Political Science Research and Methods (2024), page 1 of 18
doi:10.1017/psrm.2024.35

ORIGINAL ARTICLE

Stance detection: a practical guide to classifying political beliefs in text

Michael Burnham

Department of Political Science, Pennsylvania State University, State College, PA, USA
Email: mlb6496@psu.edu

(Received 28 November 2023; revised 28 March 2024; accepted 28 May 2024)

Abstract

Stance detection is identifying expressed beliefs in a document. While researchers widely use sentiment analysis for this, recent research demonstrates that sentiment and stance are distinct. This paper advances text analysis methods by precisely defining stance detection and outlining three approaches: supervised classification, natural language inference, and in-context learning. I discuss how document context and trade-offs between resources and workload should inform your methods. For all three approaches I provide guidance on application and validation techniques, as well as coding tutorials for implementation. Finally, I demonstrate how newer classification approaches can replicate supervised classifiers.

Keywords: natural language processing; sentiment analysis; stance detection; text as data

Stance detection is a common task for political scientists analyzing text.[1] It includes identifying support or opposition toward electoral candidates (Barberá, 2016) or social movements (Felmlee *et al.*, 2020). It may involve identifying attitudes toward political (Bor *et al.*, 2023), religious (Terman, 2017), or racial (Lee *et al.*, 2022) groups. Or, researchers may measure positions on factual matters such as whether or not COVID-19 poses a public health risk (Block *et al.*, 2022) or other types of misinformation (Osmundsen *et al.*, 2021).

To measure stance, social scientists widely rely on sentiment analysis—the identification of positive or negative emotion in text (Liu, 2010; Hutto and Gilbert, 2014). However, recent research shows sentiment is often uncorrelated with opinion (Aldayel and Magdy, 2019; Bestvater and Monroe, 2022). As AlDayel and Magdy (2021) argue, sentiment is a dimension distinct from stance.

This paper builds on AlDayel and Magdy (2021) and Bestvater and Monroe (2022) who conclude that stance detection is a classification task and recommend training supervised classifiers. Although text classification is well established in political science, opinion classification is particularly difficult and warrants special considerations (Aldayel and Magdy, 2019; AlDayel and Magdy, 2021). Further, recently developed language models introduced new classification methods that do not require supervised training.

I advance theory and methods in three ways: First I provide a precise definition of stance detection. Second, I provide an overview of three classification paradigms: Supervised classification, which trains language models for a specific task; natural language inference (NLI), which uses language models pre-trained as "universal" classifiers; and in-context learning, which uses generative models like

[1]A quick-reference guide for methods discussed, as well as a glossary of technical terms, can be found in the appendices. Coding tutorials can be found at github.com/MLBurnham/stance_detection_tutorials

GPT-4. Finally, I provide practical guidance on selecting and implementing an approach. This article presents a conceptual overview of the motivation and implementation for each paradigm. For guidance on technical implementation, I include coding tutorials in the online materials.

I particularly emphasize NLI classifiers. While supervised methods are well established and generative models have attracted significant attention from political scientists (e.g. Gilardi *et al.*, 2023; Törnberg, 2023b), NLI classifiers are largely absent from the literature despite their advantages: classification that scales to large data sets, reproduces well, and does not require training. In the final section, I use an NLI classifier to replicate the findings of Block *et al.* (2022)—which used a supervised classifier to label stances toward COVID-19 mitigation policies.

1. Stance and Stance detection

1.1 What is stance?

Stance is an individual's "attitudes, feelings, judgments, or commitment" to a proposition (Biber and Finegan, 1988). Stance detection has been synonymous with sentiment analysis, which measures the positive or negative polarity of documents (Stine, 2019). However, this introduces room for measurement error. The position a document expresses is often different from the sentiment used to express it. For example: "So excited to see Trump out of the White House!" expresses a negative stance about Trump with positive sentiment. Sentiment can potentially inform intensity of stance, but they are distinct dimensions requiring independent measurement (Aldayel and Magdy, 2019).

Bestvater and Monroe (2022) and Sasaki *et al.* (2016) propose targeted sentiment (i.e., sentiment toward Trump rather than general document sentiment) as an operationalization for stance. However, sentiment is one dimension of "attitudes, feelings, judgments, or commitment." If stance is how individuals would answer a proposition, answers need not contain sentiment. The statement "I'm voting for Joe Biden" is a positive response to the proposition "Are you voting for Joe Biden?" but lacks sentiment cues. It may be the dejected expression of progressive liberal who sees Biden as the least-bad option (negative sentiment), or an enthusiastic supporter (positive sentiment). Characterizing the statement as positive sentiment conflates sentiment and stance dimensions and excludes the former possibility. Sentiment can not be both how someone would answer a proposition and how they express that answer.

Accordingly, I use two definitions consistent with the literature and treat sentiment and stance independently:

Definition 1 *Stance:* How an individual would answer a proposition.

Definition 2 *Sentiment:* Positive or negative emotional valence.

Both sentiment and stance may be directed toward individuals, groups, policy, etc., but are distinct.

1.2 What is stance detection?

Recent literature operationalizes stance detection as entailment classification (AlDayel and Magdy, 2021). Introduced by Dagan *et al.* (2005), textual entailment is a directional relationship between two documents called the text (T) and the hypothesis (H).

Definition 3 *Textual entailment:* Text sample T entails hypothesis H when a human reading T would infer that H is most likely true.

Thus, stance detection has two components: First, the proposition on which an author may take a stance (the hypothesis) and second, the text from which a stance is inferred. For example,

if the following tweet from President Trump is paired with the following hypothesis:

T: *It's freezing and snowing in New York—we need global warming!*

H: *Trump supports global warming.* We would conclude text *T* entails hypothesis *H*. Textual entailment does not test if a hypothesis is *necessarily* true, but what humans would infer is most likely true.

AlDayel and Magdy (2021) adopt this definition and define stance detection such that "a text entails a stance to a target if the stance to a target can be inferred from a given text." I adapt their definition:

Definition 4 *Stance detection:* Text sample *T* entails stance *S* to author *A* when a human reading *T* with context *C* would infer that *T* expresses support for *S*.

To their definition, I specify that stance detection identifies the stance a document expresses. This excludes predicting topics like abortion policy preferences or party affiliation from documents not discussing those topics.

Notably, I call attention to the significance of context. Context is information relevant to the stance of interest. Available context consists of information the document contains, and what the classifier knows. Missing context increases document ambiguity, which makes classification harder. Accurate stance detection minimizes ambiguity by ensuring sufficient information between the document's contents and the classifier's knowledge base. Consider the examples in Table 1 from a common stance detection benchmark (Mohammad *et al.*, 2016). Samples were manually labeled as supporting, opposing, or neutral toward Donald Trump. Each example highlights labeling inconsistencies resulting from missing context creating ambiguity. The first does not mention Donald Trump and seemingly reasons that someone espousing certain narratives about the United States' founding probably dislikes Trump. Another labeler may think it is unrelated. Examples two and three express support for the Spanish news station Univision. Labelers for sample two seem aware that Trump feuded with Univision and associate a pro-Univision stance with an anti-Trump stance. Sample three's labelers seemingly lack this context and assigned a different label. Accurate labeling can require time-sensitive knowledge of the context that produced a document. Unless the person or algorithm assigning labels has this information, accurate inference may be impossible.

1.3 Methodological implications

Given the above, there are three methodological implications: First, stance detection is a classification task. This is because stance detection is textual entailment, and textual entailment is classification.

Second, stance is a response to a proposal—the hypothesis *H* in the definition of entailment. Researchers must define what the proposal is, how to determine if a document is relevant to a proposal, and what constitutes support or opposition to that proposal.

Table 1. Text samples from the Semeval 2016 test dataset (Mohammad *et al.*, 2016)

#	Text	Label	Notes
1	@ABC Stupid is as stupid does! Showed his true colors; seems that he ignores that US was invaded, & plundered, not discovered	Against	Based on external knowledge that this is about Trump, potentially inferred from stance about narrative of US founding
2	@peddoc63 @realDonaldTrump So I guess Univision is Fair & Balanced. These are the people that R helping shape USA!	Against	Infers stance toward Trump based on positive stance toward Univision
3	Honestly I am gonna watch #Univision so much more now, just to support the network against	Neutral	Expresses the same stance as example #2, but the labelers apparently did not have the same contextual information when inferring stance

Finally, stances inferred from a document are context dependent. For accurate inference, researchers should try to control context by accounting for what information documents contain, and what information classifiers know. In practical terms, this implies making informed decisions about how documents are sampled or prepared, and choosing the right classification approach. The following section outlines both of these decisions to aid researchers in being more mindful of context and its effects.

2. Controlling context

Here, I outline considerations for selecting and implementing a stance detection approach. To demonstrate the impact of these decisions, I use a dataset labeled for approval of President Trump. The data consist of both public tweets ($n = 1135$) and a random sample of sentences containing the word "Trump" from congressional newsletters sent by the 117th congress ($n = 1000$). The Twitter data were compiled from multiple stance detection projects and represents a variety of labeling approaches and document collection strategies. Details are listed in Appendix 2. Congressional newsletter sentences were collected from the DC Inbox project (Cormack, 2017). Data were coded twice by human annotators and discrepancies were adjudicated by a third annotator.

I use Matthew's Correlation Coefficient (MCC) as the primary performance metric due to its robustness relative to F1 and ROC AUC (Chicco and Jurman, 2023). MCC ranges from −1 to 1 with 0 indicating no correlation between true and estimated classes.

2.1 Document preparation

There are two primary ways to control context via the information documents contain: First, defining document relevancy and second, document pre-processing. Methods for determining document relevancy include keyword matching, topic classification, or using content labels from an API. Stance detection does not necessitate a particular approach and the most appropriate will vary by project. However, researchers should be mindful that by setting inclusion criteria or sampling methods, they control context by setting parameters for what information documents contain. This, in turn, can inform your classification approach by determining what information your model should know.

For example, if classifying stances on abortion I may include documents with the keyword "Roe v. Wade." Most documents will not explicitly state that support for Roe v. Wade implies support for abortion rights. Accordingly, I should ensure my classification approach can make this association.

The second mechanism for controlling context in documents is pre-processing. Most approaches to stance detection use language models that require minimal document pre-processing. Stop word removal, stemming, and other common procedures are generally detrimental for language models (Miyajiwala *et al.*, 2022). Rather, the primary concern is what constitutes a "document." Short documents, like social media posts, may contain singular stance expressions. Longer documents may contain multiple stance expressions across topics. In such circumstances, researchers should segment documents and classify relevant segments to reduce noise.

Dividing documents into coherent segments can be challenging. Existing methods include topic-based approaches (Riedl and Biemann, 2012), Bayesian models (Eisenstein and Barzilay, 2008), and supervised classifiers (Koshorek *et al.*, 2018). Here, I refer to Barberá *et al.* (2021) who compared classification for individual sentences and longer segments (roughly 5 sentences). They examined the trade-off between potentially discarding relevant text when classifying sentences, versus including irrelevant data when classifying sentence groups. They found that classifying sentences reduced noise, but performance between approaches was similar. Accordingly, I advise simplicity. If there are natural segments in your documents (e.g., newspapers leads) multi-

sentence segments may be appropriate. Otherwise, I recommend classifying sentences when segmentation is necessary. Sentence segmentation is relatively simple, and shorter documents can lead to significant computational speedups.

2.2 Model selection

With an understanding of the minimum context your documents include, you can make more informed decisions about selecting a classification method that will bring requisite knowledge to the task. There are three approaches: supervised classification, NLI classification, and in-context classification. Each approach is characterized by its training requirement and the degree of control over classifier knowledge. Supervised classifiers require training data, but offer significant control over model knowledge. NLI and in-context classifiers, however, can label documents without training—an ability known as "zero-shot" classification. NLI classifiers are more efficient but afford less control over model knowledge, while in-context classifiers provide significant knowledge control but require more compute. There is no best approach; rather, the goal is to optimize model performance, computational efficiency, and human workload.

Figure 1 presents a flow chart of key considerations, and the questions below expand on these considerations.

Q1: Are your documents missing context necessary for classification?

NLI classifiers are a good default approach for their accuracy, scalability, and reproducibility as summarized in Table 2. However, they make inferences from a straightforward interpretation of documents and are unaware of context outside the text. For example, If a corpus contains documents about a politician not explicitly named, an NLI classifier is more likely to view those as irrelevant and thus, stance-neutral. Other examples may be a corpus with many documents that require abstract associations (e.g., Table 1), or tasks using a specific operationalization of

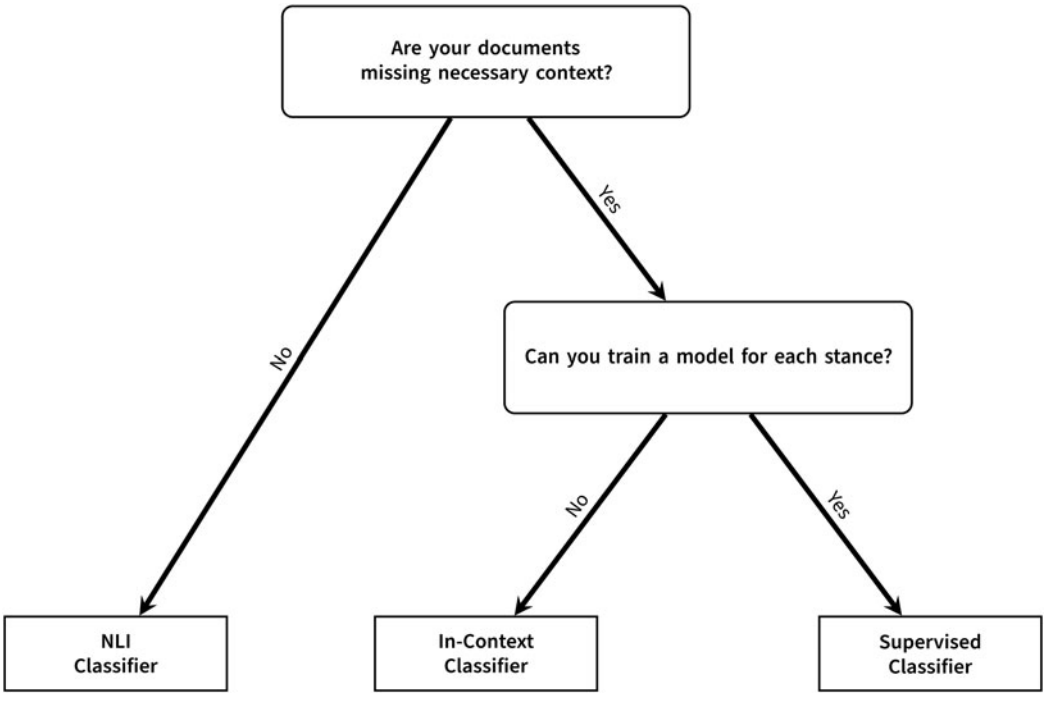

Figure 1. The appropriate classification approach depends on the content of your documents and resource constraints.

Table 2. NLI classifiers are a good starting point for most tasks because they require no training, relatively low compute, and are highly reproducible

Classifier	Requires training	Compute requirements	Reproducibility	Cost (time/money)
Supervised	Yes	CPU or single GPU	High	High
NLI	No	CPU or single GPU	High	Low
In-context	No	GPU or GPU cluster	Low	High

a concept (e.g., a particular definition of "hate"). This can include documents with coded statements, or language requiring specific social knowledge to understand. For example, the phrase "come and take it" expresses support for gun ownership among American conservatives.

There is no empirical definition of sufficient context or how prominent ambiguous documents must be to warrant a different approach. Researchers must qualitatively judge based on familiarity with the data and how document relevancy was determined. However, because NLI classifiers are free and require no training, they are a great, low-commitment starting point. **Q2: Can you train a classifier for each stance you want to label?**

If an NLI classifier is insufficient, supervised and in-context classifiers afford the researcher more control over model knowledge. Supervised classifiers offer better reproducibility and lower computational demands, but are task-specific, potentially requiring multiple models for classifying multiple stance topics. Research on similar tasks suggest a class-balanced training set of 1000–2000 samples at minimum (Margatina *et al.*, 2021; Prabhu *et al.*, 2021; Laurer *et al.*, 2024). If obtaining this many samples per topic is not feasible, consider an NLI or in-context classifier.

In-context classifiers may suit when other methods are infeasible. However, their utility is more limited due to model size and reproducibility concerns. While useful for small datasets, training data compilation, or validating other models, NLI or supervised classifiers are recommended when possible.

2.3 Effects of controlling context

To illustrate the impact of context, I categorized "ambiguous" documents in my test set as those with conflicting human coder assessments. I then determined document relevance by the presence of the term "Trump." Among documents mentioning Trump, 19 percent were ambiguous. Conversely, 34 percent of documents not mentioning Trump were ambiguous.[2]

Differences in ambiguity directly affect classification. As shown in Figure 2, the NLI classifier suffers the most performance degradation when Trump is not mentioned. Supervised and in-context classifiers adapt better to lower context. The implication is not to cull relevant observations due to ambiguity, but that researchers should consider what they know about documents in their corpus and try to address ambiguity with suitable classification methods.

3. Approaches to stance detection

3.1 Supervised classifiers

Supervised classifiers learn patterns within a set of labeled training data and then identify those patterns in the broader corpus. The primary advantage of a supervised classifier is they afford researchers control over what the model "knows" via the training data. This makes them ideal for documents with low contextual information such as the examples in Table 1. If a researcher

[2]Topic classification yielded similar results to keyword matching. 17 percent and 29 percent of documents that were and were not classified as Trump related, respectively, were ambiguous.

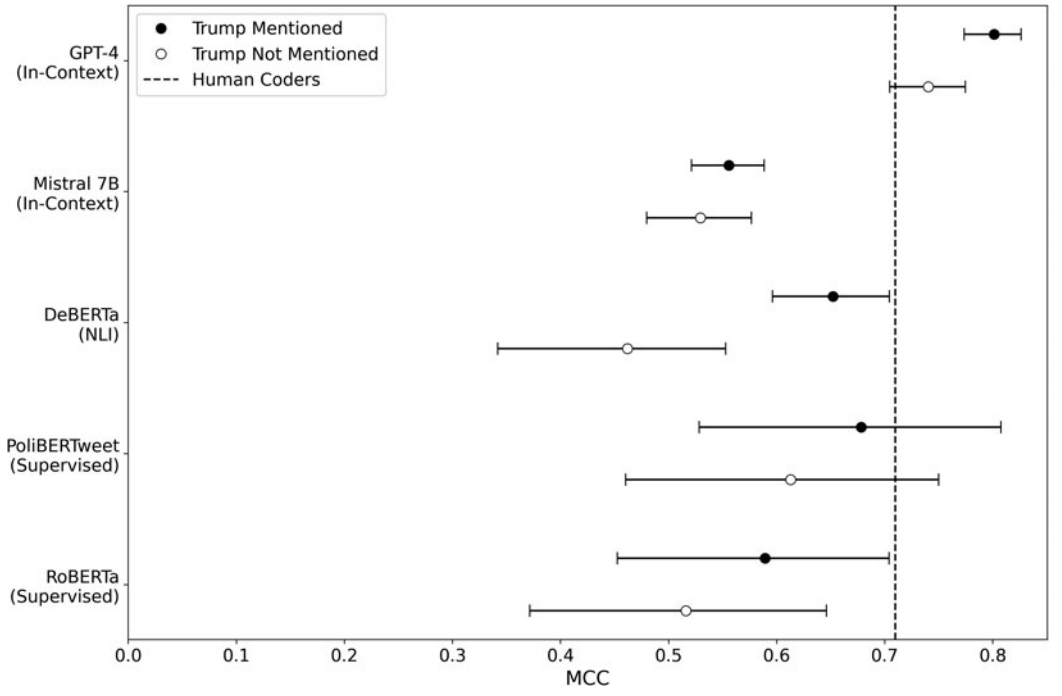

Figure 2. MCC on the testing data with bootstrapped 95 percent confidence intervals. Considering what information your documents contain should inform your approach.

knows that a pro-Univision stance implies an anti-Trump stance, they can teach this association to a supervised classifier through the training data. The primary weakness of supervised classifiers is that they are task specific, requiring additional training for new tasks. This makes them resource intensive for applications encompassing multiple topics or datasets.

Because supervised classifiers are well established in the literature, I do not provide a comprehensive overview of their application here. Rather, I focus on selecting models and training data as these have stance detection specific considerations. For model training, I encourage adherence to best practices established elsewhere in the literature by Laurer *et al.* (2024), Ash and Hansen (2023) and others.

3.1.1 Model selection

Supervised classifiers can be divided into two types: bag-of-words models and language models. The bag-of-words approach uses word counts as features and an algorithm such as logistic regression or a random forest to label documents. These models offer advantages in interpretability and speed (Ash and Hansen, 2023), but have relatively poor performance on entailment classification. Language models are neural networks pre-trained for general language comprehension, and then fine tuned to specific classification tasks—a process known as transfer learning. They are computationally demanding, but are better entailment classifiers.

Model choice has significant performance implications. Through a process called domain-adaptation, language models are pre-trained to learn language particular to a substantive domain such as legal language (Chalkidis *et al.*, 2020) or social media text (Nguyen *et al.*, 2020). When there are no models adapted to a domain of interest, NLI classifiers can be re-trained as supervised classifiers for better performance with less data than other models (Laurer *et al.*, 2024).

Table 3. Performance on detecting support for President Trump in Tweets

Model	MCC	F1	Accuracy (%)	Sweep time	Hardware
Bag-of-words classifiers					
Logistic regression	0.51	0.67	77	**1.53s**	CPU
Random Forest	0.49	0.62	76	2 min. 2 s	CPU
SVM	0.49	0.68	76	37.8 s	CPU
Language models					
RoBERTa	0.60	0.75	81	46 min	GPU
BERTweet	0.63	0.77	83	44 min	GPU
PoliBERTweet	**0.68**	**0.80**	**85**	42 min	GPU

Bold numbers represent the best performing model on a given metric. Language models offer better classification for higher compute demands. Bag-of-words classifiers were trained with an exhaustive grid search and a Bayesian sweep across 10 models was used to train the language models.

To demonstrate the significance of model selection, I trained a set of bag-of-words classifiers and language models. The language models all use the same neural network architecture but vary in their levels of domain adaptation. The first is a base RoBERTa (Liu *et al.*, 2019) model without domain adaptation, the second (BERTweet; Nguyen *et al.*, 2020) is adapted for Twitter, and the third (PoliBERTweet; Kawintiranon and Singh, 2022) is adapted for political Twitter.[3]

Table 3 compares results across models. For each classifier I conducted a hyper-parameter sweep—meaning I trained multiple instances of the model with different parameters to find the best combination. For bag-of-words classifiers I removed a standard set of English stop words, and vectorized documents using term-frequency-inverse-document-frequency and bi-grams. I used an exhaustive hyper-parameter sweep to find the optimal model. Because language models have many more hyper-parameters than other algorithms and take longer to train, an exhaustive sweep is infeasible. Rather, I trained each model ten times using a Bayesian algorithm to more efficiently sweep the parameter space. This approach first trains a model with randomized parameters and adjusts successive models based on results. Results in the table represent the best models. While compute times vary significantly based on data, hardware, and training procedure, the table provides estimates of relative training times in a single CPU or GPU context.

A concern with language models among political scientists is the perception that compute demands are too high (Häffner *et al.*, 2023). However, results here demonstrate models can quickly train on consumer grade GPUs—such as those on free cloud computing platforms. Perhaps more significant than compute time is human labor. Here, too, language models have an advantage. Laurer *et al.* (2024) found that language models with 500 training examples consistently outperformed other algorithms with several thousand. This potentially reduces manual labeling dramatically. Further, converting documents to a bag-of-words requires text pre-processing that increases labor for researchers and alters results (Denny and Spirling, 2018). This includes removing words assumed meaningless, removing suffixes, and dropping words that appear too frequently or infrequently. This necessitates evaluating how pre-processing affects results. Language models, however, minimize arbitrary decision points because they work with unedited documents. Thus, I recommend using language models for stance detection.

The results in Table 3 also demonstrate the impact of domain adaptation. RoBERTa (no domain adaptation) performs the worst ($MCC = 0.60$), while BERTweet (adapted for Twitter) preforms slightly better ($MCC = 0.63$). PoliBERTweet (adapted for political speech on Twitter) preforms the best ($MCC = 0.68$). Additionally, I trained a base DeBERTaV3 model with no domain adaptation to test the significance of model architecture. DeBERTaV3 is an evolution

[3]This section uses Twitter data only.

of the BERT architecture that also uses improved training procedures. The model performed on par with BERTweet ($MCC = 0.63$)—emphasizing that while more sophisticated models have advantages, domain adaptation should not be overlooked.

3.1.2 Training samples

Training data controls the supervised classifiers knowledge base. Researchers should pay attention to how data are sampled and what manual labelers know. Discordance between labelers' knowledge and context required for accurate labeling limits model potential due to noisy training data (Joseph *et al.*, 2017). To bridge the gap between what labelers currently know and what they need to know for accurate annotation, provide coders examples of correctly labeled text (Kawintiranon and Singh, 2021) or a prompt that provides necessary context (Joseph *et al.*, 2017). Crowd sourcing may not be appropriate if the context needed for accuracy is more than people can reasonably internalize in a short time. Here, training expert coders may be appropriate.

3.1.3 Validation

Validating classifiers is done by examining performance on data not seen during training. Two approaches are cross validation and using a test set. Cross validation randomly divides the data into several partitions. The model is trained on all data not in a particular partition, tested on the withheld data, then discarded and the process is repeated for all partitions (Cranmer and Desmarais, 2017). The results across partitions are pooled to estimate a generalized error rate.

A test set approach segments the dataset into a train, validation, and test set. The training set is used to train the model, the validation set monitors training progress, and the test set is withheld for testing after training completes (Cranmer and Desmarais, 2017). This approach offers efficiency advantages over cross validation because it only trains a single model. The train-validation-test split ratio depends on dataset size and desired error estimate precision. Common ratios are 70-15-15 and 60-20-20.

3.2 NLI classifiers

NLI classifiers are language models pre-trained for recognizing textual entailment. Because most classification can be reformulated as an entailment task, NLI classifiers are "universal," rather than task specific, classifiers. The process consists of pairing documents with hypothesis statements, and then assigning labels based on if a document entails a hypothesis (Yin *et al.*, 2019). For example, I pair each document in my data with the hypothesis "The author of this text supports Trump" and the model returns a true or false classification.

3.2.1 Model selection

Currently, the best language model for NLI classification is a DeBERTaV3 Large model trained on NLI datasets. It is the only model that achieves results comparable to humans, and is small enough for consumer hardware (Superglue leaderboard, 2022). I recommend evaluating other models relative to this baseline.

In evaluating models, first consider model size. As shown in Table 4, performance comparable to supervised classifiers only emerges in larger models. Domain adaptation or future advances may reduce the necessary model size. However, between models of the same architecture, larger models generally correlate with better performance.

A second variable to consider is the training data. Stronger models are generally pre-trained on multiple large datasets curated for NLI, such as the ANLI (Nie *et al.*, 2019) and WANLI (Liu *et al.*, 2022). After pre-training, additional tuning on classification datasets can further improve performance. For example, the model trained by Laurer *et al.* (2023) was first pre-trained on multiple NLI datasets, and then tuned on a suite of smaller datasets to help it generalize across tasks.

Table 4. Entailment classification without supervised training requires a relatively sophisticated language model

Model	MCC	F1	Accuracy (%)	Inference time (GPU)	Inference time (CPU)
DeBERTaV3 Small	0.15	0.17	56	3.06 s	2 min 37 s
DeBERTaV3 Base	0.40	0.69	70	4.98 s	5 min 8 s
DeBERTaV3 Large	**0.65**	**0.82**	**83**	13.5 s	16 min 35 s

Bold numbers represent the best performing model on a given metric.

3.2.2 Hypotheses

After selecting a model, a researcher pairs hypothesis statements with documents to classify them. This is generally done by creating a hypothesis template (e.g., "The author of this text…") and finishing the template with the relevant stance expression (e.g., "…supports Trump."). Appropriately pairing hypotheses with documents can dramatically impact performance. Consider the results in Figure 2 that demonstrate a decline in performance when Trump is not mentioned. If a portion of our corpus refers to President Trump as "Mr. President," we might change the hypotheses for those documents from "The author of this text supports of Trump" to "The author of this text supports the President."

Another consideration is how many hypotheses to use. NLI classifiers can be paired with a single hypothesis or a set of hypotheses. In the single hypothesis instance, the model returns a probability of entailment. In the multi-label approach, each hypothesis is classified for entailment and the hypothesis most likely to be entailed is the assigned label. Results may differ slightly between approaches. Consider a multi-hypothesis scenario where the model chooses between "support," "oppose," and "neutral." If the model finds that the entailment probability for each hypothesis is <0.5, it may still return the support label if it is the most likely among the three. In a single "support" hypothesis context, the document would be labeled as not supporting. The multi-label approach will increase computation time roughly linearly with the number of hypotheses since entailment probabilities are predicted for each label. It is recommended to try both approaches and validate results.

3.2.3 Validation

A straightforward approach to validation is to calculate the sample size needed to estimate performance within a confidence interval and label some data. Performance can be estimated with a 5–10 percent margin of error at a 95 percent confidence level with a fraction of the data needed to train a classifier. This can often be accomplished in a few hours.

Researchers should also conduct sensitivity analysis by generating synonymous hypotheses and classifying the data multiple times. This demonstrates results are not sensitive to hypothesis phrasing. For example, I classified my test set with nine synonymous hypotheses (located in Appendix 2) for support of Trump. Across all hypotheses, the average MCC was 0.64, with a minimum value of 0.62 and a maximum value of 0.66.

Classification results from another language model with a different architecture and training set can provide further validation. In-context learners, such as GPT-4, show particular promise here. The results on my test set shown in Figure 2 and Table 5 demonstrate that GPT-4 outpreforms humans in identifying true labels. This could be used to expand the NLI classifier's test set, or as a second validation point against human labeled documents.

3.3 In-context classification

In-context learning refers to the capability of generative language models, such as GPT-4, to learn tasks via plain language prompts describing the task (Brown *et al.*, 2020). Classification involves prompting the model with the task description, document, and a request for an appropriate label. Like supervised classifiers, they can make accurate inferences when the contextual information in

Table 5. Logit biasing can improve performance for no additional cost

Prompt	Model	MCC	F1	Accuracy (%)	Inference time (GPU)	Cost
Standard	GPT-4	0.78	0.88	89	–	$ 12.52
	GPT-3.5 Turbo	0.58	0.73	78	–	$ 0.68
	Mistral 7B	0.56	0.72	78	2 min. 32 s	–
Chain of thought	GPT-4	0.78	0.85	87	–	$ 38.62
	GPT-3.5 Turbo	0.56	0.69	77	–	$ 1.26
	Mistral 7B	0.53	0.72	76	1 hr. 50 min.	–
Logit bias	GPT-4	**0.8**	**0.89**	**90**	–	$ 12.52
	GPT-3.5 Turbo	0.63	0.81	81	–	$ 0.68
	Mistral 7B	0.56	0.72	78	2 min. 33 s	–

Bold numbers represent the best performing model on a given metric. Chain-of-thought reasoning has mixed results, but can significantly increase costs.

documents is low. This is because both the model's pre-training and the user supplied prompt can provide information not contained in the document (OpenAI, 2023a). Like NLI classifiers, their ability to label documents zero-shot means a single model can be used for many tasks.

3.3.1 Reproducibility concerns

Despite their advantages, in-context classifiers are not currently suitable for large-scale classification for two reasons: First, high performing models are currently proprietary models. Researchers cannot archive model versions for replication. Second, these models have costly compute requirements at scale. For reference, I include the cost of classifying my test set with the OpenAI API in Table 5. My test set of 2135 short documents cost $ 12.52 to label with the cheapest approach. This cost quickly balloons as the number and length of documents increases. A dataset of 1, 000, 000 Tweets would cost $ 6260 to label. Open source models that can run on local GPUs are available, but the cost is incurred via compute times. The open source model tested in this paper is an order of magnitude slower than an NLI classifier with no performance benefits.

Due to replication challenges, in-context classifiers have more limited application as classifiers. Where they show the most promise is in labeling smaller datasets, expanding training data for a supervised classifier, or validating other zero-shot classifiers alongside human labels. This technology will hopefully become more reproducible as it matures. Regardless, using non-reproducible models as the primary source of labels should be discouraged (Spirling, 2023).

3.3.2 Model selection

Reproducibility, capability, cost, and speed are the factors to consider when selecting a model. While reproducibility is primarily a function of proprietary status, judging the other factors is less straight forward. I test three different models: GPT-4, GPT-3.5, and Mistral 7B. GPT-4 is among the most capable models available across benchmarks (OpenAI, 2023a). GPT-3.5 is a cheaper alternative that is comparable to other classification methods (OpenAI, 2023a). Finally, Mistral 7B is an open source model that can be archived and run on local computers for replication (Tunstall *et al.*, 2023).

To assess capability, popular models such as GPT-4 have been benchmarked for stance classification here and elsewhere (Törnberg, 2023a). Absent direct testing, performance on shared benchmarks is informative. The HellaSwag benchmark (Zellers *et al.*, 2019), for example, is commonly used to test language inference in generative models. Mistral 7B and GPT3.5 score similarly on HellaSwag (84.52 and 85.5 respectively) and perform similarly in Table 5 (OpenAI, 2023a; Tunstall *et al.*, 2023). GPT-4 scores substantially better (95.3) and stance detection reflects this (OpenAI, 2023a).

Pricing for proprietary models is generally done per token, which are words or sub-words. The prompt, documents, and text generated by the model are all considered in calculating the total

tokens. Estimating the cost of your data can be difficult because exact counts depend on the model used. OpenAI recommends assuming 100 tokens per 75 words (OpenAI, 2023c).

Classification speed with proprietary models is determined by API rate limits and traffic.[4] With open source models, inference time varies based on document length, prompt length, and parameters used. At its fastest, Mistral 7B took roughly 2.5 min to classify the test set on a consumer grade GPU. With a longer prompt and different parameters, the same task took approximately 2 h.

3.3.3 Prompt engineering

Results vary based on the model prompt. Optimal prompt construction is an inexact science, but several validated approaches enchance in-context classification efficacy. Regardless of approach, prompt engineering should involve testing multiple prompts and validating results on a small labeled dataset.

Prompts have two components: the system message and the user message. The system message instructs the model on how to behave and respond to prompts. The user message is the prompt the model responds to. User messages should explain contextual information necessary for accurate inference and present the document to be classified. The prompt should be formatted to distinctly mark the document requiring classification (OpenAI, 2023b). Listing 1 provides an example prompt. The system message specifies the task and the range of desired responses. The user message provides context by specifying that the text concerns Donald Trump, reiterates the acceptable responses, and demarcates the text for classification.

One approach to prompting is few-shot learning. Few-shot prompts include example documents and their correct labels—similar to training data. This approach has mixed results in improving classification (Brown *et al.*, 2020; Kristensen-McLachlan *et al.*, 2023). Few-shot learning's challenge is generalizing from a small, potentially non-representative sample. This risks overfitting to prompted examples (Wang *et al.*, 2020). Thus, results can be variable depending on provided examples and arbitrary changes in the prompt (Lu *et al.*, 2021; Perez *et al.*, 2021). These instabilities persist with more sophisticated models and more labeled examples (Zhao *et al.*, 2021). Thus, while few-shot prompts can improve classification in some cases, I recommend using them with caution—particularly when zero-shot performance is sufficient.

Another prompting method is chain-of-thought reasoning (Wei *et al.*, 2022). This approach asks the model to explain its reasoning step-by-step before providing the conclusion. It improves perform-

```
system_message = "You are a text classifier and are only allowed to respond with 1
    or 0."
user_message = "You are a classifier that determines if the author of a text
    supports Donald Trump. I will post text about Trump and if the author is more
    likely to support Trump than not return 1. If it is not more likely that the
    author supports Trump return 0. Do not explain the classification or say
    anything else. You are only allowed to respond with 1 or 0. Here is the
    text.\nText: {}"

messages = [
    {"role": "system", "content": system_message},
    {"role": "user", "content": user_message.format(document)}
]
```

Listing 1. An example prompt template in the the Python OpenAI package. The system message gives the model an "identity" and the user message explains the task and presents the document to be classified.

[4]The test set was classified multiple times via the OpenAI API. Times ranged from 35 min to 6.5 h. Speed was not model dependent and driven by API traffic.

ance on a variety of tasks, including classification, but not consistently so (He *et al.*, 2023). Chain-of-thought prompting increases costs significantly because the model generates an explanation in addition to the label. Evaluating chain-of-thought responses can be cumbersome because labels must be extracted from the rest of the response. Requesting models follow response templates with predictable label locations can help, though language models may inconsistently adhere to them.

I evaluated chain-of-thought reasoning by constructing a prompt modeled after the one used by He *et al.* (2023). I found it offered no perceivable benefit in accuracy, but tripled the cost of GPT-4. I further found different compliance rates with the requested response template. GPT-4 showed 100 percent compliance, while GPT-3.5 had 98.8 percent compliance. Mistral 7B had 87 percent compliance, but saw a decrease in performance and took roughly 44 times longer to complete the task.

3.3.4 Decoding strategies

The process of generating text from prompts is called decoding. There are many approaches models use, but in the context of classification the objective is to produce deterministic and reproducible results. This involves constraining the model's potential output tokens. There are three approaches to this: adjusting the temperature parameter, prompt construction, and biasing predicted token probabilities.

Temperature can be conceptualized as a randomization parameter. Generative models randomly sample the words they generate based on estimated probabilities (Welleck *et al.*, 2020). The temperature parameter controls the shape of the distribution words are sampled from. A higher temperature flattens the distribution toward a uniform one where each word is equally likely as $T_{\to\infty}$ (Ackley *et al.*, 1985). A temperature of zero ensures the most likely token is always chosen. Thus, I recommend setting temperature to zero.

Prompts can restrict the tokens models generate by specifying the desired response range. For example, I can instruct the model to only respond with "1" or "0." However, compliance will depend on model sophistication. GPT-4, for example, was 100 percent compliant when asked to answer with only a "1" or "0." Mistral 7B showed less instruction-aligned results and required more prompt engineering for consistent binary labels on the test set.

To directly restrict the tokens models generate, users can manipulate token prediction scores, or "logits." This approach applies a flat bias to the probabilities of tokens representing classes so that their predicted probability is much greater than other tokens. Table 5 shows that logit biasing generally has a positive effect on classification. Additionally, it simplifies label extraction by reducing non-compliant responses.

3.3.5 Validation

In-context classifiers should be validated with a test set. Calculate the number of samples needed to estimate performance within a desired confidence interval and use labeled data to test prompts and model parameters. Researchers should demonstrate that results are robust to synonymous prompts and not sensitive to arbitrary changes. Thus, researchers may want to classify their data with multiple prompts. An NLI classifier can also provide a second point of validation beside a sample of human labels. This demonstrates that two models with different architectures and training procedures obtain similar results.

4. Replication: COVID-19 threat minimization

An analysis of Twitter posts by Block *et al.* (2022) showed that minimizing the threat of COVID-19 was in part ideologically motivated, but that the effect of ideology decreased as deaths within a person's geographic area increased. The original study used a supervised classifier to label documents. For this replication, I demonstrate how NLI classifiers can obtain similar results, as well as basic sensitivity analysis for zero-shot classification.

4.1 Data and design

The replication data contain 862 923 tweets about COVID-19 posted between September 1, 2020 and February 28, 2021 from 23,476 unique users. Each user's ideology is measured on a left-right scale using tweetscores, a network-based ideal point estimation (Barberá, 2015). Included are 2,000 hand labeled training tweets and coding rules for labeling tweets "non-compliant." Non-compliant tweets include statements that downplay the threat of COVID-19, as well as statements against mitigation practices (e.g., vaccination).

Based on the coding rules, I created two sets of synonymous entailment hypotheses with compliant, non-compliant, and neutral stance statements. A complete set of hypotheses is listed in Appendix 2. I then used keywords to match hypotheses to tweets and classified the remaining corpus with a DeBERTaV3 NLI classifier. If a tweet entails any of the threat-minimizing hypotheses it is considered threat-minimizing. For example, a tweet that contains the words "mask" and "vaccine" will be classified once for the hypotheses associated with masks and again for hypotheses associated with vaccines. If the model determines the document entails an anti-mask or anti-vaccine hypothesis it is considered threat-minimizing. I use 300 randomly sampled tweets from the training data to test the hypotheses—enough data to estimate performance at a 95 percent confidence level with a <5 percent margin of error, and a small enough sample that a single researcher could label the data in a few hours.

To analyze results I replicate the model used by Block *et al.* (2022)—a negative binomial regression with a count of threat-minimizing tweets made as the dependent variable. The independent variables are the user's ideology, the COVID-19 death rate within their county, and an interaction between the two. Control variables including county-level demographics, political lean, and state fixed effects are also used.

4.2 Analysis

Using the NLI classifier on my sample of 300 labeled tweets, the first set of hypotheses had an MCC of 0.67 and 87 percent accuracy while the second had an MCC of 0.74 and 90 percent

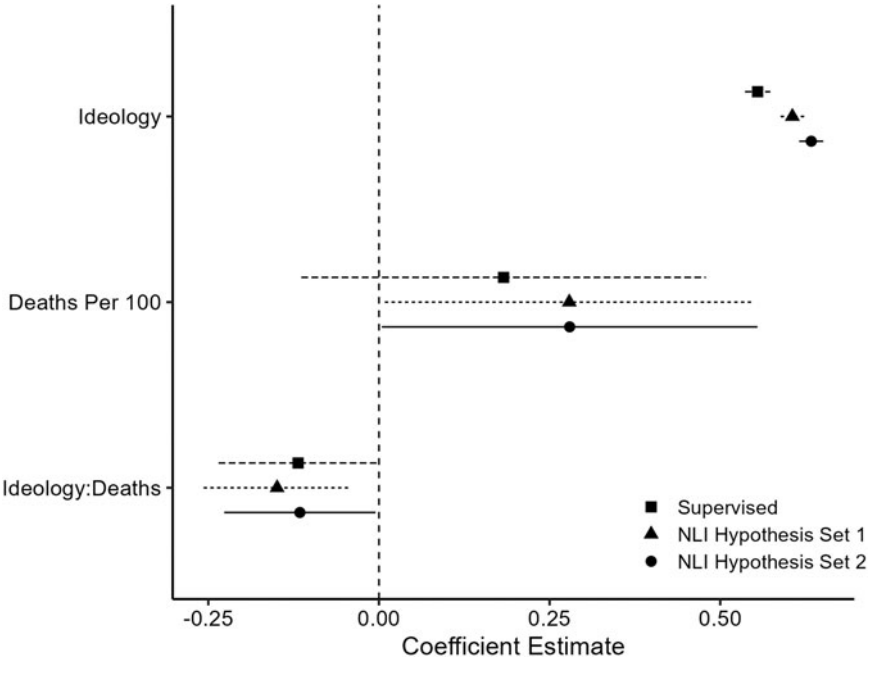

Figure 3. Replication results from Block *et al.* (2022). Supervised represents the original results from a trained Electram model, while the NLI hypothesis sets used a DeBERTaV3 model for zero-shot classification.

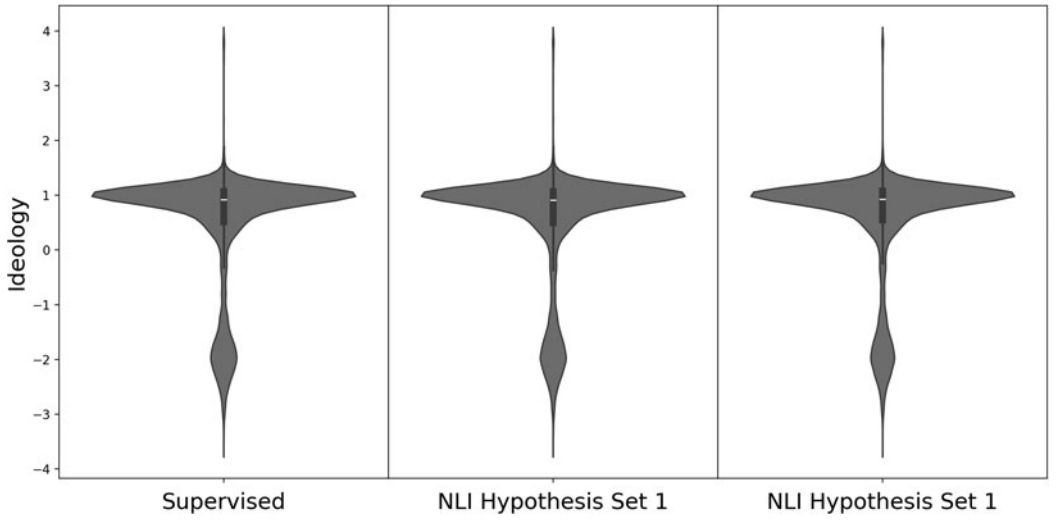

Figure 4. The distribution of tweet author ideology among tweets labeled threat-minimizing by the three classifiers.

accuracy. The original classifier achieved an MCC of 0.66 and accuracy of 86 percent on the test set. Between the models there was a high degree of intercoder reliability ($MCC = 0.75$). Figure 3 shows the results from regression models that use zero-shot labels from both sets of hypotheses, as well as the original labels from the supervised classifier. The three models show consistency in both the direction and size of the effect.

Figure 4 shows the ideological distribution of all tweets labeled threat-minimizing across the three models. The distributions appear identical with each identifying a concentration of threat-minimizing tweets on the conservative pole.

Across the entire sample, the average ideology of a tweet's author is −0.66. Among samples labeled differently by the zero-shot classifier and the original classifier, the average ideology was –0.11 for the first set of hypotheses, and −0.07 for the second set of hypotheses—indicating the NLI classifiers are slightly more likely to disagree with the supervised model when the author is more conservative. However, the models show a high level of consistency across the data in identifying a relatively nuanced "threat minimization" stance. This provides compelling evidence that zero-shot NLI models can be a useful avenue for stance detection.

5. Conclusion

In this paper, I outlined a precise definition and generalized framework for stance detection. Stance detection is entailment classification and determines how a document responds to a proposition. After defining stance proposition, there are currently three possible classification approaches: supervised classification, NLI classification, and in-context classification. Researchers have two primary considerations to determine the right approach: What information is necessary for accurate classifications, and how to balance trade-offs between resources, computing demand, and human labor. When the information documents contain is sufficient for accurate classification, NLI classifiers achieve performance comparable to supervised classifiers while eliminating the need to train a model. When context outside of the document is necessary for classification, supervised and in-context classifiers are a better choice.

A potentially fruitful direction for future research is in adapting NLI classifiers to better understand political text. Advancements in NLI classification are largely from computer science, and the data models are trained on reflect this. There are no NLI datasets specifically for stance

detection, or stance detection benchmarks. Datasets and pre-trained models with political communication in mind could mean more reliable zero-shot classification with more accessible compute demands.

Finally, stance detection methods can be advanced by moving beyond classification and toward scaling the intensity of a stance. This could possibly be achieved by integrating sentiment measures, but more research is needed.

Supplementary material. The supplementary material for this article can be found at https://doi.org/10.1017/psrm.2024.35. To obtain replication material for this article, https://doi.org/10.7910/DVN/XS0ULP

Acknowledgements. I would like to thank Michael Nelson, Bruces Desmarais, Suzie Linn, and Kevin Munger for their feedback on this manuscript.

Competing interests. None.

References

Ackley DH, Hinton GE and Sejnowski TJ (1985) A learning algorithm for Boltzmann machines. *Cognitive Science* **9**, 147–169.

Aldayel A and Magdy W (2019) Assessing sentiment of the expressed stance on social media. In *Social Informatics: 11th International Conference, SocInfo 2019, Doha, Qatar, November 18–21, 2019, Proceedings 11*, Springer, pp. 277–286.

AlDayel A and Magdy W (2021) Stance detection on social media: state of the art and trends. *Information Processing & Management* **58**, 102597.

Ash E and Hansen S (2023) Text algorithms in economics. *Annual Review of Economics* **15**, 659–688.

Barberá P (2015) Birds of the same feather tweet together: Bayesian ideal point estimation using Twitter data. *Political Analysis* **23**, 76–91.

Barberá P (2016) Less is more? How demographic sample weights can improve public opinion estimates based on Twitter data. *Work Pap NYU*.

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

Bestvater SE and Monroe BL (2022) Sentiment is not stance: target-aware opinion classification for political text analysis. *Political Analysis* **31**, 1–22.

Biber D and Finegan E (1988) Adverbial stance types in English. *Discourse Processes* **11**, 1–34.

Block R Jr, Burnham M, Kahn K, Peng R, Seeman J and Seto C (2022) Perceived risk, political polarization, and the willingness to follow COVID-19 mitigation guidelines. *Social Science & Medicine* **305**, 115091.

Bor A, Jørgensen F and Petersen MB (2023) Discriminatory attitudes against unvaccinated people during the pandemic. *Nature* **613**, 704–711.

Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh Aditya, Ziegler D, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I and Amodei D (2020) Language models are few-shot learners. *Advances in Neural Information Processing Systems* **33**, 1877–1901.

Chalkidis I, Fergadiotis M, Malakasiotis P, Aletras N and Androutsopoulos I (2020) LEGAL-BERT: The Muppets straight out of Law School. preprint arXiv:2010.02559.

Chicco D and Jurman G (2023) The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification. *BioData Mining* **16**, 1–23.

Cormack L (2017) DCinbox—capturing every congressional constituent e-newsletter from 2009 onwards. *The Legislative Scholar* **2**, 27–34.

Cranmer SJ and Desmarais BA (2017) What can we learn from predictive modeling?. *Political Analysis* **25**, 145–166.

Dagan I, Glickman O and Magnini B. (2005) The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*. Berlin, Heidelberg: Springer, pp. 177–190.

Denny MJ and Spirling A (2018) Text preprocessing for unsupervised learning: why it matters, when it misleads, and what to do about it. *Political Analysis* **26**, 168–189.

Eisenstein J and Barzilay R (2008) Bayesian unsupervised topic segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 334–343.

Felmlee DH, Blanford JI, Matthews SA and MacEachren AM (2020) The geography of sentiment towards the women's march of 2017. *PLoS ONE* **15**, e0233994.

Gilardi F, Alizadeh M and Kubli M (2023) ChatGPT outperforms crowd-workers for text-annotation tasks arxiv:2303.15056.

Häffner S, Hofer M, Nagl M and Walterskirchen J (2023) Introducing an interpretable deep learning approach to domain-specific dictionary creation: a use case for conflict prediction. *Political Analysis* **0**, 1–19.

He X, Lin Z, Gong Y, Jin A, Zhang H, Lin C, Jiao J, Yiu SM, Duan N and Chen W (2023) AnnoLLM: making large language models to be better crowdsourced annotators, preprint arXiv:2303.16854.
Hutto C and Gilbert E (2014) Vader: a parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, Vol. 8, pp. 216–225.
Joseph K, Friedland L, Hobbs W, Tsur O and Lazer D (2017) Constance: modeling annotation contexts to improve stance classification. preprint arXiv:1708.06309.
Kawintiranon K and Singh L (2021) Knowledge enhanced masked language model for stance detection. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
Kawintiranon K and Singh L (2022) Polibertweet: a pre-trained language model for analyzing political content on Twitter. In *Proceedings of the Language Resources and Evaluation Conference*. European Language Resources Association.
Koshorek O, Cohen A, Mor N, Rotman M and Berant J (2018) Text segmentation as a supervised learning task, preprint arXiv:1803.09337.
Kristensen-McLachlan RD, Canavan M, Kardos M, Jacobsen M and Aarøe L (2023) Chatbots are not reliable text annotators. preprint arXiv:2311.05769.
Laurer M, van Atteveldt W, Casas A and Welbers K (2023) Building efficient universal classifiers with natural language inference.
Laurer M, Van Atteveldt W, Casas A and Welbers K (2024) Less annotating, more classifying: addressing the data scarcity issue of supervised machine learning with deep transfer learning and BERT-NLI. *Political Analysis* **32**, 84–100.
Lee E, Rustam F, Washington PB, El Barakaz F, Aljedaani W and Ashraf I (2022) Racism detection by analyzing differential opinions through sentiment analysis of tweets using stacked ensemble GCR-NN model. *IEEE Access* **10**, 9717–9728.
Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L and Stoyanov V (2019) Roberta: a robustly optimized BERT pretraining approach. preprint arXiv:1907.11692.
Liu A, Swayamdipta S, Smith NA and Choi Y (2022) WANLI: Worker and AI Collaboration for Natural Language Inference dataset creation. preprint arXiv:2201.05955.
Liu B (2010) Sentiment analysis and subjectivity. *Handbook of Natural Language Processing* **2**, 627–666.
Lu Y, Bartolo M, Moore A, Riedel S and Stenetorp P (2021) Fantastically ordered prompts and where to find them: overcoming few-shot prompt order sensitivity. preprint arXiv:2104.08786.
Margatina K, Barrault L and Aletras N (2021) On the importance of effectively adapting pretrained language models for active learning. preprint arXiv:2104.08320.
Miyajiwala A, Ladkat A, Jagadale S and Joshi R (2022) On sensitivity of deep learning based text classification algorithms to practical input perturbations. In *Science and Information Conference*, Springer, pp. 613–626.
Mohammad S, Kiritchenko S, Sobhani P, Zhu X and Cherry C (2016) Semeval-2016 task 6: detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 31–41.
Nguyen DQ, Vu T and Nguyen AT (2020) BERTweet: a pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 9–14.
Nie Y, Williams A, Dinan E, Bansal M, Weston J and Kiela D (2020) Adversarial NLI: a new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.
OpenAI (2023a) Gpt-4 technical report. arXiv:abs/2303.08774.
OpenAI (2023b) Prompt engineering. https://platform.openai.com/docs/guides/prompt-engineering.
OpenAI (2023c) What are tokens and how to count them? https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them.
Osmundsen M, Bor A, Vahlstrup PB, Bechmann A and Petersen MB (2021) Partisan polarization is the primary psychological motivation behind political fake news sharing on Twitter. *American Political Science Review* **115**, 999–1015.
Perez E, Kiela D and Cho K (2021) True few-shot learning with language models. *Advances in Neural Information Processing Systems* **34**, 11054–11070.
Prabhu S, Mohamed M and Misra H (2021) Multi-class text classification using BERT-based active learning. preprint arXiv:2104.14289.
Riedl M and Biemann C (2012) TopicTiling: a text segmentation algorithm based on LDA. In *Proceedings of ACL 2012 Student Research Workshop*, pp. 37–42.
Sasaki A, Mizuno J, Okazaki N and Inui K (2016) Stance classification by recognizing related events about targets. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, IEEE, pp. 582–587.
Spirling A (2023) Why open-source generative AI models are an ethical way forward for science. *Nature* **616**, 413–413.
Stine RA (2019) Sentiment analysis. *Annual Review of Statistics and its Application* **6**, 287–308.
Superglue leaderboard (2022) leaderboard accessed August 25, 2022.
Terman R (2017) Islamophobia and media portrayals of Muslim women: a computational text analysis of US news coverage. *International Studies Quarterly* **61**, 489–502.
Törnberg P (2023a) ChatGPT-4 outperforms experts and crowd workers in annotating political Twitter messages with zero-shot learning. preprint arXiv:2304.06588.

Törnberg P (2023b) How to use LLMs for text analysis. preprint arXiv:2307.13106.
Tunstall L, Beeching E, Lambert N, Rajani N, Rasul K, Belkada Y, Huang S, von Werra L, Fourrier C, Habib N, Sarrazin N, Sanseviero O, Rush A and Wolf T (2023) Zephyr: direct distillation of LM alignment. preprint arXiv:2310.16944.
Wang Y, Yao Q, Kwok JT and Ni LM (2020) Generalizing from a few examples: a survey on few-shot learning. *ACM Computing Surveys (CSUR)* **53**, 1–34.
Wei J, Wang X, Schuurmans D, Bosma M, Xia F, Chi E, Le QV and Zhou D (2022) Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* **35**, 24824–24837.
Welleck S, Kulikov I, Kim J, Pang RY and Cho K (2020) Consistency of a recurrent language model with respect to incomplete decoding. preprint arXiv:2002.02492.
Yin W, Hay J and Roth D (2019) Benchmarking zero-shot text classification: datasets, evaluation and entailment approach. preprint arXiv:1909.00161.
Zellers R, Holtzman A, Bisk Y, Farhadi A and Choi Y (2019) HellaSwag: Can a machine really finish your sentence?. preprint arXiv:1905.07830.
Zhao Z, Wallace E, Feng S, Klein D and Singh S (2021) Calibrate before use: improving few-shot performance of language models. In *International Conference on Machine Learning*, PMLR, pp. 12697–12706.

Cite this article: Burnham M (2024) Stance detection: a practical guide to classifying political beliefs in text. *Political Science Research and Methods* 1–18. https://doi.org/10.1017/psrm.2024.35