.

D'Sa AG, Illina I and Fohr D (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.

Devlin J, Chang M-W, Lee K and Toutanova K (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies*, volume 1 (long and short papers), pp. 4171–4186.

Druckman JN, Peterson E and Slothuus R (2013) How elite partisan polarization affects public opinion formation. *American Political Science Review* **107**(1), 57–79.

Esberg J and Siegel AA (2023) How exile shapes online opposition: Evidence from Venezuela. *American Political Science Review* **117**(4), 1361–1378.

Goet ND (2019) Measuring polarization with text analysis: Evidence from the UK house of commons, 1811–2015. *Political Analysis* **27**(4), 518–539.

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

Gururangan S, Marasović A, Swayamdipta S, Lo K, Beltagy I, Downey D and Smith NA (2020) Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 8342–8360.

Howard J and Ruder S (2018) Universal language model fine-tuning for text classification. arXiv. https://doi.org/10.48550/arXiv.1801.06146.

Huszár F, Ktena SI, O'brien C, Belli L, Schlaikjer A and Hardt M (2022) Algorithmic amplification of politics on twitter. *Proceedings of the National Academy of Sciences* **119**(1), e2025334119.

Iyengar S, Lelkes Y, Levendusky M, Malhotra N and Westwood SJ (2019) The origins and consequences of affective polarization in the United States. *Annual Review of Political Science* **22**(1), 129–146.

Iyengar S and Westwood SJ (2015) Fear and loathing across party lines: New evidence on group polarization. *American Journal of Political Science* **59**(3), 690–707.

Jacobs AM, Matthews JS, Hicks T and Merkley E (2021) Whose news? Class-biased economic reporting in the United States. *American Political Science Review* **115**(3), 1016–1033.

Justin G and Stewart BM (2013) Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis* **21**(3), 267–297.

Karl F and Scherp A (2023) Transformers are short-text classifiers. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Cham: Springer Nature Switzerland, pp. 103–122.

Laurer M, van Atteveldt W, Casas A and Welbers K (2024) Less annotating, more classifying: Addressing the data scarcity issue of supervised machine learning with deep transfer learning and BERT-NLI. *Political Analysis* **32**(1), 84–100.

Lee J, Yoon W, Kim S, Kim D, Kim S, So CH and Kang J (2020) BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **36**(4), 1234–1240.

Licht H, Abou-chadi T, Barberá P and Hua W (2025) Measuring and understanding parties' Anti-Elite strategies. *The Journal of Politics* **87**(1), 275–290.

Lin G and Lucas C (2024) An introduction to neural networks for the social sciences. In Box-steffensmeier JM, Christenson DP and Sinclair-chapman V (eds.), *Oxford Handbook of Engaged Methodological Pluralism in Political Science (Vol 1)*. Oxford University Press.

Liu S and Ritter A (2023) Do conll-2003 named entity taggers still work well in 2023?. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, (Volume 1: Long Papers), pp. 8254–8271.

Liu Y, Ott M, Goyal N Du J Joshi M Chen D Levy O Lewis M Zettlemoyer L Stoyanov V (2019) Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Lucas C, Nielsen RA, Roberts ME, Stewart BM, Storer A and Tingley D (2015) Computer-assisted text analysis for comparative politics. *Political Analysis* **23**(2), 254–277.

Massaro TM and Stryker R (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 A and Farkas J (2021) Racism, hate speech, and social media: A systematic review and critique. *Television & New Media* **22**(2), 205–224.

Merchant A, Rahimtoroghi E, Pavlick E and Tenney I (2020) What happens to BERT embeddings during fine-tuning?. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 33–44.

Mikolov T, Chen K, Corrado G and Dean J (2013) Efficient Estimation of Word Representations in Vector Space arXiv preprint arXiv:1301.3781.

Milliff A (2024) Making sense, making choices: How civilians choose survival strategies during violence. *American Political Science Review* **118**(3), 1379–1397.

Munger K (2017) Tweetment effects on the tweeted: Experimentally reducing racist harassment. *Political Behavior* **39**(3), 629–649.

Osmundsen M, Bor A, Vahlstrup PB, Bechmann A and Petersen MB (2021) Partisan polarization is the primary psychological motivation behind political fake news sharing on twitter. *American Political Science Review* **115**(3), 999–1015.

Osnabrügge M, Hobolt SB and Rodon T (2021) Playing to the gallery: Emotive rhetoric in parliaments. *American Political Science Review* **115**(3), 885–899.

Parekh B (2012) Is there a case for banning hate speech? In *The Content and Context of Hate Speech: Rethinking Regulation and Responses*. Cambridge University Press.

Park HH, Vyas Y and Shah K (2022) Efficient classification of long documents using transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, (Volume 2: Short Papers), pp. 702–709.

Park JY and Montgomery J (2025) From Text to Measure: Creating Trustworthy Measures Using Supervised Machine Learning 1–17.

Poletto F, Basile V, Sanguinetti M, Bosco C and Patti V (2021) Resources and benchmark corpora for hate speech detection: A systematic review. *Language Resources and Evaluation* **55**(2), 477–523.

Rheault L, Rayment E and Musulan A (2019) Politicians in the line of fire: Incivility and the treatment of women on social media. *Research & Politics* **6**(1). https://doi.org/10.1177/2053168018816228

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press

Roberts ME (2016) Introduction to the virtual issue: Recent innovations in text analysis for social science. *Political Analysis* **24**(V10), 1–5.

Roberts ME, Stewart BM and Tingley D (2019) Stm: An R package for structural topic models. *Journal of Statistical Software* **91**(October), 1–40.

Rogowski JC and Sutherland JL (2016) How ideology fuels affective polarization. *Political Behavior* **38**(2), 485–508.

Ruder S, Peters ME, Swayamdipta S and Wolf T (2019) Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pp. 15–18.

Sellars A (2016) Defining Hate Speech. Berkman Klein Center Research Publication 2016–20, 16–48.

Shawky PE, ElKaffas SM and Guirguis SK (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 AA and Badaan V (2020) #No2Sectarianism: Experimental approaches to reducing sectarian hate speech online. *American Political Science Review* **114**(3), 837–855.

Steve R, Van Bavel JJ and van der Linden S (2021) Out-group animosity drives engagement on social media. *Proceedings of the National Academy of Sciences* **118**(26), e2024292118.

Sun C, Qiu X, Yige X and Huang X (2019) How to fine-tune BERT for text classification? In Sun M, Huang X, Ji H, Liu Z and Liu Y (eds.), *In China national conference on Chinese Computational Linguistics* Cham: Springer International Publishing 194–206.

Sundararajan M, Taly A and Yan Q (2017) Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328.

Tamara S, Harvey A, Jordon T, Srauy S and Miltner K (n.d.) *Histories of hating." Social Media+ Society* **1**(2).

Timoneda JC and Vallejo Vera S (2025) BERT, RoBERTa or DeBERTa? Comparing performance across transformers models in political science text. *The Journal of Politics* **87**(1), 347–364.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. *Advances in Neural Information Processing Systems* **30**.

Vries ED, Schoonvelde M and Schumacher G (2018) No longer lost in translation: Evidence that google translate works for comparative bag-of-words text applications. *Political Analysis* **26**(4), 417–430.

Wang Y (2024) On finetuning large language models. *Political Analysis* **32**(3), 379–383.

Wang Y (2023b) Topic classification for political texts with pretrained language models. *Political Analysis* **31**(4), 662–668.

Wasow O (2020) Agenda seeding: How 1960s black protests moved elites, public opinion and voting. *American Political Science Review* **114**(3), 638–659.

White AD (2021) Deep learning for molecules and materials. *Living Journal of Computational Molecular Science, 12. Attention Layers* **3**(1), 1499–99.

Widmann T (2024) Do politicians appeal to discrete emotions? The effect of wind turbine construction on elite discourse. *The Journal of Politics* **87**(1), 335–346.

Yamada I, Asai A, Shindo H, Takeda H and Matsumoto Y (2020) LUKE: Deep contextualized entity representations with entity-aware self-attention. arXiv. https://doi.org/10.48550/arXiv.2010.01057

Zhuang F, Zhiyuan Q, Duan K, Xi D, Zhu Y, Zhu H, Xiong H and He Q (2021) A comprehensive survey on transfer learning. *Proceedings of the IEEE* **109**(1), 43–76.

Cite this article: Sui CY (2026) High accuracy with low costs: the pretrain-finetune paradigm for classification with transformer-based language models. *Political Science Research and Methods*, 1–27. https://doi.org/10.1017/psrm.2026.10120

https://doi.org/10.1017/psrm.2026.10120 Published online by Cambridge University Press