Political Analysis (2025), **00**, 1–17
doi:10.1017/pan.2025.10017

PA

ARTICLE

Codebook LLMs: Evaluating LLMs as Measurement Tools for Political Science Concepts

Andrew Halterman[1] and Katherine A. Keith[2]

[1]Department of Political Science, Michigan State University, East Lansing, MI, USA; [2]Department of Computer Science, Williams College, Williamstown, MA, USA

Corresponding author: Andrew Halterman; Email: halterm3@msu.edu

(Received 11 September 2024; revised 12 February 2025; accepted 13 February 2025)

Abstract

Codebooks—documents that operationalize concepts and outline annotation procedures—are used almost universally by social scientists when coding political texts. To code these texts automatically, researchers are increasingly turning to generative large language models (LLMs). However, there is limited empirical evidence on whether "off-the-shelf" LLMs faithfully follow real-world codebook operationalizations and measure complex political constructs with sufficient accuracy. To address this, we gather and curate three real-world political science codebooks—covering protest events, political violence, and manifestos—along with their unstructured texts and human-coded labels. We also propose a five-stage framework for codebook-LLM measurement: Preparing a codebook for both humans and LLMs, testing LLMs' basic capabilities on a codebook, evaluating zero-shot measurement accuracy (i.e., off-the-shelf performance), analyzing errors, and further (parameter-efficient) supervised training of LLMs. We provide an empirical demonstration of this framework using our three codebook datasets and several pre-trained 7–12 billion open-weight LLMs. We find current open-weight LLMs have limitations in following codebooks zero-shot, but that supervised instruction-tuning can substantially improve performance. Rather than suggesting the "best" LLM, our contribution lies in our codebook datasets, evaluation framework, and guidance for applied researchers who wish to implement their own codebook-LLM measurement projects.

Keywords: Text as data; Large Language Models; Automated content analysis; Computational methods; Natural language processing

Edited by: Daniel J. Hopkins and Brandon M. Stewart

1. Introduction

Political scientists are increasingly turning to generative large language models (LLMs) for text analysis due in part to their potential to classify documents with no labeled examples, i.e., "zero-shot" (Atreja *et al.* 2025; Egami *et al.* 2023; Gilardi, Alizadeh, and Kubli 2023; Heseltine and Clemm von Hohenberg 2024; Peskoff and Stewart 2023; Rytting *et al.* 2023; Ziems *et al.* 2024). Researchers typically provide an LLM with a set of categorical labels (e.g., protest or riot), (sometimes) brief label descriptions, and instructions to classify documents. An LLM then generates the predicted (natural-language) label of each document (e.g., the LLM generates "protest"). While this approach enables cheap and fast analysis of large text corpora, it implicitly assumes that these simple instructions alone operationalize the precise concepts that researchers aim to measure from text. Because LLMs are pre-trained on extremely large text corpora, the models often have reasonable zero-shot predictive accuracy on concepts that have broad coverage and straightforward definitions in their pre-training data.

However, this use of LLMs is in tension with foundational concepts of measurement in political science. In the standard conception of measurement validity (Adcock and Collier 2001), researchers

must transform broad "background" concepts into precise, systematized constructs through careful operationalization. These operationalizations are often recorded by researchers in codebooks, documents that define concepts in precise terms and outline annotation procedures. As a brief example, different projects systematize "protest" in substantively different ways. For instance, the Crowd Counting Consortium (2024)'s definition of "protest" excludes labor strikes, while the CAMEO event ontology (Gerner *et al.* 2002)'s definition of "protest" includes them. Suppose, we choose the CAMEO definition of a protest, but our LLM has learned an internal representation of "protest" from contexts in which the word occurred in its pre-training data and these contexts included strikes. If we use the LLM zero-shot, we may have many "protest" measurements that include labor strikes and lead to incorrect substantive conclusions.

This mismatch between measurement theory and current LLM use arises from two potential sources. First, analysts often provide an LLM with only simple category labels or brief definitions rather than complete codebooks. Second, even when provided the full codebook, an LLM may not actually attend to the detailed label definitions in the codebook. In both scenarios, an LLM may generate labels for documents without updating its internal representation of what the label means based on the codebook, and the output will likely correspond to a more general background concept rather than the systematized construct, threatening measurement validity.

To address this mismatch, we introduce a framework to rigorously evaluate and improve *codebook-LLM measurement* and provide an empirical demonstration of the framework. We make four main contributions. First, our five-stage evaluation framework guides researchers through codebook preparation, label-free testing, zero-shot evaluation, robustness evaluations, and training LLMs on supervised examples. Within these stages, we contribute seven new codebook-specific *behavioral tests* (Ribeiro *et al.* 2020)—controllable interventions on LLM inputs that test LLMs basic capabilities and failure modes. Many of these require no ground-truth labels so can be used early in a project's lifecycle. Second, we collect, format, and release three real-world political science codebooks, their associated hand-labeled datasets, and source texts covering contentious politics in the United States, political violence in Pakistan, and party manifestos. The codebook settings are realistic but potentially challenging for LLMs due to their domain-specific concepts, large number of class labels (up to 142), and lengthy codebook definitions. Third, we use several open-weight LLMs to classify documents from these datasets, and find limitations in select LLMs' ability to comply with codebook instructions in zero-shot settings. Finally, we show that supervised *instruction-tuning*—updating LLM weights directly from human-coded examples—can substantially improve performance (by up to 55%), and provide detailed guidance for researchers who wish to implement parameter-efficient versions of this training.

2. A Framework for Codebook-LLM Measurement

In this section, we provide an overview of our framework to evaluate *codebook-LLM measurement*, classifying unlabeled documents with LLMs given a human-written codebook. For applied readers new to this space, we provide a glossary of technical terms in SI A in the Supplementary Material. In this work, we focus on multi-class (single-label) classification of N documents, $\{\hat{Z}_i \in \mathcal{Z} | X_i\}_{i=1}^{N}$, where $\mathcal{Z}$ is the finite set of labels described in the codebook, $\hat{Z}_i$ is an LLM-predicted class of a document i, and X_i is the document text. Most often these measurements, $\hat{Z}_i$, are then used in downstream analysis (Knox, Lucas, and Cho 2022). Although, recent work has shown one can adjust for noisy measurement-induced bias in downstream analysis (Egami *et al.* 2023), more accurate measurement will often yield more precise estimates (and thus tighter valid confidence intervals).

Our proposed codebook-LLM measurement evaluation framework consists of a preparatory stage then four main stages. The rest of this article expands on each stage.

- *Stage 0: Codebook preparation*: Even when using LLMs, researchers must prepare their codebook by operationalizing their social science concepts. We suggest a semi-structured format with standardized components including definitions, clarifications, and examples that can be used by both humans and machines.

- *Stage 1: Label-free behavioral testing*: Before investing in hand-labeling, we suggest researchers use our proposed label-free behavioral tests to assess an LLM's ability to follow instructions and make consistent predictions "off-the-shelf". The results can help guide both LLM selection and the choice between zero-shot use and supervised learning.
- *Stage 2: Zero-shot evaluation with labels*: The next stage requires researchers to hand-label a small evaluation set and then assess LLM zero-shot performance. Quantitative results can reveal weaknesses in either the codebook or the LLM's ability to predict from it, potentially requiring codebook revision or a different LLM.
- *Stage 3: Zero-shot error analysis*: We suggest thorough error analysis using ablation experiments, behavioral tests requiring labels, and manual analysis of outputs to understand systematic errors.
- *Stage 4: Supervised fine-tuning*: If zero-shot performance proves inadequate, researchers can update the LLM weights directly via instruction-tuning on human-coded examples. We provide guidance on parameter-efficient techniques to make this computationally feasible.

The stages can be performed sequentially or iteratively as needed, with results from later stages potentially informing revisions to earlier decisions. The rest of the article details each stage.

3. Related Work

Recent work has demonstrated LLMs' potential for classifying social science concepts. Early zero-shot evaluations showed promise: Halterman *et al.* (2021) achieved F1 scores up to 0.74 using entailment models for political event classification, while Ziems *et al.* (2024)'s evaluation across 25 computational social science benchmarks found respectable zero-shot LLM performance on some tasks. Several studies found LLM's annotation abilities to be at human-level: Gilardi *et al.* (2023) showed ChatGPT outperformed crowd workers on tasks like stance detection and Mellon *et al.* (2024) found similar accuracy to human annotators when coding survey responses. Some researchers have explored using LLMs indirectly for classification: Halterman (2025) used LLMs to generate synthetic training data and Pangakis and Wolken (2024) used LLMs to produce labels to train downstream classifiers. However, most of this research relied on providing LLMs with a set of labels to choose from, without detailed coding instructions.

Our work builds more directly on previous work that examines LLMs' responsiveness to prompts and ability to apply precise category definitions. Atreja *et al.* (2025) vary prompt components across four social science tasks, finding that both task characteristics and prompt design substantially affect performance. Burnham (2025) emphasized the importance of providing LLMs with detailed definitions and broader context. Most directly relevant to our work, Thalken *et al.* (2023) found that even state-of-the-art LLMs struggle to apply a three-category legal reasoning codebook, though fine-tuned models showed better performance.

After obtaining predicted document labels (likely from LLMs), another line of work has examined the consequences of using these noisy labels—labels that are not perfectly accurate—in downstream inference (Chen, Bhattacharya, and Keith 2024; Egami *et al.* 2023; Knox *et al.* 2022). Specifically, Egami *et al.* (2023) find that even when LLMs achieve decent labeling accuracy, using these noisy labels in downstream analysis can induce severe bias and coverage issues and suggest a correction. In this work, we focus on studying LLMs' accuracy with complex categories, but note that even highly accurate labels may still result in estimation issues, and point applied researchers to this existing work.

Finally, both academic and industry researchers have developed a large set of benchmarks for evaluating LLM performance. While our work proposes new codebook-specific LLM evaluation techniques, we leave a detailed discussion of existing LLM benchmarks to SI C in the Supplementary Material.

4. Empirical Set-Up: Data and LLMs

4.1. Codebook Datasets

To serve as real-world training and evaluation datasets, we collect three separate English-language political science datasets, each of which provide document-level or (quasi-) sentence-level labels

Table 1. Descriptive statistics about the codebook datasets.

Dataset	BFRS	CCC	Manifestos
No. of classes	12	8	142
Per-class definition median whitespace toks	20	28	14
Codebook total whitespace toks	1,614	608	3,910
Codebook total Llama toks	2,083	721	5,145
Input text (doc.) median whitespace toks	28	437	16
No. of train instances	20,978	4,710	8,081
No. of dev. instances	4,495	1,009	1,732
No. of test instances	4,496	1,010	1,732

Note: For length of the codebooks, we report both whitespace tokens and number of tokens after using the Llama-3 tokenizer (Llama toks.).

according to a codebook-defined schema: the Crowd Counting Consortium (CCC) dataset on protests in the United States (Crowd Counting Consortium 2024), the BFRS dataset on violence in Pakistan (Bueno de Mesquita *et al.* 2015), and the Manifesto Project corpus (Lehmann *et al.* 2017). We obtain their original codebooks and a selection of the English-language text that the data was coded from (see SI Section D for additional details). We believe this collection reflects the real-world difficulty of the codebook-LLM measurement due to the datasets' (1) construction by social scientists to measure specific political concepts that potentially do not exist in LLMs supervised training data; (2) large number of classes (up to 142; see Table 1); (3) longer documents (in the CCC dataset), and (4) long codebook lengths.

In our curation effort, we reformat these datasets for single-label, multi-class classification. For the Manifestos dataset, we use the Manifesto Project issue category label for a (quasi) sentence in a political party's manifesto. For BFRS and CCC, we focus solely on classifying event types from news articles. We randomly split the data 70-15-15 into training, development, and test splits, respectively. In this work, we only use the training and development sets to guard against overfitting to the test set.

SI D in the Supplementary Material provides more details on the dataset preprocessing, including the process of scraping or obtaining raw text, sampling documents, converting multi-label examples to single-label, multi-class labels, and in the case of BFRS, using an alternative text source. We believe that all three of these datasets are mostly safe from training set contamination, that is, that they are not present in the LLM's pre-training data (see SI D.2 in the Supplementary Material).

4.2. Choosing LLMs

Our evaluation framework is model-agnostic, but we constrain our empirical demonstration to open-weight LLMs for reproducibility and to models that fit on a consumer GPU (24GB VRAM). We select four high-performing LLMs in the 7–12 billion (B) parameter range: Mistral-7B-Instruct-v0.2 ("Mistral-7B"), Mistral-NeMo-Instruct-2407 ("Mistral-NeMo-12B"), Llama-3.1-8B-Instruct ("Llama-8B"), and OLMo-7B-0724-Instruct-hf ("OLMo-7B"). SI Section F provides details and citations for these models. We emphasize that we select open weight models to ensure reproducibility. Applied researchers may make a different accuracy–reproducibility tradeoff and select larger closed-weight models (Palmer, Smith, and Spirling 2024).

These models have varying context lengths: Llama-8B and Mistral-NeMo-12B handle 128K tokens, Mistral-7B handles 32K, and OLMo-7B a maximum input of 4096 tokens. We note that effective context length for information retrieval may be shorter (Hsieh *et al.* 2024).

4.3. LLMs for Multi-Class Classification

We use LLMs for multi-class classification by providing the entire codebook and document as input, then selecting the first generated token sequence matching a valid label as the predicted label $\hat{Z}$. Some research has attempted to constrain the output vocabulary or bias the generated tokens towards the valid set of labels (Ziems *et al.* 2024). However, other work has found using the generated text outputs is more accurate (Wang *et al.* 2024) and our choice of LLMs almost always outputted valid labels (see Test I in Stage 1), so we did not modify the LLM generation process for simplicity.

As an alternative to inputting the entire codebook with all class labels and descriptions, one could use LLMs in a *one-versus-rest* approach. In this approach, the LLM takes as input a single class description and makes a binary prediction, that the document is that label or not, e.g., Burnham *et al.* (2024). However, in preliminary experiments, we found no accuracy improvements in a one-versus-rest set-up. We hypothesize that for the datasets we gathered, the full codebook and all labels together are required to delineate between classes (see SI F in the Supplementary Material). For example, an LLM used in a one-versus-rest set-up might incorrectly predict that a drone strike meets the BFRS codebook's definition of an Assassination, unless it has access to the full codebook with the (correct) Drone Assassination category. Furthermore, one-versus-rest significantly increases the computational time.

We leave to future work a full engineering effort comparing these implementation details. Furthermore, with the rapid advancement of LLM development, we believe the quantitative results we present about particular LLMs will quickly be surpassed. However, we believe our evaluation framework, behavioral test templates, and curated codebook datasets will stand the test of time. We now describe in more detail the five stages of our empirical evaluation framework and provide an empirical demonstration of this framework with our codebook datasets and selected LLMs.

5. Stage 0: Codebook Preparation

In preparation for our main stages, researchers prepare the codebook by writing natural-language operationalizations of the variables and formatting the codebook so that is readable by both machines and humans. While LLMs offer exciting possibilities for scaling up measurement, this stage cannot be bypassed or automated. Researchers should also establish a desired level of classification accuracy for their downstream task and use this target to guide their progress through the remaining steps. In our case, we hoped to exceed F1 scores of 0.7.

5.1. Codebook Operationalization

Codebook-LLM measurement is fundamentally different from typical NLP classification because the same background concept or label—e.g., a "protest"—could have several different possible natural-language written operationalizations in the codebook. The goal is to have an LLM correctly attend to the specific operationalization, otherwise different political variables may be incorrectly conflated or correlated in downstream analysis.

5.2. Machine-Readable Semi-Structured Codebook Format

We propose a new generalizable and consistent format for codebooks that is both human and machine-readable. Previous instruction-tuning research suggests that explicitly providing a *definition*, *positive example*, and *negative example* can improve zero-shot performance on a diverse set of NLP tasks (Wang *et al.* 2022). Separating parts of each codebook definition into components also allows us to experimentally isolate and ablate components to evaluate changes in performance (see Section 8.3).

An excerpt of the restructured BFRS codebook is shown in Figure 1 and example output in Figure 2. We manually restructure each of the codebooks into this format. Future researchers could write codebooks from the outset to be both human- and machine-readable.

Prompt excerpt from the BFRS Codebook

Instructions:

You're an expert political scientist categorizing news stories from Pakistan into categories. Carefully read the definitions below, read the story, and write the Label that best matches the story. Use only the provided labels.

Classes:

Label: RIOT

Definition: A riot is a violent clash between two or more sizeable groups or when a single informally assembled crowd becomes violent.

Clarification: Neither group can be a state force. However, if there are two or more clashes around a single event in which the police are involved, then it would be a classified as a riot.

Negative Clarification: For example, the students of the Lal Masjid versus the police would not be considered a riot, it would be a Violent Political Demonstration.

Positive Example: "Five workers of the Muslim Students Federation, the student wing of the ruling Muslim conference, sustained bullet wounds in a clash with the activists of pro-independence National Students Federation."

Negative Example: "An angry mob thrashed police personnel, accusing them of manhandling people and using obscene language when they (the police) raided a gambling den." This incident involves police, so it is coded as a "violent political demonstration."

Label: VIOLENT_POLITICAL_DEMONSTRATION

Definition: A violent political demonstration is characterized as mobilization of crowds in response to a political event that at some point becomes violent.

Clarification: Violence may be initiated by the crowd or by the police upon the crowd.

Negative Clarification: A violent clash between two non-state groups is a riot, not a violent political demonstration.

Positive Example: "Hundreds of school children and armed supporters of Lashkar-i-Islam stormed the Bara bazaar and set the main tehsil, Qambaraband, Al Haaj Market, Sheikhan and Qasaban checkposts of the Khasadar force on fire. The Lashkar-i-Islam had warned of protests if their demands were not met."

Negative Example: "A man killed and his brother injured when they were attacked by their rivals over a land dispute." The motivation here is not a political event, thus this event is a riot, not a violent political demonstration.

...plus 10 additional classes [omitted for space]

Document:

Shia schoolboys held a demonstration demanding release of a student detained by Rangers in connection with an earlier disturbance. The students were about to disperse after negotiations with the police when they were encountered by Rangers at a barricade which led to scuffles and firing.

Output reminder: Write the name of the Label that fits best, with no other text. For example, "Label: ASSASSINATION", "Label: VIOLENT_POLITICAL_DEMONSTRATION", etc.

Figure 1. BFRS codebook as part of an LLM prompt. From the BFRS codebook, this is a (truncated to 2/12 labels) example of how the codebook is formatted as the LLM input.

LLM generated output

VIOLENT_POLITICAL_DEMONSTRATION
Explanation: The news story describes a demonstration by Shia schoolboys demanding the release of a detained student, which turned violent when they were encountered by Rangers at a barricade leading to scuffles and firing. This fits the definition of a violent political demonstration as it involves mobilization of crowds in response to a political event that at some point becomes violent.
Note: The use of the term "scuffles and firing" indicates that violence was initiated by both the crowd and the Rangers, making it a violent political demonstration and not a riot, as the crowd was not clashing with another group.
Therefore

Figure 2. Zero-shot output generated by Mistral-7B given the full BFRS prompt, an excerpt of which appears in Figure 1. We deterministically match the first part of the beginning of the output with the set of valid labels to determine the predicted label.

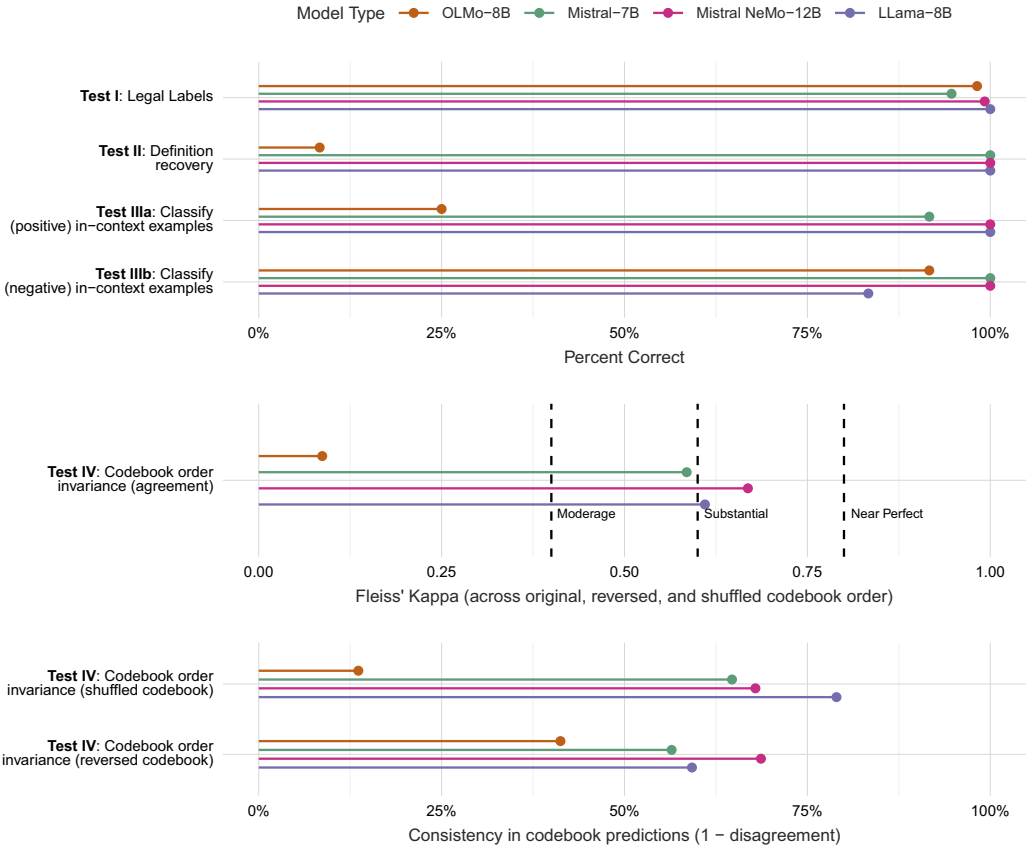

Figure 3. Label-free behavioral test results on the BFRS dataset. For all tests, higher numbers are better (see Table 2 for details of each test). For Test IV, dashed lines are the Fleiss Kappa heuristics from Landis (1977).

Our format consists of the following codebook components (see Figure 1):

1. *Label*: The exact label that the LLM should return if predicting the class.
2. *Label Definition.* We provide a succinct definition of the class, which is generally a single sentence.
3. *Clarification & Negative Clarification.* Most classes require further details to define them. This includes both additional inclusion criteria, as well as exclusion criteria for the class. For example, the BFRS category for RIOT does not apply if one of the groups involved is a police force.
4. *Positive & Negative Examples.* Optionally, a codebook can provide examples of documents that both fit and do not fit, with explations. "Few shot" or "in-context learning", when an LLM is provided example input–output pairs in the prompt, has been shown to improve LLM performance on new tasks (Brown *et al.* 2020).

In the LLM prompt, we also include an overall codebook-specific instruction describing the task and an "Output reminder". Depending on the measurement task, researchers can omit components or add others, e.g., adding a structured output (JSON) requirement.

6. Stage 1: Label-Free Behavioral Testing

At the outset of a project, applied researchers can benefit from cheap (i.e., requiring no ground truth labels) and interpretable tests to assure themselves that an LLM can perform basic tasks with the codebook and choose between LLMs early on in their project's lifecycle. We propose four label-free "behavioral tests" (Ribeiro *et al.* 2020) to measure an LLM's ability to recall basic parts of the codebook, to comply with codebook instructions, and to "comprehend" the definitions in the codebook. Table 2 provides a summary of the tests.

Test I provides a simple test of the LLM's ability to follow prompt instructions to only return the valid labels defined in the codebook.[1] Specifically, we check if any of the legal labels appear in the output text. *Tests II and III* measure the basic ability of the LLM to "memorize" parts of the prompt/codebook. For *Test II*, we provide the LLM with a prompt that contains the structured codebook, along with a (verbatim) class definition and an instruction to provide the label for the class definition. Passing the test requires the basic ability to match the provided definition with the (identical) definition in the codebook. For *Test III*, we provide verbatim positive and negative examples from the codebook and ask for their labels. None of these tasks require "reasoning", just simple pattern matching. Failure on these tasks may indicate problems with the LLM's ability to recall components of the codebook, which could have serious ramifications for codebook-LLM measurement as a whole.

If the LLM is "comprehending" the codebook, then its predicted labels should not change if the order that the categories are presented in the codebook changes (*Test IV*). We evaluate the LLM sensitivity to codebook order in two ways. First, we calculate the percentage of predicted labels that remain the same between the original codebook and predictions when the category order is (1) reversed or (2) randomly shuffled. Second, we calculate the inter-"coder" agreement, measured using Fleiss's kappa, in predictions across the original, reversed, and shuffled codebooks.

6.1. Results

We evaluate the four LLMs described in Section 4.2 on these four codebook-specific behavioral label-free tests. Because only the BFRS codebook had positive and negative examples, we conduct these tests only the BFRS dataset (see Figure 3 for results). Mistral-7B, Mistral-NeMo-12B, and Llama-8B perform well on Tests I through III, checking for legal outputs and verbatim definition and example recovery. We find that all models are sensitive to the order that categories are described in the codebook (IV), which may indicate problems with the LLM's attention for our long prompt contexts (see Liu *et al.* 2024, Zhao *et al.* 2021.

[1] Code for running these tests is available in the replication materials in `behavioral_tests.py`.

Table 2. Proposed behavioral tests for codebooks.

	Test	Ramifications of failed test
	Label-free behavioral tests	
I	*Legal labels*	
	Does the LLM only return labels defined in the codebook?	Inability to follow basic instructions or to recall the set of legal labels provided in the prompt.
II	*Definition recovery*	
	Can the LLM correctly label a verbatim codebook definition?	Inability to "memorize" or retrieve portions of the codebook.
III	*Classify in-context examples*	
	Can the LLM correctly label verbatim examples provided in the codebook?	Inability to "memorize" or retrieve portions of the codebook.
IV	*Codebook order invariance*	
	Are LLM predictions unaffected by codebook category order?	Inconsistent attention across the length of the prompt or ordering effects. Predicted labels depend on the order of classes in the prompt.
	Labels-required behavioral tests	
V	*Exclusion criteria consistency*	
	Across the four combinations of (modified, original) × (document, codebook), are the LLM predictions consistently correct?	Inability to follow instructions and ignore irrelevant distractions. May reveal problems attending to specific inclusion or exclusion criteria.
VI	*Generic label accuracy*	
	Can the LLM classify examples when given non-informative labels?	Over-reliance on the label as opposed to the definitions. May indicate that the predicted label reflects the background concept instead of the operationalized concept.
VII	*Swapped label accuracy*	
	Can the LLM classify examples according to the codebook's definitions when (informative) labels are randomly swapped?	Over-reliance on the names of the labels (vs. definitions).

Note: See Section 6 for "label-free" tests and Section 8 for "labels-required" tests.

Just as an applied researcher using our framework may do after this stage, we omit OLMo-7B from our later evaluation stages since its performance was very poor compared to the other models. Mistral-NeMo-12B performs well on some tests, but the improvement is not great enough to justify the larger model size and increased computational costs. Thus, we only compare Mistral-7B and Llama-8B in the next stages.

7. Stage 2: Zero-Shot Evaluation With Labels

After identifying promising models through label-free behavioral tests, applied researchers can hand-label a subset of their data and evaluate the zero-shot performance of LLMs on labeled data. This is not a new evaluation approach—see (Atreja *et al.* 2025; Ziems *et al.* 2024, *inter alia*)—but, in this work, we evaluate LLMs on our curated codebook datasets which we believe provide a more realistic picture of codebook-LLM measurement.

Table 3. Zero-shot weighted F1 scores for each LLM on each development dataset.

Dataset	Codebook format	Llama-8B	Mistral-7B
BFRS	Ours	0.57 [0.55-0.58]	0.53 [0.52-0.55]
	Original	0.55 [0.53-0.56]	0.44 [0.42-0.45]
CCC	Ours	0.61 [0.58-0.64]	0.65 [0.62-0.68]
	Original	0.48 [0.45-0.52]	0.51 [0.48-0.54]
Manifestos	Ours	0.19 [0.17-0.21]	0.15 [0.13-0.17]
	Original	0.21 [0.19-0.23]	0.14 [0.12-0.16]

Note: We compare our semi-structured codebook format ("ours") and the original authors' format (the latter only changed by prepending "Label:" when appropriate). Square brackets indicate 95% confidence intervals via 500 bootstrap resamples of the (predicted, true) pairs.

Table 4. Codebook ablation results for zero-shot predictions from Mistral-7B on the BFRS development dataset.

Codebook component						
Class definition	Output reminder	Positive example	Negative example	Clarification	Negative clarification	Dev F1
						0.53 *
	✗					0.42
		✗	✗			0.46
		✗	✗	✗	✗	0.43
	✗	✗	✗	✗	✗	0.20
✗	✗	✗	✗	✗	✗	0.29

Note: We report the weighted F1 scores, and ✗ indicates the component of the codebook that we ablated. *The unablated zero-shot result in the first row is the same as Table 3.

7.1. Zero-Shot Results

Table 3 shows the zero-shot weighted F1 scores[2] on the development set of each of our three datasets for Llama-8B and Mistral-7B. We compare our new semi-structured codebook format with the original codebooks as written by the original authors, only changing them by prepending "`Label:`" to each label in the codebook to indicate what the LLM should predict in its output.

In Table 3, the development-set zero-shot results range from very poor with 0.21 (weighted) F1 on Manifestos, to marginal with 0.65 and 0.57 F1 on CCC and BFRS, respectively. There is no clear "winner" between Llama-8B and Mistral-7B. This weak performance suggests that using either LLM on complex codebook tasks—with either the original or re-written codebook—is unlikely to be useful to applied analysts without fundamental improvements in the base LLMs to read, comprehend, and comply with codebook instructions or further updating the LLM weights on supervised examples (i.e., instruction-tuning in Stage 4). There is significant variation in the per-class F1 (see SI Tables 1, 2, and 3). We examine the robustness and further try to understand these zero-shot results in Stage 3.

[2]Specifically, we implement this via scikit-learn's function `f1_score(y_true, y_pred, average='weighted')`. This takes the average of per-class F1 scores, weighted by sample size, to account for imbalance.

8. Stage 3: Zero-Shot Error Analysis

Stage 3 of our framework expands zero-shot evaluation of LLMs beyond aggregate performance metrics in order to better understand the LLMs' errors and robustness. We use three complementary approaches: (1) we propose three additional *behavioral tests* that require labeled examples; (2) we systematically *ablate* different components of the codebook to understand how these different components affect LLM performance; and (3) we conduct detailed *manual analysis* of the output and explanations. We encourage applied researchers to use all of these approaches on their own codebook-LLM measurement projects—regardless of the aggregate zero-shot performance—to understand the errors the specific LLM they use is prone to make.

8.1. Labels-Required Behavioral Tests

We build from the label-free behavioral tests in Stage 1, to conduct additional behavioral tests that require ground-truth labels. We describe and implement three additional tests (Test V, VI, and VII) (see Table 2 and Figure 4).

Test V–Exclusion criteria consistency borrows a technique proposed by Karpinska *et al.* (2024) and examines LLMs' handling of exclusion rules. We modify a codebook to add a simple rule that invalidates a label when a specific word appears (e.g., "This category does not apply if the document discusses *elephants*"). We then create four test conditions by varying whether:

1. The document contains the trigger phrase (e.g., "We support *elephants*").
2. The codebook contains the exclusion rule.

A model that follows codebook instructions should only reject the label when both the trigger word is present *and* the exclusion rule exists in the codebook. To achieve perfect performance requires the model to attend to and apply the exclusion rule when present and ignore irrelevant words when there is no exclusion rule.

Tests VI and VII provide a direct measure of how much the LLM is relying on the label itself versus following the instructions in the codebook. In *Test VI–Generic Label Accuracy,* we replace the original labels with a non-informative labels (e.g., LABEL_1). In *VII–Swapped Label Accuracy,* we permute the original labels in the codebook so each label is paired with a different definition. An LLM should ideally follow the definitions despite the distraction of the swapped labels.

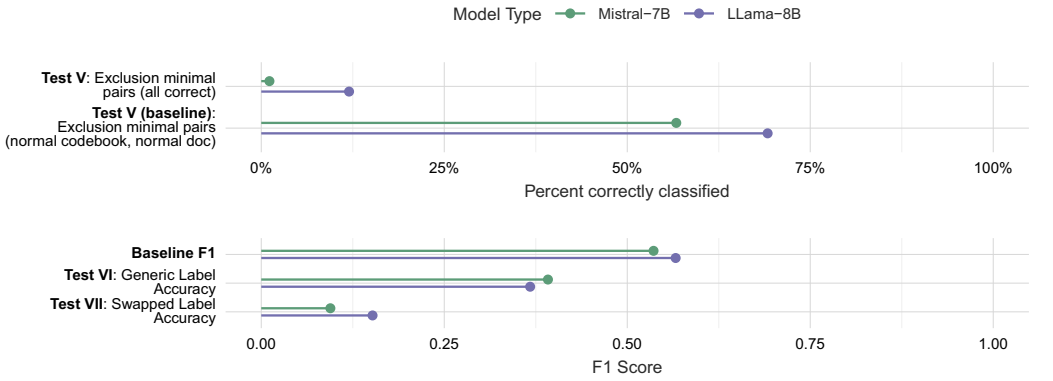

Figure 4. Labels-required behavioral tests of the two LLMs on the BFRS development dataset (see Table 2 for details of each test).

8.2. Results

Figure 4 shows the performance of Mistral-7B and Llama-8B on the labels-required behavioral tests, again using the BFRS development dataset and codebook. The models perform poorly on Test V, providing evidence that LLMs are not following specific codebook operationalizations. Finally, tests VI and VII show that the LLMs are indeed relying on the semantic content of the labels themselves, performing worse when the labels are replaced with generic terms or swapped. We caution applied researchers that if their constructs in their codebooks deviate too far from the background concepts an LLM has learned during pre-training, the zero-shot results may have systematic errors that could affect overall substantive results.

8.3. Zero-Shot Ablations

Next, to understand how different components of the codebook affect LLM performance, we conduct an ablation experiment on the BFRS dataset by systematically removing (ablating) different codebook elements (e.g., dropping "Negative Clarification" or "Positive Example") and measuring the impact on Mistral-7B's zero-shot F1 scores (see Table 4). We used the BFRS codebook because its original codebook was the most similar to our proposed new format (i.e., it already included positive and negative examples). We explore a limited set of the 2^6 possible options, focusing on the ones we believe are most relevant to applied researchers.

8.4. Results

Table 4 shows the results of five ablation experiments. As expected, systematically ablating components decreases performance, indicating most components are necessary to achieve even the original F1 score of 0.53. Ablating the output reminder results in decreased performance (from 0.53 to 0.42 F1). Notably, comparing the fifth to sixth rows in Table 4 results in surprisingly *better* performance when dropping the class definition, providing only the label itself. In combination with our other error analysis results in this stage, this finding may suggest that the LLM is not fully adhering to the label definition in the codebook, which could be consequential for any codebook-LLM measurement that compared the same label across two operationalizations (e.g., the two "protest" operationalizations we mentioned in the introduction). In this codebook–LLM combination, including all codebook components improves performance, but we are cautious about drawing universal conclusions about optimal codebook components and encourage applied researchers to perform ablation exeriments for their own applications.

8.5. Manual Analysis of Zero-Shot Generative Outputs

We recommend applied researchers manually analyze LLM outputs. Although the explanations generated by an LLM are not necessarily always logically consistent or "faithful" to their internal representations and predictions (Jacovi and Goldberg 2020; Lyu, Apidianaki, and Callison-Burch 2024), inspecting the output can illuminate some of the LLM's failure modes. As a demonstration, we conducted a careful manual analysis on a sample from each dataset of the zero-shot outputs from Mistral-7B. Then, we (the two authors) manually analyzed the model inputs, model outputs, and gold-standard labels. After an initial pilot round, we manually labeled LLM's outputs into one of six classes (see SI K for details in the Supplementary Material).

8.6. Results

Table 5 shows the aggregate results of the manual analysis. Mistral-7B compliance with prompt instructions related to allowable labels was excellent for BFRS (0% non-compliance) and CCC (2% non-compliance). However, for Manifestos, 45% of the sampled LLM outputs were non-compliant. These

Table 5. Manual error analysis results on a sample of the zero-shot generative outputs for Mistral-7B given each development dataset.

	BFRS	CCC	Manifestos
Sample: No. of examples	24	50	62
Sample: No. of unique classes	12	5	50
A. LLM correct	0.38	0.48	0.11
B. Incorrect gold standard	0.04	0.10	0.08
C. Document error (scraping/lack context)	0.04	0.02	0.03
D. LLM non-compliance	0.00	0.02	0.45
E. LLM semantics/reasoning mistake	0.50	0.26	0.29
F. Other	0.04	0.10	0.02

Note: For each dataset, we report the number of examples and the number of unique classes in that sample as well as the proportion of outputs in categories A-F as judged by the authors (see SI K for definitions of each category).

included errors such as outputting multiple labels or more egregious errors such as hallucinating labels (see Table SI 4).

We find evidence suggesting that Mistral-7B is relying on heuristics and shortcuts and not actually "reading" the codebooks. In one instance, a Manifesto example supporting increased education funding gets the predicted label "WELFARE_POSITIVE (or EDUCATION_POSITIVE)" (see SI Table 4). However, an explicit constraint in the Manifesto's codebook for the WELFARE_POSITIVE label is "This category excludes education."

We also find evidence that Mistral is using lexical overlap heuristics (Levy, Ravfogel, and Goldberg 2023)—selecting label words that appear in the text, even if the label is incorrect. For example, in CCC-1 in SI Table 4, the word "rally" occurs in the first sentence of the text and the LLM predicts the RALLY label even though the text clearly aligns with the codebook definition of a DEMONSTRATION.

These errors reveal limitations in how Mistral-7B interprets and applies the precise definitional boundaries that make codebooks valuable as measurement tools, and motivates Stage 4 to update the weights of an LLM given additional supervised examples.

9. Stage 4: Supervised Fine-tuning

If Stages 1–3 show zero-shot performance is inadequate, researchers may need to abandon the zero-shot approach. If this is the case, they can return to (familiar) supervised classification techniques: training a "classic" supervised machine learning model from scratch or fine-tuning a BERT-based model to predict the label for each document (Grimmer, Roberts, and Stewart 2022). Rather than repeat the extensive literature on supervised text classification, we describe and provide an empirical demonstration for a promising new technique–*instruction-tuning* in which an LLM's weights are further updated on supervised input/output pairs via a next token prediction objective function (Longpre *et al.* 2023; Sanh *et al.* 2022; Wang *et al.* 2023; Wang *et al.* 2022; Wei *et al.* 2022). While zero-shot prompting provides instructions to an off-the-shelf model at inference time, instruction-tuning allows us to potentially improve the model's ability to follow instructions, using supervised learning. This approach differs from traditional supervised learning—rather than training a supervised model on (document, label) pairs to return the correct label, we train it on (instruction, document, label) pairs, and obtain a generated label (e.g., y = "protest"). (For a glossary of terms used in this section, see SI A in the Supplementary Material.)

Directly updating the weights in LLMs given supervised examples is very expensive in terms of training time and memory because most LLMs consist of billions of weights. However, we describe how to use parameter-efficient techniques of *quantization*—loading model weights with less precision, i.e., using fewer bits—and *low-rank adaptation (LoRA)* in which only a fraction of the weights are updated.

In practice, instruction-tuning involves providing the LLM with training examples from the original datasets, consisting of:

1. the codebook (C);
2. a document to classify (X_i);
3. the correct natural-language label for that document e.g., "protest" (y_i).

For our instruction-tuning set-up, a training dataset—BFRS, CCC, or Manifestos—consists of D tuples $\{(m_i, y_i)\}_{i=1,}^{D}$ where m_i is the entire "prompt" (consisting of the concatenation of C and X_i as well as any other instructions; see Figure 1).

Following previous work (Wang *et al.* 2023), we do not compute the language modeling loss on the prompt, only on the output (See SI G). For each example $i = 1, \ldots, D$, let N_{m_i} be the total number of tokens for the example's prompt m_i and N_{y_i} be the total number of tokens for the corresponding output y_i. Then, the negative log-likelihood loss function (masking inputs) is:

$$L_\theta \equiv -\sum_{i=1}^{D} \sum_{j \in [N_{m_i}, N_{y_i}]} \log p_\theta(t_{i,j} | t_{i,<j}) \times \begin{cases} 0 \text{ if } j \in N_{m_i} \\ 1 \text{ otherwise,} \end{cases} \tag{1}$$

where $t_{i,j}$ is the gold-standard jth token in either the input m_i or output y_i. The model weights θ are then updated to minimize this loss function.

9.1. QLoRA Training: Quantization + LoRA

Fully updating all the weights in an LLM during instruction-tuning is costly in computing time and memory requirements. To address this, we use "quantized low-rank adaptation" (QLoRA) (Dettmers *et al.* 2023; Hu *et al.* 2022), a technique that makes fine-tuning LLMs more efficient in two ways. First, it "quantizes" each weight. Quantization is a tradeoff between a model's memory footprint and accuracy; quantization reduces the numerical precision of the model's weights (e.g., from 32 bits to 4 bits) to use less memory but this decrease in information can also decrease the model's performance. Second, instead of updating the full matrix of weights, it decomposes the weight matrix into two low-rank weight matrices to update.[3] This approach closely matches the set-up described in (Jindal, Rajpoot, and Parikh 2024), which won the 2023 NeurIPS challenge for efficient LLM instruction-tuning.

We use a fixed rank of 16 in the results shown in Table 6. LoRA models have a broader set of hyperparameters, which we keep fixed; future work could look to domain-specific hyperparameter tuning (see Section G in the Supplementary Material for details).

9.2. Instruction-Tuning Results

Table 6 reports the weighted F1 scores of our instruction-tuned models. For each model, we conduct QLoRA fine-tuning on the entire training set and evaluate on the entire dev set. We find consistent improvement in all instruction-tuned models over their zero-shot results. For example, on BFRS, Mistral-7B improves its performance over its zero-shot results by 0.29 F1 (from 0.53 zero-shot to 0.82 instruction-tuned) which is a 55% relative improvement. Performance gains for CCC are lower across both models and only slightly better than the baseline of the majority class. We hypothesize this could be due to CCC documents being much longer and thus much more difficult to classify. We omit the result for Llama on Manifestos—even with a batch size of one, the training process exceeded our GPU's memory.

[3]That is, rather than updating the entire weight matrix $W \in \mathcal{R}^{d \times d}$, LoRA decomposes W into $d \times r$, $r \times d$ matrices, with significantly fewer weights than the full weight matrix (with r=16, ~ 0.5%.)

Table 6. Results of the LLMs after instruction-tuning on each training dataset.

Dataset	N	LLM	Dev F1	Δ Zero-shot	Train time
BFRS	20,978	Baseline: Majority class	0.16	–	–
		Llama-8B	0.81 [0.80–0.82]	+0.24	6h 01m 11s
		Mistral-7B	0.82 [0.81–0.83]	+0.29	6h 51m 59s
CCC	4,710	Baseline: Majority class	0.51	–	–
		Llama-8B	0.68 [0.64–0.70]	+0.07	1h 29m 52s
		Mistral-7B	0.72 [0.69–0.75]	+0.07	1h 40m 38s
Manifestos	8,081	Baseline: Majority class	0.03	–	–
		Llama-8B	–*	–	–
		Mistral-7B	0.38 [0.35–0.40]	+0.23	17h 53m 47s

Note: N is the number training examples (same as Table 1). We report a baseline of predicting only the class that was seen the most during training (majority class), the weighted F1 score on the development set (Dev F1), the change from the corresponding model's zero-shot results in Table 3 (Δ zero-shot), and the total training wall time on our single NVIDIA RTX 4090 GPU. Square brackets indicate 95% confidence intervals via 500 bootstrap resamples of the (predicted, true) pairs. *We do not report Llama result on the Manifestos due to out-of-memory error with batch size=1.

The fine-tuning process was quite costly in terms of total training time. However, the loss plateaued after about 10% of the total training examples so future work might be able to speed this up by training on fewer examples or better "early stopping." Overall, these results indicate that instruction-tuning is a viable approach for improving a model's ability to predict classes given challenging codebooks and poor zero-shot results, albeit with increased computational and annotation costs.

10. Future Work

We see several avenues for future research. Our set of three datasets and codebooks provide a challenging test bed (especially Manifestos) for a real-world codebook-LLM measurement. Future engineering work will improve on our zero-shot and instruction-tuning results presented in this article by using newer models, employing techniques such as chain-of-thought reasoning (Wang *et al.* 2022) or self-improving prompts (Khattab *et al.* 2024), and experimenting with different instruction-tuning hyperparameters. We emphasize that our article's core contributions lie in formalizing codebook-LLM measurement, proposing a set of behavioral tests, providing benchmark datasets, and establishing baseline performance results. We fully expect future work to achieve improved performance.

LLMs could also be used in-the-loop for codebook development. Codebooks are often developed iteratively during annotation as edge cases arise or annotators raise questions. Updating the codebook during annotation requires retraining annotators and potentially re-annotating existing documents, which can slow the annotation process. LLMs have the potential to reduce the number of changes a codebook requires after annotation begins by identifying gaps in the codebook.

Finally, future work could explore different tasks, such as information extraction or multi-label classification, and performance on non-English language text.

11. Conclusion

LLMs offer exciting possibilities for analyzing and classifying text with less human effort. However, using LLMs for codebook-based measurement is different than many other applications of LLMs: It requires the LLM to follow the lengthy and detailed operationalization of concepts in a codebook and to faithfully apply the provided coding criteria rather than solely relying on what it learned from pre-training. We believe the three codebook datasets we gathered and curated in this article represent the

real-world complexity of the codebook-LLM measurement task, and there likely exist many more that could be added to this collection.

For applied researchers looking to use LLMs now, we recommend following our five-stage empirical framework to systematically evaluate whether and how to employ LLMs for their specific measurement tasks. As LLMs improve, we expect measurement accuracy to improve as well. Rather than identifying a "best" LLM, we provide guidance and tools for researchers to make informed decisions about codebook preparation, model selection, and the choice between zero-shot use and supervised fine-tuning. We expect our framework, particularly our behavioral tests and error analysis techniques, will remain valuable tools for assessing measurement validity as the technology evolves.

Acknowledgements. We thank Dallas Card, Luke Sanford, Maria Antoniak, the anonymous reviewers, and attendees at PolMeth 2024 for helpful comments on the manuscript.

Data Availability Statement. Replication code and data for this article is available as Halterman and Keith (2025) and available via Dataverse at https://doi.org/10.7910/DVN/NUWHQP.

Supplementary Material. For supplementary material accompanying this paper, please visit https://doi.org/10.1017/pan.2025.10017.

References

Adcock, R., and D. Collier. 2001. "Measurement Validity: A Shared Standard for Qualitative and Quantitative Research." *American Political Science Review* 95 (3): 529–546.

Atreja, S., J. Ashkinaze, L. Li, J. Mendelsohn, and L. Hemphill. 2025. "What's in a Prompt?: A Large-Scale Experiment to Assess the Impact of Prompt Design on the Compliance and Accuracy of LLM-Generated Text Annotations." *Proceedings of the International AAAI Conference on Web and Social Media* 19: 122–145.

Brown, T., et al. 2020 "Language Models are Few-Shot Learners." In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.) *NeurIPS*, virtual 33: 331877–1901.

Bueno de Mesquita, E., C. C. Fair, J. Jordan, R. B. Rais, and J. N. Shapiro. 2015. "Measuring Political Violence in Pakistan: Insights from the BFRS Dataset." *Conflict Management and Peace Science* 32 (5): 536–558.

Burnham, M. 2025. "Stance Detection: A Practical Guide to Classifying Political Beliefs in Text." *Political Science Research and Methods* 13 (3): 611–628.

Burnham, M., K. Kahn, R. Y. Wang, and R. X. Peng. 2024. "Political Debate: Efficient Zero-Shot and Few-Shot Classifiers for Political Text." Preprint, arXiv:2409.02078.

Chen, J. M., R. Bhattacharya, and K. A. Keith. 2024. "Proximal Causal Inference with Text Data." In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.) *NeurIPS*, Vancouver, Canada 37: 135983–136017.

Crowd Counting Consortium. 2024. crowdcounting.org.

Dettmers, T., A. Pagnoni, A. Holtzman, and L. Zettlemoyer. 2023. "QLoRA: Efficient Finetuning of Quantized LLMs." In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.) *NeurIPS*, New Orleans 36: 10088–10115.

Egami, N., M. Hinck, B. Stewart, and H. Wei. 2023. "Using Imperfect Surrogates for Downstream Inference: Design-Based Supervised Learning for Social Science Applications of Large Language Models." In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.) *NeurIPS*, New Orleans 36: 10088–10115.

Gerner, D. J., P. A. Schrodt, O. Yilmaz, and R. Abu-Jabr. 2002. "Conflict and Mediation Event Observations (CAMEO): A New Event Data Framework for the Analysis of Foreign Policy Interactions." *International Studies Association conference paper*. New Orleans.

Gilardi, F., M. Alizadeh, and M. Kubli. 2023. "ChatGPT Outperforms Crowd Workers for Text-Annotation Tasks." *PNAS* 120 (30): e2305016120.

Grimmer, J., M. E. Roberts, and B. M. Stewart. 2022. *Text as Data: A New Framework for Machine Learning and the Social Sciences*. Princeton, NJ: Princeton University Press.

Halterman, A. 2025. "Synthetically Generated Text for Supervised Text Analysis." *Political Analysis* 33: 181–194.

Halterman, A., K. Keith, S. Sarwar, and B. O'Connor. 2021. Corpus-Level Evaluation for Event QA: The IndiaPoliceEvents Corpus Covering the 2002 Gujarat Violence." In C. Zong, F. Xia, W. Li, and R. Navigli (Eds.) *Findings of the ACL-IJCNLP*, virtual 4240–4253.

Halterman, A., and K. A. Keith. 2025. "Replication Data for "Codebook LLMs: Evaluating LLMs as Measurement Tools for Political Science Concepts"." Harvard Dataverse. https://doi.org/10.7910/DVN/NUWHQP

Heseltine, M., and B. Clemm von Hohenberg. 2024. "Large Language Models as a Substitute for Human Experts in Annotating Political Text." *Research & Politics* 11(1).

Hsieh, C.-P., et al. 2024. "RULER: What's the real context size of your long-context language models?" *COLM*.

Hu, E. J., et al. 2022. "LoRA: Low-Rank Adaptation of Large Language Models." In *ICLR*, virtual.

Jacovi, A., and Y. Goldberg. 2020. "Towards Faithfully Interpretable NLP Systems: How Should we Define and Evaluate Faithfulness?" In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault (Eds.) *ACL*, virtual. 4198–4205.
Jindal, A. K., P. K. Rajpoot, and A. Parikh. 2024. "Birbal: An Efficient 7B Instruct-Model Fine-Tuned With Curated Datasets." Preprint, arXiv:2403.02247.
Karpinska, M., K. Thai, K. Lo, T. Goyal, and M. Iyyer. 2024. "One Thousand and One Pairs: A "Novel" Challenge for Long-Context Language Models." In Y. Al-Onaizan, M. Bansal, and Y. Chen (Eds.) *EMNLP*, Miami, Florida, USA 17048–17085.
Khattab, O., et al. 2024. "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines." In *ICLR*, Vienna.
Knox, D., C. Lucas, and W. K. T. Cho. 2022. "Testing Causal Theories with Learned Proxies." *Annual Review of Political Science* 25: 419–441.
Landis, J. 1977. "The Measurement of Observer Agreement for Categorical Data." *Biometrics* 33 (1): 159–74.
Lehmann, P., T. Matthieß, N. Merz, S. Regel, and A. Werner. 2017. *Manifesto Corpus. Version: 2017b*. Berlin: WZB Berlin Social Science Center.
Levy, M., S. Ravfogel, and Y. Goldberg. 2023. "Guiding LLM to Fool Itself: Automatically Manipulating Machine Reading Comprehension Shortcut Triggers." In H. Bouamor, J. Pino, and K. Bali (Eds.) *Findings of EMNLP*, Singapore. 8495–8505.
Liu, N. F., et al. 2024. "Lost in the Middle: How Language Models Use Long Contexts." *Transactions of the Association for Computational Linguistics* 12: 157–173.
Longpre, S., et al. 2023. "The Flan Collection: Designing Data and Methods for Effective Instruction Tuning." In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett (Eds.) *ICML*, Honolulu, Hawaii, USA 202: 22631–22648.
Lyu, Q., M. Apidianaki, and C. Callison-Burch. 2024. "Towards Faithful Model Explanation in NLP: A Survey." *Computational Linguistics* 50 (2): 657–723.
Mellon, J., J. Bailey, R. Scott, J. Breckwoldt, M. Miori, and P. Schmedeman. 2024. "Do AIs Know What the Most Important Issue is? Using Language Models to Code Open-Text Social Survey Responses at Scale." *Research & Politics* 11 (1).
Palmer, A., N. A. Smith, and A. Spirling. 2024. "Using Proprietary Language Models in Academic Research Requires Explicit Justification." *Nature Computational Science* 4: 2–3.
Pangakis, N., and S. Wolken. 2024. "Knowledge Distillation in Automated Annotation: Supervised Text Classification with LLM-Generated Training Labels." In D. Card, A. Field, D. Hovy, and K. Keith (Eds.) *NLP+CSS Workshop*, Mexico City, Mexico 113–131.
Peskoff, D., and B. M. Stewart. 2023. "Credible Without Credit: Domain Experts Assess Generative Language Models." In A. Rogers, J. Boyd-Graber, and N. Okazaki (Eds.) *ACL*, Toronto, Canada 427–438.
Ribeiro, M. T., T. Wu, C. Guestrin, and S. Singh. 2020. "Beyond Accuracy: Behavioral Testing of NLP Models with CheckList." In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault (Eds.) *ACL*, virtual 4902–4912.
Rytting, C. M., et al. 2023. "Towards Coding Social Science Datasets with Language Models." Preprint, arXiv:2306.02177.
Sanh, V., et al. 2022. "Multitask Prompted Training Enables Zero-Shot Task Generalization." In *ICLR*, virtual.
Thalken, R., Stiglitz, E., Mimno, D., & Wilkens, M. (2023). "Modeling Legal Reasoning: LM Annotation at the Edge of Human Agreement." In H. Bouamor, J. Pino, and K. Bali (Eds.) *EMNLP*, Singapore 9252–9265.
Wang, X., C. Hu, B. Ma, P. Röttger, and B. Plank. 2024. "Look at the Text: Instruction-Tuned Language Models are More Robust Multiple Choice Selectors Than You Think." In *COLM*, Philadephia, PA, USA.
Wang, Y., et al. 2023. "How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources." In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.) *NeurIPS*, New Orleans, LA, USA 74764–74786.
Wang, Y., et al. 2022. "Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks." In Y. Goldberg, Z. Kozareva, and Y. Zhang (Eds.) *EMNLP*, Abu Dhabi, United Arab Emirates 5085–5109.
Wei, J., et al. 2022. "Finetuned Language Models are Zero-Shot Learners." In *ICLR*, virtual.
Zhao, Z., E. Wallace, S. Feng, D. Klein, and S. Singh. 2021. "Calibrate Before Use: Improving Few-Shot Performance of Language Models." In M. Meila, T. Zhang (Eds.) *ICML*, virtual 139: 12697–12706.
Ziems, C., W. Held, O. Shaikh, J. Chen, Z. Zhang, and D. Yang. 2024. "Can Large Language Models Transform Computational Social Science?" *Computational Linguistics* 50 (1): 237–291.

Cite this article: Halterman, A. and Keith, K. A. (2025). Codebook LLMs: Evaluating LLMs as Measurement Tools for Political Science Concepts. *Political Analysis*, 1–17. https://doi.org/10.1017/pan.2025.10017