BERT, RoBERTa or DeBERTa? Comparing Performance Across Transformer Models in Political Science Text

Joan C. Timoneda, Purdue University

Sebastián Vallejo Vera, University of Western Ontario

Conditionally Accepted for Publication at *Journal of Politics*

Abstract

Transformer models such as BERT, RoBERTa, and DeBERTa have revolutionized the field of Natural Language Processing in recent years with substantial improvements in the contextual understanding of text. While political scientists have begun adopting these models, their performance differences are not well understood, especially in cross-lingual applications. This article introduces Transformer models, compares their performance using three different text-as-data political science projects, and shows how to fine-tune them to fit the specific needs of the researcher. We find that RoBERTa and DeBERTa greatly outperform BERT in certain circumstances, and that further training boosts performance in specialized text. In cross-lingual applications, XLM-RoBERTa significantly outperforms both multilingual BERT and multilingual DeBERTa.

One of the most abundant sources of data available to social and political scientists today is text. The rise of social media and open access to Twitter data partly explain this phenomenon, as does recent progress in text digitization techniques that have brought books and manuscripts to digital life that were long hidden away in archives and libraries. Text can help us answer substantive questions in political science on topics as diverse as political campaigns (Hobbs and Lajevardi, 2019; McGregor, 2020), political polarization and radicalization (Medzihorsky, Littvay and Jenne, 2014), media studies (Matalon et al., 2021), public opinion (González-Bailón and Paltoglou, 2015), Supreme Court decisions (Strother, 2017), gender and politics (Gleason, 2020), and many others. As the availability of text grows, so does the need for computer-based text analysis techniques to supplement those done by humans.

Recent advances in Natural Language Processing (NLP) have spearheaded a text-as-data revolution. In particular, the development of a novel type of deep learning architecture, Transformers, in 2017 has allowed language models such as BERT, RoBERTa, DeBERTa to understand, classify, and artificially generate text with groundbreaking levels of contextual accuracy (Tunstall, von Werra and Wolf, 2022; Liu et al., 2019; He et al., 2020). Political scientists have begun using Transformer-based models, but there is still a need for clarity around the performance differences across BERT, RoBERTa and DeBERTa models when applied to political science texts. This is especially true in the case of multilingual classification problems, where multiple models such as mBERT, XLM-RoBERTa and mDeBERTa exist but there is little consensus over which one performs best or by how much. Lastly, we illustrate how researchers can fine-tune an off-the-shelf Transformer model to apply it to specialized text. First, they can further train the model with new unlabelled

text data to suit specific tasks, thus improving contextual understanding of specialized language. Second, they can adapt the model to any specific application with labelled training data. Combining both of these strategies, we argue, yields substantial gains in performance.

In this article, we aim to introduce Transformer models to a broader audience, comparing their performance in English and multilingual models and showing how researchers can further train them on specialized text.[1] Beyond these contributions, we provide detailed evidence for what we believe are the practical advantages of these models for political and social scientists: (1) lower costs to generate data from text through accurate classifiers, (2) potential for large-N analysis for otherwise small-N projects, and (3) the ability to generate new theoretical questions and empirical tests that would not be possible without these models. We highlight their importance through three existing political science projects that use text-as-data for sentence or text classification.[2]

The culmination is three main findings that provide evidence for determining which model to use based on the needs of the researcher. First, RoBERTa is the model that generally offers the greater balance between performance and computational cost. It consistently outperforms BERT by what we consider is a substantial margin. DeBERTa, on the other hand, has similar performance to RoBERTa but uses about twice as much computational power. Second, creating custom-made BERT, RoBERTa or DeBERTa models through further training yields substantial improvements in classifier accuracy. Third, as expected,

[1]Further training is the fine-tuning process whereby we provide a model with new words (tokens) and new text on which to learn how the new words are used in context.

[2]Our goal is in no way to assess or issue any judgments on any of these projects but rather to highlight the alternatives that Transformers-based models offer, especially when compared to traditional human coding approaches and other widely used NLP classification techniques.

all Transformers-based models improve on the performance of other widely used machine learning approaches.[3] With performance gains and the fine-tuning flexibility of Transformers, we believe a growing number of political science projects using text data can benefit from a big data approach.

Indeed, early work applying Transformer-based models showcases the potential of BERT and RoBERTa in political science. Abercrombie et al. (2019) and Abercrombie and Batista-Navarro (2022) employ BERT to detect policy preferences (up to 34 topics) from members of Parliament using debate motions. Similarly, Alemán, Micozzi and Vallejo Vera (2022) use XLM-RoBERTa to classify legislative speeches by topic. Bonikowski, Luo and Stuhler (2022) apply RoBERTa to identify frames in U.S. presidential campaigns. Since the focus of these papers is not solely methodological, the selection of their model of choice is unclear (and beyond their scope). Questions also remain around what additional steps scholars could follow to further improve the performance of their models. Our paper provides a systematic overview of Transformer-based models and how to apply them in different classification tasks. While not comprehensively, we show how these models can be fine-tuned for various tasks, including multilingual classification. In comparing the performance of the models, we detail the advantages of using different types of models in examples familiar to social scientists (e.g., performance and computational cost, additional training). Ultimately, we offer an approachable explanation of Transformer models with the aim of making them more accessible to a larger number of applied researchers.

The article is structured as follows. We first briefly introduce the Transformers family and show how it differs from earlier NLP approaches adopted in the social sciences. We then

[3]We compare the performance of Transformer models with SVM and Bi-LSTM recurrent neural networks.

introduce our main arguments in favor of using Transformers-based models in political and social sciences. Three existing text-as-data projects are then used to illustrate our arguments and show the full practical potential of BERT-based models.[4] We conclude with our findings, recommendations, and links to resources for researchers to use these models in their projects.

NLP in the Social Sciences

The field of NLP is in the midst of a major revolution. In the past decade, scholars have gone from computing word frequencies and generating broad descriptive assessments to building deep learning systems that understand the contextual meaning of words and sentences (see Grimmer, Roberts and Stewart, 2022; Tunstall, von Werra and Wolf, 2022). Two factors explain the NLP boom. First, computational power has multiplied recently, providing the necessary technology for computationally intensive text analysis. The second reason is the increased availability of accurate NLP models. Companies like Google and Facebook have invested large amounts of resources to improve text-to-speech and translation technology to detect and weed out (however halfheartedly) certain types of hate speech and disinformation from their platforms. Translation tasks have become increasingly important in an interconnected world. To respond to these needs, new Transformers text models emerged, bringing generational leaps in accuracy and performance.

In political science, machine learning models have become increasingly popular to tackle tasks such as supervised and unsupervised text classification, named entity recognition, sentiment analysis, text similarity, among others. Topic models in particular have been

[4]We will focus solely on supervised sentence classification tasks to make the article tractable, even though the models we introduce have broader applications.

widely used to cluster texts into groups without labelled training data (Grimmer, 2010; Roberts, Stewart and Tingley, 2016; Catalinac, 2016; King, Pan and Roberts, 2013). These models have worked well with newspaper articles and official statements from political and social elites, but they are less accurate when the text is informal or short (Grimmer, Roberts and Stewart, 2022). Supervised text classification (the object of this article) uses a labelled training set to train a model that can accurately classify unseen text in the same categories as the training set (Barberá et al., 2021; Pan and Chen, 2018). Event extraction from text has also begun using machine learning approaches despite relying on dictionary approaches for years (Ward et al., 2013; Beieler et al., 2016), and named entity recognition has registered improvements in recent years with the growth of libraries such as NLTK and spaCy.[5]

In supervised text classification, common machine learning approaches include Support Vector Machines (SVM) and Logistic Regression (LR) classifiers. These models use a bag-of-words approach, with an off-the-shelf tokenizer like NLTK to predict the category of a given text.[6] A tokenizer breaks down sentences into tokens, or word chunks that the model can understand, which are then converted into numeric vectors using various strategies.[7] These vectors enable the model to capture how important words are in a given text sequence. Machine learning models then use these vectorized representations of text to produce classification predictions based on some outcome of interest. While simple, these

[5]The Natural Language Toolkit, or NLTK, is a suite of libraries used to perform many NLP tasks. Text also must be pre-processed first, removing stopwords ('the', 'and', and so on) and special characters for greater accuracy.

[6]More recent models such as Bi-LSTMs (introduced below) use conditional bag-of-words approaches that preserve context to a much greater degree.

[7]One of the most popular is NLTK's (`https://www.nltk.org`), which is often combined with 'term frequency–inverse document frequency', or 'TF–IDF' vectorizer, for improved performance.

models can yield good accuracy scores in tasks where context is not particularly important for performance.

Improving on word vectorization, one of the first revolutionary advances in NLP was Word2Vec word embeddings, developed in 2013 by Google (Mikolov et al., 2017). Word embeddings are mathematical representations of words in a vector space, where vectors closer to each other represent words that are more similar in meaning. A commonly used English-language Word2Vec model trained on a large set of Google News text contains 3 million word embeddings, each of which is a numeric vector of size 1 x 300.[8] Other widely used word embeddings are Stanford CoreNLP's *GloVe* and Facebook's *fastText* embeddings (Pennington, Socher and Manning, 2014; Bojanowski et al., 2017).[9] Word2Vec and GloVe embeddings are often paired with Recurrent and Convolutional Neural Network architectures for text classification (RNN and CNN).[10] One of the most commonly used are Long Short-Term Memory (LSTM) networks, a type of RNN that takes in *sequential* input but keeps important information from further back in the sentence that can be useful to understand new words (Chang and Masterson, 2020). Bidirectional LSTM networks, or Bi-LSTMs, process sequential information both forward and backward, further improving contextual understanding. We discuss these alternative models in detail in our first application when we describe the baselines for comparing our main Transformers models. The problem with these approaches, however, is that word vectors are static, meaning that each word has a corresponding fixed mathematical vector after training. In Word2Vec,

[8] As a result, the computational demands of using these large Word2Vec embeddings increased exponentially. However, nowadays they can be handled easily by most mid- and top-end CPUs.

[9] See Rodriguez and Spirling (2022) for embeddings created specifically for political science applications.

[10] Neural networks are deep learning algorithms that recognize patterns in data through different layers and nodes. Nodes process information via weights and each layer produces a new and more accurate representation of the input (see Albawi, Mohammed and Al-Zawi, 2017).

GloVE and fastText, the numeric vector for the word 'bear' is the same in 'grizzly bear', 'teddy bear', 'bear fruit' or 'bear a loss'. This is where the Transformers deep learning architecture innovates: it can dynamically capture the different meaning of words based on context. In the example above, a Transformers-based model would produce four different word embeddings for 'bear', one for each specific use of the word. While these approaches are computationally much more expensive, their benefits greatly outweigh the drawbacks. We now introduce these models and their architecture.

The Intuition Behind Transformers Networks

NLP models based on the Transformers deep learning architecture have yielded unparalleled accuracy in sentence classification, question answering, language translation, and other tasks. Transformers are the engines that make BERT, RoBERTa, XLM-RoBERTa, and DeBERTa run.[11] What exactly does the Transformers neural network architecture do to yield these results? They generate dynamic word embeddings that change for every word depending on the context. That is, no word embedding for a given word will be the same across different texts. Each embedding incorporates the word's position in a sentence and its relationship with words that come before and after. The embedding for the word 'bear' is different in sentences containing 'grizzly bear', 'teddy bear', 'bear fruit' or 'bear a loss'. The dynamic nature of these embeddings allows for greater contextual understanding as the context largely determines the word embedding itself. This is the key difference between the Transformers deep learning architecture and other well-known deep learning architectures such as RNNs or CNNs, which mainly use static embeddings.

[11] Other models, such as XLM-Net and GPT-3, also use the Transformers architecture.

To accomplish this, Transformers networks use a mechanism called *self-attention*. This process allows the network to take an entire text as input and process its words *all at once* rather than sequentially, as RNNs and previous approaches in NLP do (Vaswani et al., 2017; Tunstall, von Werra and Wolf, 2022). However, processing all words at once is computationally intensive, and the Transformers architecture is complex. Let us begin by describing its encoder layers. Encoder layers are the different layers in neural networks, including Transformers, that perform computations. Each encoder is identical and includes both the self-attention mechanism and a feed-forward neural network (FFNN). Information passes from one layer to the next becoming progressively simpler and more useful by reducing the amount of information and distilling its essence until it can produce the most accurate output for a specific application (say a classification or translation task) . Figure 1 describes how the encoder of a Transformers model works. The input ("I love this city") passes through three encoder layers, starting with each token's initial representation or embeddings (x_1 through x_4). These embeddings change as they move from one layer to the next, until the embeddings reach their final form (f_1 to f_4). The diagram in Figure 1 shows a three-layer encoder ($encoder_1$ to $enconder_3$), but BERT and RoBERTa base models have 12 layers.

An element within each encoder layer that helps implement the self-attention mechanism is the *attention head*. Each attention head uses multiple matrices to compute the mathematical relationship between all words in a sentence.[12] In the example of Figure 1,

[12]More technically, each input consists of 'queries' and 'keys' matrices. In the sentence 'Mary likes books', to give meaning to the word 'Mary' (query) we look at the whole sentence, a find the word that is most related to it, in this case, 'likes' (key). This process happens for each word in the sentences. Similarly, the self-attention mechanism will estimate the distance (dot product) between the vector representation of every query to the already provide vector representation of every key. Note that, while the tokens can be the same, the representations will be different (as they they come from previous layers). Mathematically, we

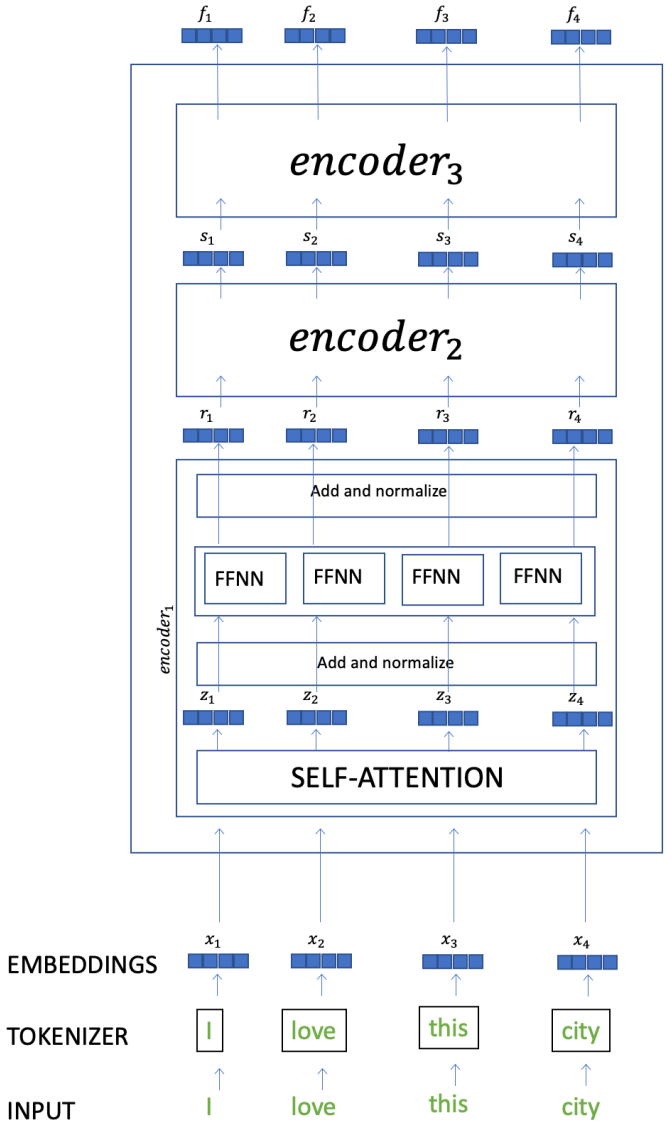

Figure 1: Diagram of a three-stack encoder of a Transformers model. Input text is tokenized and given an initial embedding (vectorized representation) simplified in our figure as x_1 through x_4. The initial embeddings are transformed as they enter the first $encoder_1$. In it, the self-attention mechanism updates the embeddings (z_1 through z_4), which are then passed through a feed-forward neural network. They exit the encoder as a more accurate set of embeddings (r_1 through r_4). The process is repeated for all encoders in the neural network. For example, pre-trained BERT-base models use 12 encoder layers.

the self-attention mechanism allows the model to associate "this" with "city" (rather than

obtain the dot product of K(eys) and Q(ueries) such that $Attention(Q, K, V) = softmax(QK^T)/\sqrt{d_k})V$ where d is the dimension of K and used as an scaling factor. See Appendix F for further details.

with "I"). Thus, the new representation of "this" (z_3 and eventually r_3 in Figure 1) will have some information from the other words in the batch, thus imbuing the embedding of "this" with *contextual information* from all surrounding tokens. $encoder_1$ will produce one output matrix (composed of r_1 to r_4) that moves on to the following encoder layer ($encoder_2$) and attention head, beginning the process again and repeating it as many times as there are attention heads –12 in total for base BERT and RoBERTa models (see Ravichandiran (2021) and Tunstall, von Werra and Wolf (2022) for more detailed information on these calculations). The last encoder outputs a final representation (f_1 to f_4), which can then be decoded to generate a specific outcome such as a translation into another language or a classification into a given category (Vaswani et al., 2017; Tunstall, von Werra and Wolf, 2022; Ravichandiran, 2021). Usually, the more layers and attention heads there are, the more precise the final representation is.

As shown in Figure 1, the vector representation of each token changes as it progresses through the encoder layers of the Transformers network (Ravichandiran, 2021). For instance, the numeric representation for 'bear' will be different if it is followed by 'fruit' or if it is preceded by 'teddy'. Each attention head will output an increasingly accurate word embedding for the word 'bear' that reflects its meaning in the sentence. In 'the child loved the teddy bear because it was soft', the Transformers architecture will use all words in the sentence and give special weight to those it thinks are related to 'bear' –'child', 'teddy', and 'soft'– to produce the appropriate embedding for the word 'bear', one which captures the idea of a teddy bear rather than a grizzly bear. Since embeddings are numerical representations of words (vectors that follow algebraic rules), the distance between two embeddings is an approximation of the relation between words –e.g., vectors that are closer

together are more similar than vectors that are farther apart. For example, in the sentences: 'The kid had to bear the loss of misplacing his teddy bear and his doll,' the cosine similarity between (teddy) bear and doll should be higher than the cosine similarity between (teddy) bear and bear (the loss). In Appendix H we provide code and results using RoBERTa to show that this is the case. Also, in Appendix F, we provide a more in-depth and technical discussion of Transformers networks.

Lastly, note that we emphasize the difference between Transformer models and Bi-LSTMs in terms of these dynamic word embeddings, but key differences across these models warrant further discussion. First, Transformers process words all at once, using positional embeddings to understand word order.[13] This differs from LSTMs, which have to process tokens sequentially. The second main difference is the self-attention mechanism we described above, which produces contextual word embeddings to fully understand words in context. LSTMs, on the other hand, use static word embeddings. Bi-LSTMs can maintain the meaning of relevant words further back or forward (they are bidirectional) in the sequence to improve predictions, but the embeddings do not change dynamically with context. Due to their capacity to understand context better than traditional LSTMs or CNNs, we expect Bi-LSTMs combined with the most recent state-of-the-art word embeddings (GloVe) to provide the best benchmark on which to compare Transformer models.

The Transformers Family: BERT, RoBERTa, and DeBERTa

Google AI's BERT and Facebook AI's RoBERTa and XLM-RoBERTa are the encoders of a

[13]Positional embeddings are an additional matrix where each token is given a number to represent its position within a text.

Transformers model. After all, BERT stands for 'Bidirectional Encoder Representations from Transformers'.[14] Google's BERT encoder consists of 12 encoder layers and 12 attention heads in its BERT-base configuration, and 24 encoder layers and 16 attention heads in its BERT-large configuration (Ravichandiran, 2021).[15] Also, note the word 'bidirectional' in BERT, which points to its ability to read text forwards *and* backwards, relating each word to all words in a sentence.

BERT and related models leverage the Transformers architecture we just described, but that in itself does not 'train' the models. For training, these models need (1) *information* as well as a way to (2) *learn* from that information. Step (1) is relatively simple, if computationally intensive. The creators of BERT used 11,038 books from the Toronto BookCorpus and all of English Wikipedia to train BERT –a total of 16GB worth of text (Devlin et al., 2018). Facebook AI's RoBERTa, on the other hand, used the same data as BERT and added more data from Common Crawl (CC-News), Open WebText, and a subset of Common Crawl named Stories,[16] for a total of 160GB of text, or ten times more data (Liu et al., 2019). Cross-lingual RoBERTa, XLM-R, was trained using Wikipedia for all languages and data from Common Crawl (Conneau et al., 2019). While it is true that models have increasingly diversified their sources of training data, the over-reliance on Wikipedia and web data has implications for specialized applications, as the language from these websites does not include the types of words and text required for high performance

[14]RoBERTa stands for 'Robustly optimized BERT approach' and XLM-RoBERTa stands for "Cross-lingual RoBERTa."

[15]Each layer of a BERT-base encoder outputs word vectors of length 768, while the BERT-large model outputs word vectors of length 1,024 (the same applies to the base and large versions of RoBERTa and XLM-R). Longer vectors contain a more accurate representation of a word, but also require more space and computational power.

[16]Common Crawl is a repository of historical websites.

in specialized tasks. Our third application below addresses this issue and provides a solution for researchers to improve task-specific performance.

Step (2) is a bit more complex. How do BERT and similar models learn how words relate to each other using large amounts of text? The method they all share for learning is called Masked Language Modeling (MLM).[17] MLM masks about 15% of tokens in a text corpus. Masking means replacing the actual token with <MASK>, and then using the full power of the self-attention mechanism from Transformers to *predict* the masked words. For instance, in the sentence 'I love visiting the windy city, <MASK>, the cultural and commercial capital of the Midwest', BERT and RoBERTa will use the information before and after <MASK> to predict 'Chicago' While 'windy city' may provide a clue, the fact that the city is an important Midwest metropolis is key in predicting the word correctly. MLM takes advantage of the two most innovative and powerful features of Transformers-based models: bidirectionality and self-attention. To predict 'Chicago' BERT and RoBERTa use words before and after the <MASK>, not only words before. They use all relevant information in the sentence ('city', 'windy', 'midwest', 'metropolis', and 'commercial capital') to come up with a probability for the most likely candidate word to replace the <MASK>. During training, BERT and similar models use MLM to predict 15% of all words. Through MLM, these models become highly accurate at word prediction, which means that in large numbers, they can understand all words in a text and how these words relate to one another. In sum, while Transformers is the neural network architecture that produces the most accurate representations through the self-attention mechanism, MLM is what allows

[17]BERT also uses next sentence prediction as a training method, but MLM remains the most common training method across the different models.

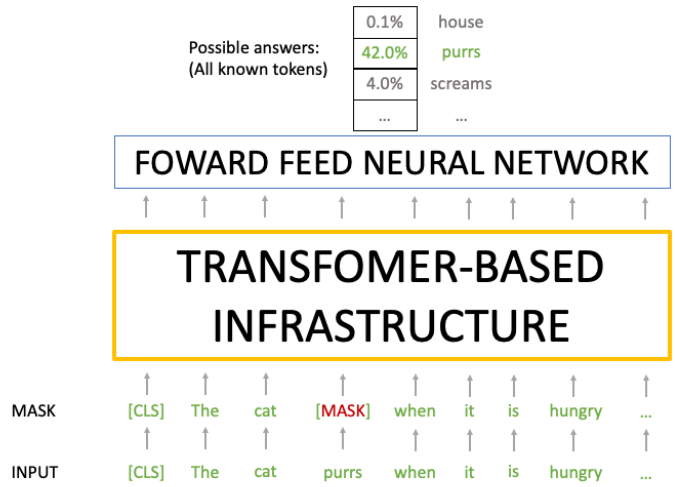

Figure 2: Representation of the Masked Language Modeling. Transformer models randomly mask 15% of tokens. After running the corpus through the Transformers and neural-network architecture, it asks the model to predict the masked word. Transformer models then calculate loss and the required gradient changes to optimize the model's weights and obtain better representations.

BERT, RoBERTa and other models to learn about a text.[18] Figure 2 provides a graphical description of MLM.

We hightlight four key implications of MLM. First is its ability to understand context well. As explained above, by masking words in text and making the model predict the masked words, we have achieved substantial improvements in performance. The second is the inductive and social biases that emerge from using MLM. Inductive biases help Transformer models using MLM to learn sentence structures, which can lead to downstream performance gains (Zhang and Hashimoto, 2021). However, they are also known to "encode worrying levels of social biases", especially around gender and race (Kaneko and Bollegala, 2022). Third is the lack of clarity that remains around the share of words in the training

[18]BERT models use MLM to estimate the probability distribution for all tokens to replace the <MASK> object. Once we uncover the masked object, BERT can estimate loss, the difference between the probability distributions for each output token and the true labels. In the next iteration, BERT will correct its prediction accordingly. This process appears in more that one stage of the training process. For example, during fine-tuning (explained below), we can use MLM to improve our predictions as well.

data that should be masked. The creators of Transformer models masked 15% of words in the unlabelled training data, which has become common practice. Others have begun to question whether masking more words can yield better results (Wettig et al., 2022). This naturally leads to the fourth implication, namely, that authors require between 0.5 and 1 gigabyte of additional, specialized unlabelled data to make noticeable performance gains.[19] We address this issue further in application 3.

Note also that BERT, RoBERTa and XLM-RoBERTa all need to first tokenize a sentence before they can compute an output. That is, they need to break the text into words and word chunks that have meaning according to the training process. Since BERT was trained on less data than RoBERTa, it has 30,522 unique vocabulary elements, while RoBERTa has 50,265. The cross-lingual model XLM-RoBERTa, on the other hand, has the largest number of unique vocabulary elements at 250,002 (Conneau et al., 2019). These unique vocabulary elements can handle out-of-vocabulary words by concatenating different word chunks.[20] For instance, the word 'training' would be tokenized as "train, ##ing", with the double hashtag indicating that 'ing' is a subword token that follows the token 'train'. This approach to sub-word tokenization has proven to be accurate in handling out-of-vocabulary words (Ravichandiran, 2021; Tunstall, von Werra and Wolf, 2022).

[19] It could be that by masking a higher percentage of words, MLM would perform better at predicting newly itroduced vocabulary elements, as they would be masked more often. Future research should address this issue.

[20] Transformers models use different approaches for subword tokenization. BERT uses byte-pair encoding, which reduces words to the character level and creates character groups based on unique vocabulary elements according to their frequency in the training data. RoBERTa uses Byte-level Pair Encoding, which converts characters to bytes (the letter 'b' is byte 62, for instance) and then combines the bytes into groups also according to their frequency in the training data. Generally speaking, if a group of characters or bytes exists as a token in the vocabulary, it is used as a token. If not, it is further broken down until a vocabulary token matches it. Because characters are converted to bytes, this method is more helpful in multilingual contexts. These methods differ from techniques such as lemmatization or stemming in that the full word and all the information are kept in subword tokenization, and it is up to the Transformers matrices to calculate how important each subword token is in relation to all other tokens in the sentence.

So far we have discussed BERT, RoBERTa, and XLM-RoBERTa, but not DeBERTa. Decoding-enhanced BERT with disentangled attention (DeBERTa) model makes two technical innovations on BERT and RoBERTa (He et al., 2020). One is that it separates the word's content from its position in a sentence and thus computes the final embedding from the Transformer in a way that makes the representation more accurate. The second is that it refines the final decoder layer to achieve better fine-tuning (He et al., 2020).[21] DeBERTa was trained on 78GB of data and reported improvements between 0.9 and 3.6 percentage points over RoBERTa (large) even when using half of the training data (He et al., 2020). The downside of DeBERTa is that it uses *double* the GPU RAM of RoBERTa (large) and over three times that of BERT (large).[22] In the applications section we compare DeBERTa performance to RoBERTa and BERT to see whether these reported performance benefits outweigh the computational costs.

Fine-tuning a Transformers-based model

Where Transformers-based models' abilities shine brightest is in their general applications, a process known as *fine-tuning*. During fine-tuning we use a pretrained Transformers-based model and modify its final encoder layer to suit our particular task. Thus, we can harness the great knowledge that BERT and RoBERTa already have about words and text to produce highly accurate sentence classification of campaign slogans, judicial decisions, racism on Twitter, and myriad other important political science research applications. The following subsection details the general procedure to fine-tune and further train

[21] We will not go into depth about the details of DeBERTa's innovations for want of space and an emphasis on clarity and applications. Please refer to He et al. (2020) for more information.

[22] This is, to the authors' knowledge, the first paper in the discipline to report these findings on computational cost.

Transformer models.

In general, there are two ways to apply these models: *standard fine-tuning* and *further training*. Standard fine-tuning is the most common application and refers to using an already trained or off-the-shelf Transformers-based model like BERT and applying it to a specific task using manually labelled training data. Say we are using BERT and trying to classify a text as positive or negative (sentiment). We first need a labeled training set, usually a sample of text manually labeled 'positive' or 'negative'. The size necessary for the training set varies, but one of the advantages of BERT, RoBERTa, and similar models is that they can be fine-tuned using relatively small amounts of training data.[23] We then tokenize all the labeled text using BERT's tokenizer and divide the training data into training and test sets.[24] We set the appropriate model hyperparameters[25] and apply BERT to our specific classification problem. We use 10-fold cross-validation to fully evaluate the capacity of the model to generalize to unseen data.[26] Once the model has been cross-validated, we train the final version using the full set of training data without splits and obtain our final classified dataset.[27]

[23]No hard rule exists on the amount of labeled text per category required to train a Transformers model, but it is ideal to have between 200 and 500 observations depending on the task. Transformer-based models perform well with small training sets (Khan et al., 2021).

[24]The training set is a subset of the training data that the model will use to "learn" how to classify the task at hand. The model then uses the test set to evaluate its prediction accuracy on unseen data. Common practice is to split the data into 80% train and 20% test sets in five-fold cross-validation and 90% and 10% in ten-fold cross-validation.

[25]Learning rate, number of epochs, batch size, optimizer, and number of warm-up steps.

[26]In all our applications below, we perform 10-fold CV on the full dataset on a set of 90-10 train/test splits. We do not hold out an extra 10 or 20% of the data for a true out of sample test for two reasons. First, we report the test averages for the full CV run as our final test scores, and we do not select a specific 'best model' from the CV run. The second, and more constraining, reason is that we do not have large amounts of data for applications 1 and 2, and withholding data from training affects performance noticeably. We further explain cross-validation and how to set hyperparameters in Appendix B.

[27]Note the difference between the cross-validation and final fine-tuning step. The cross-validated performance metrics give us a measure of the model's performance to generalize to unseen data. The final fine-tuning step uses the full training set without splits for training and test data and the best model hyperparameters from cross-validation. See Appendix B for a more detailed discussion on cross-validation and Appendix C

The second way to fine-tune a BERT –or similar– model is through *further training* the model. To do so, researchers provide *additional* unlabelled text data to the model to improve accuracy for a specialized task. That is, we take all the knowledge BERT has and add (1) new raw text data and (2) new vocabulary elements to allow it to understand specialized text better. As a practical guide, the researcher must first gain access to a BERT, RoBERTa or DeBERTa pretrained model that they want to train further. After that, they must add to the model's tokenizer a new set of unique vocabulary elements specific to the domain of study. The choice of vocabulary should be grounded on the researcher's expertise on a topic. Third, the researcher must collect, tokenize[28] and administer new unlabelled data containing the new vocabulary elements to the base pretrained model. The fourth step is to retrain the model, which allows it to see the new vocabulary elements in context and understand them better. Lastly, the model can then be saved and applied using standard fine-tuning to build a classifier. We provide a more detailed guide with Python code snippets on how to further train a model in Appendix E. For example, in application 3 in the next section, we further train a RoBERTa-large model to better recognize text that contains the words 'covid' and 'coronavirus', two terms that did not exist in 2018 when Facebook AI trained the original RoBERTa. We use an unlabelled set of 6,076 academic abstracts and 4.8 million tweets and news headlines, all of which are about Covid-19. The idea is that performance improves when adding 'covid' and 'coronavirus' as new unique vocabulary elements to the model and training it to understand what these two new elements mean in relation to other words in different texts. Note that

for a step-by-step guide with Python code snippets on how to perform standard fine-tuning.

[28]Using the updated tokenizer with the new vocabulary elements.

further training generates a new variant of the BERT or RoBERTa model, which can then be used for standard fine-tuning as described above. We further detail how to further train a RoBERTa model in application 3.

Using Data from Transformers in Downstream Analyses

Once researchers have built and saved the final classifier, they usually apply it to a new and unseen dataset to generate larger amounts of data. It is important, however, to be cognizant of the potential measurement bias present in Transformers-generated data, as it can lead to estimation bias. We closely follow recommendations by Egami et al. (2023), who propose a method to reduce measurement bias in data generated through large language models. The key contribution of the method is in comparing human labelled data with data generated through a large language model, and using that information to estimate the extent of measurement error in the larger sample. More specifically, the authors suggest using human-coded data to produce a pseudo-outcome of the target model and estimate a correction term from the difference in outcome from the surrogate model (predicted data) and the pseudo-outcome (human-coded data). We suggest that researchers apply this methodology once they have created the full dataset for analysis. In Appendix I, we provide a more detailed illustration of Egami et al.'s (2023) methodology using an example based on Application 3 below.

In sum, by fine-tuning a Transformers-based model for our own application, we can improve *contextual* understanding and therefore task-specific performance. We can then apply the custom-built model to unseen data to generate larger amounts of labeled data to be used for analysis, correcting for any potential measurement biases present in the machine-labeled data. We now illustrate the techniques described above using three different applications relevant to political scientists.

Applications

We use three projects within or relevant to political science to compare the performance of BERT, RoBERTa and DeBERTa models in different types of text data. We selected the projects carefully to provide a wide array of potential applications within political science. The first project uses English text from Twitter and produces a binary classification of civil and incivil tweets. The second project uses text in 29 languages and classifies it into four categories. The final project uses Covid-related text in English and a binary classification of true and fake news. This final application helps illustrate the advantages of further training BERT and RoBERTa. We also provide results from two non-Transformer baseline models for reference: (1) Support Vector Machines (SVM), which uses a simpler machine learning model to classify text, and (2) a Long-Short Term Memory recurrent neural network, which is the non-Transformer state of the art model in the literature. We provide further details on these two baseline models below.

We run all of our models Python, the language of choice in computer science for the NLP models described in this article. The code uses three common machine learning and deep learning Python libraries: Transformers, Torch and Scikit-learn. Code for these li-

braries is widely available and accessible to all applied researchers. We have also made our code available on GitHub (omitted). All pretrained models used in this article are publicly available at `huggingface.co`. These include BERT-large (`bert-large-uncased`), RoBERTa-large (`roberta-large`), DeBERTa-large (`deberta-v3-large`), XLM-RoBERTa (`xlm-roberta-large`), mDeBERTa (`mdeberta-v3-base`) and mBERT (`bert-base-multilingual-uncased`). We use learning rates within the ranges suggested by the authors of these models, the weighted Adam optimizer, and 10 warm up steps.[29] We use 4 epochs for BERT, 5 epochs for RoBERTa and DeBERTa, and 6 epochs for the cross-lingual models, all with an early stopping mechanism to prevent overfitting.[30] The computational requirements for each model vary widely, and researchers should consider the perfomance-cost trade-off when deciding on which model and which platform to use.[31] We provide further details on the models in Appendix B, including how long each model takes to run and the maximum number of tokens used per application (see Table B.1). We also provide the standard deviation for the F1 scores of all the cross-validation runs in the tables below.

Lastly, while this article and the applications that follow focus primarily on the predictive performance of Transformer models, they are also useful for measurement. Indeed, we would emphasize the importance of using these models to create new and more accurate measures of complex political phenomena. For instance, Transformer models can help us generate more precise measures for Supreme Court Justices' ideal points, interest group interactions in Latin American parliaments, democratic incumbents' social media use, and a host of other phenomena where text is a primary source of data. In fact, the variables that result from these models, and which are then used in statistical models, are often their most powerful contribution.

1. Incivility on Twitter

The first project studies incivility in US state legislatures (Gervais and Morris, 2019). The authors leveraged increased Twitter activity by state legislatures (the institutions themselves have Twitter accounts) and organized parties in state legislatures between 2006 and 2018,

[29]The learning rate is the parameter that determines how quickly the model 'learns.' A low learning rate can lead to slow convergence or the model lingering in local optima. A high learning rate can often lead to lack of convergence because the model overshoots the solution. Scholars should monitor loss and accuracy gains to ensure they pick the right learning rate to maximize performance. Recommended learning rates are $3e-4, 1e-4, 5e-5$, or $3e-5$ for BERT-large-uncased (our choice: $3e-5$), $1e-5, 2e-5$, or $3e-5$ for RoBERTa-large (our choice: $3e-5$), and $5e-6, 8e-6, 9e-6$, or $1e-5$ for DeBERTa-V3-large (our choice: $1e-5$). Each researcher can then move the needle up or down to fit each specific task, testing learning rates in these ranges. For XLM-R, $5e-6$ is recommended but again, this may vary slightly between $5e-4$ and $5e-7$ in most applications. The batch size is the number of samples that will be propagated through the network in each iteration. A size of 16 is preferred over one of 8 and will produce better performance, especially with more than two labels. After that, a batch size of 32 makes the model run faster but does not meaningfully improve results and is much more computationally demanding (in terms of GPU RAM). Weighted Adam is an adaptive optimizer that helps improve convergence and generalizability(see Loshchilov and Hutter, 2017).

[30]Training ends after the first increase in validation loss when compared to training loss.

[31]In Appendix A, we include a breakdown of the computational requirements and costs associated with each model on Table A.1. In this article, we use Jupyter Notebooks that we run either on Google Colab (free GPU accelation up to 16GB of GPU RAM) or datacrunch.io for more demanding tasks. Table B.1 in Appendix B details which computing platform we use depending on the model and the number of tokens.

collecting all twitter activity in this period for all 50 US state legislatures and organized state parties. The process yielded 344,000 total tweets and the authors built a sample of 2,076 tweets, and used three research assistants to code each tweet as either civil or incivil. Examples of incivil tweets are: ".@AKIndDems Wrong. Please stop lying to the twitterverse. Interested in the truth? Read here- https://t.co/D2p8OZAjBl #akleg" and "@AKIndDems - that's not true. Don't play fast and loose."[32]

The coders agreed on the labeling 75% of the time and the average agreement across coder pairs was 78.7%. The Kappa score for inter-coder reliability is 0.591, indicating moderate agreement. The authors of the article decided to code as 'incivil' tweets where at least two coders agree. This led to a total of 534 incivil tweets (25.7%) and 1,542 civil tweets (74.3%) in our final dataset, a slightly imbalanced panel for which we will report F1 scores. The F1 score is the harmonic mean between precision and recall scores, and therefore better captures accuracy in unbalanced panels.[33] The mean length for these tweets is 19.29, resulting in an average of 42.32 tokens per tweet using the RoBERTa tokenizer and a maximum number of tokens of 147. We shuffle the full 2,076 tweet sample and use ten-fold cross-validation (CV) to test the true ability of the model to generalize to unseen data. We report the average scores from repeated (10 times) 10-fold CV for all models in Tables 1 and 2. Repeated 10-fold CV consists in performing 10-fold CV multiple times (in our case ten times), yielding an average of model performance across 100 models from 3 different sets of 10 folds. This approach provides a much more accurate estimate of true out-of-sample model performance.[34]

In Table 1 we report our first set of tests using the ideal 'type' variable where at least two coders agree.[35] We compare the performance of BERT, RoBERTa and DeBERTa and provide two other non-Transformer baseline models that have been widely used in the literature. First is a machine learning approach using SVM with an NLTK English language tokenizer and a TF-IDF vectorizer.[36] NLTK is a powerful NLP library in Python that helps us convert text into tokens that models can understand (Loper and Bird, 2002). The TF-IDF vectorizer is a commonly used tool that penalizes common words and gives particular importance to rare but more meaningful words, which helps the model to understand text better. This model

[32]The authors set four criteria for incivil tweets: 1) name-calling, mockery, sarcasm, and character assassination; 2) spin and misrepresentative exaggeration; 3) emotionality/digital stridency; 4) conspiracy theory.

[33]This is because it considers both how well the model has identified true positives as opposed to generating false positives (precision), and the model's ability to identify true positives as opposed to generating false negatives (recall). Some models may generate a lot of true positives and very few false positives (high precision), but they may also generate a lot of false negatives (low recall) –or vice versa. This is a larger problem in unbalanced panels. When only 5% of observations are 1, models can produce more false negatives because the 0 category dominates, leading to low recall, but they may also have high levels of accuracy because most observations will be classified correctly. By using the F1 score instead of the accuracy score, we get a much more representative picture of the model's performance.

[34]We use a learning rate of 1e-05 (Deberta) and 3e-05 (Bert and Roberta), a batch size of 32, and train the models over 4, 5 or 6 epochs. Monitoring validation loss versus training loss is a standard approach to determining whether the model is overfitting. When the validation loss exceeds the training loss, the model is trying to increase accuracy on the training data *at the expense* of generalizability. This is reflected in worse performance on the validation set.

[35]This variable tends to produce the best performance across all models.

[36]We apply the TF-IDF vectorizer, which weighs token frequencies once document frequencies are also considered, to the SVM models to maximize their performance as they do not use word embeddings. The TF-IDF vectorizer we use is from the scikit-learn Python library.

shows how well machine learning models can classify text based on advanced calculations of word frequencies and word importance, but it should perform worse than the rest of the models, given its relative simplicity. The second model is a Bidirectional Long Short-Term Memory (Bi-LSTM) recurrent neural network, which we pair with GloVe word embeddings (see Chang and Masterson, 2020).

Table 1: Model performance by category (main 'type' variable; 10x repeated CV scores)

Model	**Civil**				**Incivil**				**Macro-Average**			
	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}
ML - SVM	0.787	0.832	0.809	0.005	0.420	0.350	0.379	0.012	0.591	0.604	0.594	0.007
Bi-LSTM - GloVe	0.806	0.841	0.820	0.006	0.479	0.409	0.423	0.009	0.620	0.644	0.627	0.007
BERT-large	0.828	0.921	0.872	0.002	0.672	0.448	0.527	0.021	0.684	0.751	0.700	0.010
RoBERTa-large	0.873	0.915	0.893	0.004	0.716	0.610	0.650	0.026	0.763	0.795	0.771	0.014
DeBERTa-v3-large	0.867	0.906	0.885	0.005	0.694	0.598	0.638	0.017	0.752	0.780	0.761	0.011
Random baseline (F1)	0.618											
Majority baseline (F1)	0.743											

*Random and majority baselines represent macro-average F1 scores.

The results from Table 1 align with our expectations, confirming that Transformers models produce the most accurate classifications for civil and incivil tweets. The F1 score for RoBERTa's classification of civil tweets is hight at 0.893. For incivil tweets, on the other hand, RoBERTa's F1 score stands at 0.650 on average over 10-times repeated 10-fold cross-validation (100 models in total). The model's overall F1 accuracy for RoBERTa stands at 0.771. DeBERTa performs similarly to RoBERTa across all models, but BERT's performance is significantly worse. BERT's F1 score for incivil tweets is 0.527, a drop-off in performance of 0.123 when compared to RoBERTa's 0.650. This is a statistically significant difference considering both models' standard deviations. BERT performs well with civil tweets ($F1 = 0.872$) but its overall performance is still worse than RoBERTa and DeBERTa.

Compared to the baselines, all Transformer models improve upon the performance of simple machine learning and standard neural network approaches. As expected, the Bi-LSTM RNN with GloVe embeddings performs better than SVMs in classifying incivil and civil tweets. Simpler machine learning models cannot understand the linguistic nuances in civil and incivil tweets and produce low levels of accuracy for incivil tweets.[37] RoBERTa still improves on Bi-LSTM-GloVe's performance by 53.2% in the incivil tweets category and 23% in overall model performance (when comparing F1 scores). Note that overall levels of accuracy may appear artificially high for these two models, considering that civil tweets are easiest to classify as they represent 74.3% of the sample. Indeed, the random accuracy for the (unbalanced) dataset is 74.3% for zeroes (civil) and 25.7% for ones (incivil).[38]

[37]The higher accuracy of SVM and Logistic Regression in civil tweets is trivial. The models default to predicting civil when they cannot distinguish between the two because it is the category with the most tweets in the sample.

[38]Random accuracy refers to the underlying probability that the model classifies an event correctly using only the share of a given category in the data. Random accuracy reflects only a lucky guess by the model without the need for further learning.

We therefore recommend that researchers fine-tune a RoBERTa model to generate labeled data from a larger set of texts. Equipped with this larger dataset with both human-labeled and machine-labeled data, we the recommend applying the Egami et al. (2023) method in downstream statistical analyses. This helps identify and correct for measurement bias in the machine-labeled data. We provide a full example of this method in Application 3 and Appendix I.

2. Classifying multi-lingual speeches

The second project we use to illustrate the advantages of Transformers-based models of NLP is the Global Populism Database (GPD) introduced by Hawkins et al. (2019).[39] The GPD project started in 2006 with the goal of creating a large dataset of global populist discourse by political leaders. The project currently contains 1,161 speeches by 234 leaders from 73 countries. The speeches are in 29 different languages[40] and their length ranges from 18 to 20,587 words, with a mean length of 2,449.8 words and a median length of 1,938 words. The languages in the sample are diverse. Pre-trained models in high-resource languages–i.e. languages with a large amount of data available– such as English, Spanish and German, use around 100GB of data. For low-resource languages, such as Albanian, Latvian, and Tagalog, the pre-trained data can be closer to 10GB. The lack of training data for some of these languages poses a general challenge for XLM-RoBERTa models to accurately classify text in certain languages (see Conneau et al., 2019). For us, the challenge is to fine-tune two cross-lingual classifiers using a relatively small sample size (1,161 speeches).

GPD's coders have generated multiple variables from the speech data. We will focus on one of these, the *type of speech*, which has four categories: *international* (the audience is foreign and is preferably given outside the country); *campaign* (usually the opening or closing of the campaign); *ribboncutting* (given to a local audience), and *famous* (a widely circulated speech that shows the leader at his or her best).[41] Classifying the type of speech helps identify the context in which populism most frequently occurs. The distribution of the categories is as follows: 304 speeches are 'famous' (26.2%), 304 are 'international' (26.2%), 294 are 'ribboncutting' (25.3%), and 259 are 'campaign' (22.3%). A majority of speeches are over 512 words and tokens (99.6%), so we set a maximum length of 512 for the model and select the first 512 tokens.

The challenges with this dataset are three. First, speeches are long (some have over 50,000 words), and Transformers-based models are limited to 512 tokens –and with 512 tokens, the computational requirements for graphics acceleration are high. The other problem is sample size.[42] Having 843 speeches across 3 or 4 categories leaves only between 200 and 300 speeches

[39]The full dataset and the codebook can be found at: `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/LFTQEZ`.

[40]Bulgarian, Czech, German, Greek, English, Spanish, Estonian, Finnish, French, Hungarian, Croatian, Italian, Japanese, Lithuanian, Latvian, Macedonian, Dutch, Norwegian, Polish, Portuguese, Romanian, Russian, Slovak, Slovenian, Albanian, Swedish, Tagalog, Turkish, and Ukrainian.

[41]See also The Guardian, 'How we combed leaders' speeches to gauge populist rise", 9 March 2019. `https://www.theguardian.com/world/2019/mar/06/how-we-combed-leaders-speeches-to-gauge-populist-rise`.

[42]There are important implications related to the 512 token limit for Transformer models. First, longer texts (say Supreme Court decisions) may have multiple sections and thus key pieces of information may be scattered. If, say, we set a rule to use the first 512 tokens, we may miss important information for the

per category to train the model and a lower number for the training set after the train-test split required in 10-fold cross-validation. Lastly, this multilingual dataset covers tokens for English, Spanish, Japanese, Chinese, Albanian, among many others. Considering these challenges, the GPD data provide a strong test for the ability of XLM-R to classify speech types accurately.

For this test, we compare the performance of XLM-R, multilingual BERT (mBERT), and multilingual DeBERTa (mDeBERTa). As in the first application, we provide baseline results for SVMs and Bi-LSTM with GloVe embeddings.[43] Note, however, that the SVM's tokenizer (NLTK) and the Bi-LSTM's word embeddings (GloVe) are not cross-lingual, so we have first to translate all speeches into English (we could also translate them into other languages, but English is usually the most accurate). We did this using Google Translate, a service whose main engine is a refined and proprietary multilingual BERT model.[44] We used these models the same way we did with the English text in the first application above. We used all speeches in their original languages for the XLM-R, mBERT, and mDeBERTa models.[45]

Table 2: Model performance by speech type (10x repeated 10-fold CV scores)

Model	**Campaign**				**Famous**				**International**				**Ribbon-cutting**				**Macro-Average**			
	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}
ML - SVM	0.716	0.668	0.684	0.008	0.559	0.618	0.580	0.009	0.809	0.831	0.817	0.010	0.775	0.708	0.740	0.010	0.708	0.715	0.705	0.006
Bi-LSTM - GloVe	0.730	0.715	0.709	0.007	0.552	0.558	0.540	0.015	0.759	0.708	0.719	0.029	0.697	0.668	0.682	0.017	0.668	0.685	0.663	0.005
mBERT	0.794	0.777	0.783	0.002	0.722	0.668	0.689	0.004	0.844	0.870	0.854	0.011	0.819	0.853	0.832	0.002	0.792	0.795	0.789	0.002
XLM-RoBERTa	0.838	0.845	0.839	0.004	0.792	0.724	0.750	0.011	0.863	0.933	0.895	0.007	0.865	0.854	0.856	0.014	0.839	0.840	0.835	0.009
mDeBERTa	0.784	0.777	0.775	0.011	0.697	0.652	0.670	0.012	0.831	0.882	0.852	0.005	0.833	0.831	0.828	0.004	0.786	0.786	0.781	0.001
Random baseline	0.25																			
Majority baseline	0.262																			

*Random and majority baselines represent macro-average F1 scores.

Table 2 shows the results. The F1 scores show that all Transformer models substantially improve upon the performance of SVM and Bi-LSTM, which is to be expected. Within the Transformers family, XLM-R performance is particularly impressive. Its F1 score for all types of speeches is much higher than mBERT and mDeBERTa, which score similarly. The largest difference is with 'famous' speeches, which XLM-R classifies 8 percentage points more accurately than mDeBERTa (6.1 when compared with mBERT). This represents a 11.94%

classifier. This is, in our view, the most relevant drawback of the 512 limit. The authors have found in our testing (both for this article and other independent research), that classifier performance does not improve after using around 300 tokens in texts longer than 512 tokens. This may appear counterintuitive, but in fact the classifiers are so accurate that it is often the case that around 300 tokens are sufficient to comprehend the meaning of a text, and the increases in performance after that are marginal. The exception are texts where information may be hidden in sections beyond the 512 limit. For those applications, we recommend pre-processing the text to improve on the rule for which set of tokens to select from the text.

[43] We use a learning rate of 5e-06, a batch size of 16 and 4 epochs for XLM-R; and 3e-05, 32 batch size and 4 epochs for both mBERT and mDeBERTa. We set the max length to 512 tokens per speech, the maximum we can fit into a Nvidia A100 GPU (80GB of RAM).

[44] We used Google's API and kept the translations to a maximum of 3000 words per speech to keep the comparison with other models fair and costs down.

[45] We do not expect the translation to be a source of bias, considering the improvements in accuracy of machine translation in recent years especially with English as the target language. It is an inherent weakness of non-Transformer models that they cannot handle text in more than one language at a time.

increase in performance. XLM-R outperforms the SVM model by 29.3% and the Bi-LSTM by 38.9% in classifying 'famous' speeches. Furthemore, XLM-R outperforms mDeBERTa by around 4.5 percentage points on average in the other three categories. With XLM-R, two of the four speech type categories have F1 scores over 0.839 (campaign and ribbon-cutting speeches) and close to 0.90 (international speeches). High precision and recall scores show that neither false positives nor false negatives are of concern. Across all speech categories, mBERT and mDeBERTa perform better than the non-Transformer models but noticeably worse than XLM-R. This difference is likely because XLM-R is a larger model than mBERT and mDeBERTa, with more training data, tokens, and parameters, making it more accurate for cross-lingual applications with longer texts overall.[46]

These results are particularly encouraging considering the data limitations mentioned above. Even though the classes are well-balanced (around 300 observations in each speech type),[47] there are only 1,161 speeches to train and test the model. Moreover, the speeches are in 29 languages, some of which are trained on much less text data than others. Compounding this, we can only take the first 512 tokens from each speech for classification due to model and GPU constraints. However, the model performs very well, showing the true potential of XLM-RoBERTa and other Transformers-based cross-lingual models in text classification tasks, especially with small training sets.

Given the results above, we recommend that researchers fine-tune an XLM-RoBERTa model over mBERT or mDeBERTa to generate labels in a larger dataset based on the human coded data from the GPD. Again, once the machine-labeled data is available, applied researchers should follow Egami et al. (2023) to detect and correct for any measurement bias in the data before conducting their final analyses (see Application 3 and Appendix I).

3. Detecting Covid-19 Fake News

The third application focuses on detecting fake news around the Covid-19 pandemic and shows the flexibility of Transformers models and how to apply them to specific tasks through further training. This application is especially relevant to show the opportunity that Transformers models hold for increasing performance in specialized domains. While we focus on a specific case here (COVID-19), there are many domains in which researchers can greatly improve classifier performance –and therefore generate better data– by following the steps we outline in this section.[48] We use a manually labeled dataset of true and fake news around the Covid-19 pandemic (see Cheng et al., 2021), an increasingly salient topic of study within political science (Calvo and Ventura, 2021; Greer et al., 2021; Timoneda and Vallejo Vera, 2021). The authors of the project gathered 7,179 news headlines and Twitter posts containing the words 'coronavirus' or 'covid' between December 2019 and September 2020. Through fact-checking websites, they labeled each story as fake, true, or undetermined.[49] Lazer et al. (2018) define 'fake news' as 'fabricated information that mimics news media content in form

[46] Please see Appendix A. For XLM-R, both $xlm-roberta-large$ and $xlm-roberta-base$, a smaller model, exist. However, for mBERT and mDeBERTa, only *base* versions exist. We take the best possible model available from each for this application to show the true power of available cross-lingual models.

[47] The exact numbers are: famous, 304; international, 304; ribbon-cutting, 294; campaign, 259.

[48] For instance, researchers have increased performance of RoBERTa models to increase performance in the classification of US Supreme Court decisions (citation ommitted) and of racist text in Spanish (citation ommitted).

[49] The dataset is available at: `https://github.com/MickeysClubhouse/COVID-19-rumor-dataset`.

but not in organizational process or intent'. Examples of fake news from the aformentioned dataset are: 'Coronavirus was created in a government lab as a bioweapon and then released on the people of China' or 'Japanese doctors advice that taking a few sips of water every 15 mins will prevent the new coronavirus from entering your windpipe and lungs'. We use the final dataset which has 3,681 fake (51.27%), 1,878 true (26.16%), and 1,620 (22.57%) undetermined news stories and tweets. The mean length for the headlines is 21.56 words, and the longest is 143 words, resulting in an average of 29.75 tokens per sentence using the RoBERTa tokenizer and a maximum number of tokens of 160.

Data on the coronavirus pandemic provides a clear example of the advantages of further training the different transformer models. When RoBERTa, for instance, was originally trained in 2018-19, Covid-19 did not yet exist. Coronaviruses had circulated for years, but none had resulted in a global pandemic. The word 'covid' did not exist until February of 2020, so this case provides an intuitive application for how to further train a model on highly specialized words that we can be certain the original model was never trained to understand. Original RoBERTa, therefore, cannot know what Covid-19 is or how the words 'coronavirus' or 'covid' are used in context today without further training. Our solution is to further train a new RoBERTa model with new data containing the words 'coronavirus' and 'covid' and add those two vocabulary elements to the tokenizer. The new model should be able to classify texts containing these two new words more accurately than original RoBERTa.

There are four steps to further train RoBERTa and other Transformers models. First, we add two new vocabulary elements –'covid' and 'coronavirus'– to the RoBERTa-large (fast) tokenizer, neither of which exists in the original. The tokenizer now has 50,267 unique vocabulary elements, two more than the original's 50,265. Second, we take the pre-existing vector representations for 'virus' and 'respiratory' from the existing set of RoBERTa vocabulary elements and assign the mean of those two word vectors to be the vector representation for the newly created elements 'covid' and 'coronavirus'.[50] Third, to improve upon these approximate initial representations, we feed a set of unlabelled texts containing the words 'covid' and 'coronavirus' in English to the model and train it again. In our case, we used 6,079 abstracts from academic articles on the topic of Covid-19 and the coronavirus pandemic. We also added 1GB of short news headlines around the Covid-19 pandemic obtained from Twitter. The format and nature of these texts closely matches the type of training text data that we use to build the classifier (tweets). Note that none of the unlabelled tweets used for further training are present in the labelled training data. The amount of text needed for improved task-specific performance varies from task to task, but usually the more text, the more accurate the model becomes. Fourth, we save the new model and apply it to our classification task –building a classifier using labelled training data– in the same way we would apply original RoBERTa.[51]

We repeat this process for each of the transformers models we compare in this arti-

[50]This choice is arbitrary though based on theory. Given the numerical representations of word embedding, we assume that virus + respiratory ≈ coronavirus. This is only a starting point for the new vocabulary elements. Once we further train the model, the embeddings for coronavirus and covid will adjust and become more accurate representations of their meaning.

[51]See Appendix E for a more detailed step-by-step guide on how to further train a Transformer model with Python code snippets.

cle: BERT, RoBERTa, and DeBERTa, using each model's own tokenizer.[52] We compare the performance of the resulting three new models (BERT-Covid, RoBERTa-Covid, and DeBERTa-Covid) with that of the original models. We use 10-times repeated 10-fold CV.[53]

Table 3: Model performance on Fake News Dataset, 10x repeated 10-fold CV

Model	**Fake**				**True**				**Undetermined**				**Macro-Average**			
	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}	Prec.	Rec.	F1	SD_{F1}
ML - SVM	0.717	0.749	0.731	0.010	0.701	0.731	0.713	0.006	0.743	0.680	0.708	0.007	0.720	0.720	0.717	0.004
Bi-LSTM - GloVe	0.719	0.732	0.717	0.014	0.720	0.702	0.703	0.013	0.712	0.692	0.694	0.013	0.709	0.717	0.705	0.11
BERT	0.805	0.766	0.782	0.004	0.745	0.806	0.772	0.009	0.748	0.714	0.724	0.003	0.762	0.767	0.760	0.003
BERT-Covid	0.857	0.784	0.815	0.003	0.772	0.809	0.786	0.002	0.744	0.762	0.749	0.005	0.785	0.791	0.783	0.003
RoBERTa	0.872	0.778	0.819	0.010	0.753	0.827	0.783	0.009	0.748	0.745	0.742	0.011	0.783	0.791	0.781	0.008
RoBERTa-Covid	0.875	0.835	0.851	0.008	0.788	0.840	0.810	0.004	0.784	0.758	0.767	0.005	0.811	0.816	0.809	0.003
DeBERTa	0.865	0.820	0.838	0.007	0.754	0.832	0.788	0.008	0.798	0.738	0.761	0.006	0.797	0.806	0.796	0.004
DeBERTa-Covid	0.881	0.834	0.855	0.007	0.772	0.850	0.808	0.003	0.786	0.742	0.760	0.002	0.809	0.813	0.808	0.001
Random baseline	0.44															
Majority baseline	0.57															

*Random and majority baselines represent macro-average F1 scores.

The results are in Table 3. We compare the performance of BERT, RoBERTa, and DeBERTa with (1) their respective covid-specific further-trained models, and (2) the same two baseline models from tables 1 and 2 –SVMs and Bi-LSTM. The results confirm our expectations. First, all of the Transformers models outperform the baseline models significantly. BERT, the lowest performing Transformer model of the three, still outperforms SVM and Bi-LSTM-GloVe by 7% and 9.1%, respectively, in terms of their F1 scores when classifying fake news. Similar gains apply to true and undetermined news. RoBERTa and DeBERTa show even more significant gains in performance when compared to the two baselines.

The most relevant results are in comparing original BERT, RoBERTa and DeBERTa with their respective covid-trained models. All the covid-trained models show substantively significant gains in performance when comparing F1 scores, especially when classifying fake news. First, BERT-Covid improves upon BERT by 4.22% in fake news, 1.81% in true news, and 3.45% in undetermined. RoBERTa-Covid bests RoBERTa by 3.91% in fake news, 3.45% in true news, and 3.37% in undetermined. Lastly, DeBERTa-covid outperforms DeBERTa by 2.03% in fake news, by 2.56% in true news, and there is no improvement in undetermined news. The main conclusion is this: *all* three further-trained models increase performance in classifying fake news around the COVID-19 pandemic by a range between 2.03% and 4.22%. RoBERTa sees the largest improvement in classifying true stories with 3.45%.

These results are especially significant in substantive terms for two reasons. First is the amount of training data that we used to train the new COVID-19 models. We further trained the Transformer models with 6,079 abstracts of academic articles on the topic of COVID-19, or 8.4 *megabytes* of text. We then added 4.8 million short news headlines in English that

[52]From the Transformers library, we use BertTokenizer for $bert-large-uncased$; RobertaTokenizerFast for $roberta-large$; and DebertaV2TokenizerFast for $deberta-v3-large$.

[53]We use the same hyperparameters to train the original and new models to ensure the results are comparable.

total 978 megabytes of text. Original RoBERTa, on the other hand, was trained on 160 *gigabytes* of text. Our unlabelled data represent a fraction of the total training data in terms of quantity and yet there are noticeable increases in overall performance. It is reasonable to expect larger performance gains with more training data, or if we retrained the model from scratch. Second, our results are averages of 100 different models across ten different sets of 10-fold cross-validation. We can be certain that if we were to draw more model samples the differences between the original and the Covid models would remain.

In all, these results show the potential that further training models holds for increasing performance in task-specific applications. Differences across models are significant, especially considering that original BERT, RoBERTa and DeBERTa already perform well in this application. Note, however, that performance will vary across domains, and researchers should decide whether further training the model is warranted for their specific application. We argue that doing so is especially important in domains where classifiers generally do not perform as well, which is often due to especially complex contextual understanding or to the use of highly specialized language. (citation omitted), for instance, find that further training an XLM-RoBERTa model improved classifier performance by 8%. In other situations, where researchers deem the performance of the original Transformer models to be sufficient, continuing without additional training can be an optimal choice.

In light of these results, our recommendation in this application is to fine-tune a RoBERTa model, further training it with unlabeled Covid-related data. To illustrate how to incorporate machine-labeled data in downstream statistical analyses, we provide an example by analyzing the effect of the length of a tweet in words on whether the content is fake news. We draw a random set of 3,000 news tweets around the coronavirus pandemic from the CoAID dataset by Cui and Lee (2020), who code each tweet as fake or true. These tweets are neither in the unlabeled dataset for further training nor in the final labeled dataset for standard fine-tuning. 1,500 tweets are used as human-coded, gold-standard data. Another set of 1,500 are used as unlabelled data, and we apply our `Roberta-Covid` model to predict the labels. The final dependent variable is whether a tweet is fake news or not. The independent variable is tweet length. Further details of these tests as well as the complete set of results are in Appendix I. The code is available from GitHub (omitted).[54] We find that there can be substantial discrepancies in model coefficients when using gold-standard or machine-coded labels, which is indicative of measurement bias. Applying Egami et al.'s (2023) test significantly alleviates measurement bias, and we recommend that scholars incorporate it into their workflow in downstream analyses using labels produced with fine-tuned Transformers models.

Limitations

Despite the substantial gains in performance and the flexibility of Transformers models, they have limitations. First, performance gains depend on setting the right hyperparameters for each application. This requires the researcher to perform various cross-validation test runs to determine the learning rate that maximizes performance, minimizing both underfitting and overfitting.[55] Second, these models will most often be used to create categorical variables that

[54]The original Egami et al. (2023) code is available at: $osf.io/gjt87/?view\_only = 8f755cdf147f452a9429797\ 3eb83d85d$.

[55]Usually, a model overfits when the training loss is much smaller than the test loss. This means the model learns patterns in the training data too closely, thus minimizing training loss, but those patterns may not

researchers can then add to their analyses. Since there is uncertainty and error around the Transformer model's classified labels, this error will then bleed into the analyses conducted with them. We recommend following Egami et al.'s (2023) method to detect and correct for measurement bias in downstream statistical analysis. That said, any coding procedure will suffer from some form of coding error. Hence, researchers should focus on training models that produce the highest levels of accuracy for the categories that will then be used in their analyses. It is especially important to report variation across cross-validation runs to understand the extent of the uncertainty around model results.

Third, the labelled training data in the final standard fine-tuning step that produces the classifier needs to be balanced for best performance. This may run counter to the balance of the categories in the real world, where some categories may be rare events but require a larger share of the training data for the model to operate correctly.[56] Fourth, NLP models are known to suffer from multimodality, which occurs when the "function we are trying to optimize is not globally concave" (Roberts, Stewart and Tingley, 2016, p.2). When estimating a topic model, for example, we cannot know if the topics obtained were global maxima or local maxima (the function has multiple modes). This would suggest that our models are sensitive to starting values. Transformer-based models might suffer from multimodality too, which is one of the reasons we emphasize the importance of repeated cross-validation as a strategy to validate model results. In our results we show that performance is quite consistent across different cross-validation runs. Researchers must set hyperparameters carefully, using cross-validation to determine the best set for their particular application (see Appendix B).[57]

Lastly, The performance gains from Transformers models come at a computational cost. The sheer amount of parallel calculations involved in the self-attention mechanism requires computational power beyond the fastest CPUs available. For this reason, Transformers models are almost always run in one or multiple graphics processing units, or *GPUs*, which have thousands of small cores and can perform all these calculations orders of magnitude faster.[58] For smaller RoBERTa models, scholars can use Google Colab to run our sample code. Alternatively, they can access other cloud-based solutions with top-end GPUs, whose cost is now small and accessible accessible.[59] While more costly, researchers can also install one or more GPUs on their local Linux, or Windows desktop with CUDA enabled, or use institutional resources dedicated for this purpose.[60]

---

exist in the test data and, therefore, the model does not generalize well. Conversely, a model underfits when the training loss is larger than the test loss, meaning that there is still more that the model can learn from the training data to generalize better to unseen data. Cross-validation helps find the right training balance that avoids both underfitting and overfitting.

[56] Note that the training data need not be perfectly balanced. Rather, researchers need to keep the ratio between 2:1 and 3:1. With larger ration, the model will tend to overpredict the more frequent category to the detriment of the more infrequent one.

[57] In Appendix B we document the hyperparameters used to train our cases as reference for readers.

[58] We understand that there is unequal access to computational resources in academia. While the costs of using Transformers-based models are not prohibitive (see Appendix A), they still pose a major obstacle to their use for research. Future work should aim to democratize these resources' use by lowering their cost.

[59] Examples are datacrunch.io or lambdalabs.com, who offer access to top GPUs for amounts between $0.8 and $2 per hour. See Appendix D for further details on model-specific GPU performance and resources available to scholars.

[60] CUDA is Nvidia's free software that allows parallel computing on GPUs (available on Linux and Windows).

Conclusion

This article introduces Transformers-based classification models for English and multilingual text. We compare the performance of BERT, RoBERTa and DeBERTa with other current state-of-the-art models used in political science and find multiple advantages. First, their capacity to understand context is greater than non-Transformer-based models. Understanding context increases model performance when generalizing to unseen data. Yet not all models perform equally well: RoBERTa and DeBERTa generally perform much better than BERT, as our first application demonstrates. Second, the multilingual variants of different Transformers models, in particular XLM-R, perform exceptionally well when the training data is in multiple languages. Multilingual availability is important in comparative politics and international relations, especially with low-resource languages.[61] Third, Transformers models are flexible and can be further trained to fit a specific task, allowing researchers to 'customize' a model to increase performance.

As stated above, the aim of the paper is to compare the performance of recently developed Transformer models and highlight the versatility of these methods in social science research. In Appendix G, we also showcase the application of our models in an established case by replicating Abercrombie and Batista-Navarro (2022) classification task of policy preferences in parliamentary speeches. By applying RoBERTa to their training set, we are able to increase performance considerably.[62] Furthermore, there are other practical benefits of using Transformers models. Given the model's overall improvements in out-of-sample accuracy, scholars can use them more consistently to turn their projects into big data projects, using manually labeled data as training data instead of as the full sample. This benefit is especially relevant with text data, which is often so abundant that the real limitations researchers face are related to resources and scaling costs. Transformers models –as other machine learning models– also allow researchers to understand their data much better. These models can also help us answer substantive questions that were impossible to address until now, primarily because other machine learning and deep learning approaches had difficulty understanding the contextual nuances of text.

We think the advantages of Transformers models greatly outweigh the limitations introduced in the previous section. This is especially true considering that the models introduced in this article represent a paradigm shift in NLP applications. They open up new avenues and opportunities for political science research across all subfields. Indeed, considering that text data is and will continue to be one of the biggest sources of data in the discipline, harnessing the power of these new models –and the ones that will inevitably follow– can have a transformational effect in applied political science research.

[61] Transformers models allow political science researchers to overcome structural limitations of NLP when analyzing corpora of low-resource languages–i.e., languages lacking large monolingual or parallel corpora or manually crafted linguistic resources sufficient for building statistical NLP applications (Magueresse, Carles and Heetderks, 2020). The multi-lingual capabilities of models like mBERT and XLM-RoBERTa reduce the cost researchers have to incur in, for example, building new dictionaries or training Word2Vec-type embeddings.

[62] Abercrombie and Batista-Navarro (2022) classify 34 topics to estimate policy preferences in speeches from the English Parliament. They apply a BERT-based model and obtain an F1 score of 50.9. We replicate their findings and apply a RoBERTa model, obtaining a 9.3 percentage point increase in the benchmark test. See Appendix G for the performance table and an explanation of the replication process.

References

Abercrombie, Gavin, Federico Nanni, Riza Batista-Navarro and Simone Paolo Ponzetto. 2019. Policy preference detection in parliamentary debate motions. Association for Computational Linguistics.

Abercrombie, Gavin and Riza Theresa Batista-Navarro. 2022. Policy-focused Stance Detection in Parliamentary Debate Speeches. In *Northern European Journal of Language Technology, Volume 8*.

Albawi, Saad, Tareq Abed Mohammed and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In *2017 international conference on engineering (ICET)*. pp. 1–6.

Alemán, Eduardo, Juan Pablo Micozzi and Sebastián Vallejo Vera. 2022. "Congressional Committees, Electoral Connections, and Legislative Speech." *Political Research Quaterly* .

Barberá, Pablo, Amber E Boydstun, Suzanna Linn, Ryan McMahon and Jonathan Nagler. 2021. "Automated text classification of news articles." *Political Analysis* 29(1):19–42.

Beieler, John, Patrick T Brandt, Andrew Halterman, Philip A Schrodt, Erin M Simpson and R Michael Alvarez. 2016. "Generating political event data in near real time." *Computational social science* p. 98.

Bojanowski, Piotr, Edouard Grave, Armand Joulin and Tomas Mikolov. 2017. "Enriching word vectors with subword information." *Transactions of the association for computational linguistics* 5:135–146.

Bonikowski, Bart, Yuchen Luo and Oscar Stuhler. 2022. "Politics as Usual? Measuring Populism, Nationalism, and Authoritarianism in US Presidential Campaigns (1952–2020) with Neural Language Models." *Sociological Methods & Research* 51(4):1721–1787.

Calvo, Ernesto and Tiago Ventura. 2021. "Will I get COVID-19?" *Latin American politics and society* 63(1):1–26.

Catalinac, Amy. 2016. *Electoral reform and national security in Japan: From pork to foreign policy*. Cambridge University Press.

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

Cheng, Mingxi, Songli Wang, Xiaofeng Yan, Tianqi Yang, Wenshuo Wang, Zehao Huang, Xiongye Xiao, Shahin Nazarian and Paul Bogdan. 2021. "A COVID-19 rumor dataset." *Frontiers in Psychology* 12.

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer and Veselin Stoyanov. 2019. "Unsupervised cross-lingual representation learning at scale." *arXiv preprint* .

Cui, Limeng and Dongwon Lee. 2020. "Coaid: Covid-19 healthcare misinformation dataset." *arXiv preprint arXiv:2006.00885* .

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2018. "Bert: Pre-training of bidirectional transformers for language understanding." *arXiv preprint* .

Egami, Naoki, Musashi Jacobs-Harukawa, Brandon M Stewart and Hanying Wei. 2023. "Using Large Language Model Annotations for Valid Downstream Statistical Inference in Social Science: Design-Based Semi-Supervised Learning." *arXiv preprint arXiv:2306.04746* .

Gervais, Bryan T. and Irwin L. Morris. 2019. When Parties Tweet: Assessing and Explaining Incivility in State Legislatures. In *Southern Political Science Association annual meeting*.

Gleason, Shane A. 2020. "Beyond mere presence: Gender norms in oral arguments at the us supreme court." *Political Research Quarterly* 73(3):596–608.

González-Bailón, Sandra and Georgios Paltoglou. 2015. "Signals of public opinion in online communication: A comparison of methods and data sources." *The ANNALS of the American Academy of Political and Social Science* 659(1):95–107.

Greer, Scott L, Elizabeth King, Elize Massard da Fonseca and André Peralta-Santos. 2021. *Coronavirus politics: The comparative politics and policy of COVID-19*. University of Michigan Press.

Grimmer, Justin. 2010. "A Bayesian hierarchical topic model for political texts: Measuring expressed agendas in Senate press releases." *Political Analysis* 18(1):1–35.

Grimmer, Justin, Margaret E Roberts and Brandon M Stewart. 2022. *Text as data: A new framework for machine learning and the social sciences*. Princeton University Press.

Hawkins, Kirk A, Rosario Aguilar, Bruno Castanho Silva, Erin K Jenne, Bojana Kocijan and Cristóbal Rovira Kaltwasser. 2019. Measuring populist discourse: The global populism database. In *EPSA Annual Conference in Belfast, UK, June*. pp. 20–22.

He, Pengcheng, Xiaodong Liu, Jianfeng Gao and Weizhu Chen. 2020. "Deberta: Decoding-enhanced bert with disentangled attention." *arXiv preprint arXiv:2006.03654* .

Hobbs, William and Nazita Lajevardi. 2019. "Effects of divisive political campaigns on the day-to-day segregation of Arab and Muslim Americans." *American Political Science Review* 113(1):270–276.

Kaneko, Masahiro and Danushka Bollegala. 2022. Unmasking the mask–evaluating social biases in masked language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36 pp. 11954–11962.

Khan, Junaed Younus, Md Tawkat Islam Khondaker, Sadia Afroz, Gias Uddin and Anindya Iqbal. 2021. "A benchmark study of machine learning models for online fake news detection." *Machine Learning with Applications* 4:100032.

King, Gary, Jennifer Pan and Margaret E Roberts. 2013. "How censorship in China allows government criticism but silences collective expression." *American political science Review* 107(2):326–343.

Lazer, David MJ, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild et al. 2018. "The science of fake news." *Science* 359(6380):1094–1096.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer and Veselin Stoyanov. 2019. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692* .

Loper, Edward and Steve Bird. 2002. "Nltk: The natural language toolkit." *arXiv preprint* .

Loshchilov, Ilya and Frank Hutter. 2017. "Decoupled weight decay regularization." *arXiv preprint* .

Magueresse, Alexandre, Vincent Carles and Evan Heetderks. 2020. "Low-resource languages: A review of past work and future challenges." *arXiv preprint arXiv:2006.07264* .

Matalon, Yogev, Ofir Magdaci, Adam Almozlino and Dan Yamin. 2021. "Using sentiment analysis to predict opinion inversion in Tweets of political communication." *Scientific reports* 11(1):1–9.

McGregor, Shannon C. 2020. ""Taking the temperature of the room" how political campaigns use social media to understand and represent public opinion." *Public Opinion Quarterly* 84(S1):236–256.

Medzihorsky, Juraj, Levente Littvay and Erin K Jenne. 2014. "Has the tea party era radicalized the Republican party? Evidence from text analysis of the 2008 and 2012 Republican primary debates." *PS: Political Science & Politics* 47(4):806–812.

Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch and Armand Joulin. 2017. "Advances in pre-training distributed word representations." *arXiv preprint* .

Pan, Jennifer and Kaiping Chen. 2018. "Concealing corruption: How Chinese officials distort upward reporting of online grievances." *American Political Science Review* 112(3):602–620.

Pennington, Jeffrey, Richard Socher and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pp. 1532–1543.

Ravichandiran, Sudharsan. 2021. *Getting Started with Google BERT*. Packt Publishing.

Roberts, ME, BM Stewart and D Tingley. 2016. "Navigating the local modes of big data: the case of topic models. Comput." *Soc. Sci* 4.

Rodriguez, Pedro L and Arthur Spirling. 2022. "Word embeddings: What works, what doesn't, and how to tell the difference for applied research." *The Journal of Politics* 84(1):101–115.

Strother, Logan. 2017. "How expected political and legal impact drive media coverage of Supreme Court cases." *Political Communication* 34(4):571–589.

Timoneda, Joan C and Sebastián Vallejo Vera. 2021. "Will I die of coronavirus? Google Trends data reveal that politics determine virus fears." *Plos one* 16(10):e0258189.

Tunstall, Lewis, Leandro von Werra and Thomas Wolf. 2022. *Natural Language Processing with Transformers*. "O'Reilly Media, Inc.".

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. pp. 5998–6008.

Ward, Michael D, Andreas Beger, Josh Cutler, Matthew Dickenson, Cassy Dorff and Ben Radford. 2013. "Comparing GDELT and ICEWS event data." *Analysis* 21(1):267–297.

Wettig, Alexander, Tianyu Gao, Zexuan Zhong and Danqi Chen. 2022. "Should you mask 15% in masked language modeling?" *arXiv preprint* .

Zhang, Tianyi and Tatsunori Hashimoto. 2021. "On the inductive bias of masked language modeling: From statistical to syntactic dependencies." *arXiv preprint* .