PNAS Nexus, 2024, pp. 1–6

doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

PAPER

Coding with the machines: machine-assisted coding of rare event data

Henry David Overos,[1] Roman Hlatky,[2] Ojashwi Pathak,[1] Harriet Goers,[1] Jordan Gouws-Dewar,[1] Katy Smith,[3] Keith Padraic Chew,[4] Jóhanna K. Birnir[1,*] and Amy Liu[3]

[1]Government and Politics, University of Maryland at College Park, College Park, MD, USA, [2]Political Science, University of North Texas, Denton, TX, USA, [3]Government, University of Texas at Austin, Austin, TX, USA, [4]School of Politics and Global Studies, Arizona State University, Tempe, AZ, USA and [*]To whom correspondence should be addressed: jkbirnir@umd.edu
FOR PUBLISHER ONLY Received on Date Month Year; accepted on Date Month Year

Abstract

While machine-coding of data has dramatically advanced in recent years, the literature raises significant concerns about validation of LLM classification showing, for example, that reliability varies greatly by prompt and temperature tuning, across subject areas and tasks – especially in "zero-shot" applications. This paper contributes to the discussion of validation in several different ways. To test the relative performance of supervised and semi-supervised algorithms when coding political data, we compare three models' performances to each other over multiple iterations for each model and to trained expert coding of data. We also examine changes in performance resulting from prompt engineering and pre-processing of source data. To ameliorate concerns regarding LLM's pre-training on test data, we assess performance by updating an existing dataset beyond what is publicly available. Overall, we find that only GPT-4 approaches trained expert coders when coding contexts familiar to human coders and is more consistent coding across contexts. We conclude by discussing some benefits and drawbacks of machine coding moving forward.

Key words: machine-coding, political event data, GPT, BERT, machine learning

Significance statement
Validation of machine-coded political data remains a challenge. We assess the performance of three models (a semi-supervised, dictionary-based Bayes classifier, BERT, and GPT) relative to human expert coders by coding data previously unavailable for the training of LLMs. GPT outperforms the other models and approximates trained human coders. Furthermore, our exercise highlights new issues in validation including variance in quality of human coding, the task dependent relevance of prompt engineering, and the effect of data type and pre-processing on machine performance metrics.

Machine-coding and analysis of textual data have advanced dramatically in the last three decades [14, 13, 21]. Researchers now use large language models (LLMs) – instead of human coding [13, p. 28] – to classify complex text whether it is across cases [6] or within a case [17]. Indeed, there are scholars who suggest that LLMs - such as GPT - now rival human crowd-sourced coding of data [4] and even expert coding [16]. At the same time, there are concerns that reliability varies greatly – not just by prompt and temperature tuning [12] – but also across source data [10] and tasks [22]. These concerns are especially pronounced for LLM classifications that rely on "zero-shot"[1] applications [7]. Consequently, researchers call

[1] As opposed to multi-shot and fine-tuning, which train models on a subset of the data.

for LLM classifications to be validated [10, 15]. The question remains: How do we validate machine-coded data?

This paper makes multiple contributions to the discussion of validation. First, to counteract concerns with contamination – i.e., testing on data that may have been used in the training of LLMs (e.g., GPT) – we assess performance by coding new data that updates an existing dataset beyond what is publicly available. Second, to validate the machine coding, we compare the performance of three models – to each other, iteratively to themselves, and to human-coded, new data. In doing so, we focus on the crucial topic of prompt engineering, with an eye to improving results in zero-shot applications and beyond [21, 9]. Third, while our human-coding follows the literature [4, 16, 10] in relying on expert (student) coders, we iteratively trained them to improve their coding output. This, we believe, provides a more realistic – albeit higher – bar for comparing between machine and human coding. Fourth, and related, in contrast to restricting coding to cases where experts have the requisite familiarity [4, 16], we explicitly compare expert-coding performance across familiar and unfamiliar cases to identify differences between human coders and LLMs. Finally, we address the under-explored topic of the effect of task [22] and source data [10] on LLM performance especially as they relate to prompt engineering.

We first apply a semi-supervised dictionary-based Bayes classifier model to code ethnic group protests across time and space. We use this model to identify rare or difficult-to-categorize topics such as protest – using a well-defined dataset on identity groups. The *All Minorities at Risk* (AMAR) project[2] documents ethnic minorities worldwide 1945-2006. Today, AMAR (and its predecessor) data are widely used in the study of ethnic politics – from elections to rebellion [1]. Using the AMAR framework, we code protest events across ethnic groups in different countries during years not previously included in the data. Next, we apply two LLMs developed for Natural Language Processing (NLP) to code the same data from the same sources. The first model is Bidirectional Encoder Representations from Transformers (BERT). This model uses a transformer mechanism to learn contextual relations between words in both directions of a text simultaneously. The second LLM is the Generative Pre-trained Transformer (GPT). Both BERT and GPT are pre-trained on large volumes of text data, designed to predict the next word in a text, and can be fine-tuned for downstream tasks such as classification.

We begin with a discussion of the data to be coded. Next, we introduce the central features of each model and the procedure for identifying relevant events mentioned in news articles.[3] We then (1) assess the effectiveness of each coding type for the same data using common metrics – e.g., precision, recall, F1 scores, and ROC graphs; and (2) compare the results to human-coding across cases. Overall, we find that transformer models – both an out-of-the-box GPT-4 model and a fine-tuned BERT model – outperform the semi-supervised dictionary-based classifier bag-of-words approach. Furthermore, GPT-4 performs consistently whereas the reliability of human coding varies substantially with their expertise on a topic. The model's performance suggests some important refinement to our thinking about prompt engineering. We conclude by identifying topics to consider when incorporating machine coding to the collection of political data.

The Setting

The AMAR data codes information for a representative sample of socially-relevant ethnic groups across the world between 1945-2006 [1]. The source material – from news reports to research articles, from books to websites – is all *publicly available*. This makes AMAR's data collection efforts uniquely transparent and verifiable. The principal drawback with such a strategy, however, is that human-coding of this scale is prohibitively expensive and slow – thus making updating and other maintenance untenable. One solution is to draw on advances in computational methods [5] – in conjunction with the use of publicly-available source material. A demonstrably reliable model of the data identification process can reduce the cost of data-coding, while increasing its speed.

Since multiple commercial services collect publicly-available electronic texts, we outsource the first step – creating a corpus of new material relevant to the unit of analysis – to BuzzSumo.[4] Here, the units of analysis are ethnic groups and for this article, we focus on **African-Americans** in the US and **Dalits** in India in 2021. This data is currently **not** coded in the publicly available AMAR data. To retrieve a corpus, we provide BuzzSumo with our ethnic dictionaries and search terms, limiting our results by language (to English) and the defined list of countries.[5] To ensure that the resulting corpus includes information about the concerned ethnic groups, we also give BuzzSumo a list of newspapers that we know are likely to publish articles about said groups.

To assess accuracy of machine coding of a variable – in this case, **protest** – we rely on commonly-used metrics: (1) *precision*, i.e., does the model identify only the relevant data points?; (2) *recall*, i.e., does the model find the relevant cases?; and (3) *F1*, i.e., what is the overall accuracy score based on precision and recall?[6]

The literature uniformly agrees that the best way to assess accuracy is to compare the predictions from the machine-coded data to the same observations that have also been human-coded [10, 15]. The simplified procedure is:

1. Have a research team annotate news articles.
2. Split the data into two sub-samples for training (80%) and testing (20%) the language models.
3. Task the language models to code the same data by learning from the training sample.
4. Assess the accuracy of machine-coded data by comparing human-coded and machine-coded results (on testing sample).

Human Coding

Expert coders are commonly students or research assistants [4, 16] with subject area expertise. We recruited multiple teams (N=7) of undergraduate research assistants (N=6 students per team on average). Students received class credit for participation. Unlike prior validation assessments, we further trained our experts with coding. In the first meeting, we

[2] AMAR is an independent continuation of Ted Gurr's *Minorities at Risk* data.

[3] See appendix for a detailed account of each method.

[4] Users need to consider selection issues with the curating of sources.

[5] See appendix for the search terms used to build the corpus.

[6] See appendix for metric description.

introduced the project, the AMAR codebook, and the coding process. We walked the students through several example articles (N=10). We then gave students the above-referenced random sample of news articles. We asked the students to identify whether an article provided sufficient evidence to be coded as about: (1) a specific ethnic group found in the AMAR data, and (2) protest by the subject group. These were independent decisions. We also asked students to substantiate their decisions by providing a quote from the article to support their coding.

For the first week, we assigned every student the same 25-article batch. In the following week, we reconvened for further training where we discussed "controversial" articles – i.e., those that produced split decisions among coders – and adjudicated based on group consensus. If we believed the students still struggled with the coding assignment (only one team struggled), we continued the training by giving every student another 25-article batch. Once the students grasped the assignment, we assigned more articles (N=50); we also dropped the number of coders per article to four students. Coding happened over multiple semesters. In all, the teams coded 1718 news articles for this analysis.[7]

Intercoder reliability varied. For example, the ICC among the six coders responsible for labeling news articles related to African Americans ranged from 0.93 to 0.97. For the Dalit articles (four coders), there was less reliability as to whether the article was about Dalits (ICC=0.48). Note, however, the ICC for Dalit protest indicated greater reliability (0.84).

We took several steps to remedy low ICC scores. First, we identified potentially problematic categories early on during consensus coding due to the number of emerging "split" decisions. In these cases, second, we returned to the codebook; we went over it with the students to ensure adequate understanding of a given category. Third, we group-coded several articles with the students to verify sufficient understanding. Finally, we allocated more time on split decisions during consensus-coding sessions.

Despite coding articles together as a team and the consensus coding of controversial articles, reliability for the Dalit category remained low. Moreover, – compared to the African-Amerian sample – the ICC scores for protest also suffered when the sample was drawn from articles about Dalits. The different ICC scores for the African-American and Dalit categories illustrate that while reliability is high when experts code a familiar category – e.g., African-Americans – it suffers when they code less familiar ones even when trained. Furthermore, while the largest potential problems may arise when coders categorize "unfamiliar" identity groups, reliability also decreases when categorizing "transferable" concepts (e.g., protest) in unfamiliar contexts.

Model 1: Newsmap

We use a naive Bayes classifier as our baseline algorithm for comparison to the human coding. The model, *Newsmap*, was originally designed to improve the geolocation of news articles over current gazetteers [18]. As a bag-of-words model partially trained on a researcher-provided set of "seed words", *Newsmap* requires fewer assumptions for generating predictions – thus making it especially useful when researchers are interested in a well-defined set of cases and variables. We derive the seed words to train the model from two sources: (1) directly from the AMAR codebook and (2) from existing news sources used to justify AMAR coding decisions. The *Newsmap* workflow is a trained machine application for coding. Table 1 summarizes the model's workflow.

Table 1. Overview of the *Newsmap* Model Process

1. Obtain corpus of relevant documents for analysis.
2. Create dictionary of seed words related to topic of interest.
3. Identify relevant documents containing seed words from dictionary.
4. Classify portion of documents (80% of data) using labeled documents from step 3 as training data.
5. Take model results from step 4 and introduce more unlabeled documents into total data pool.
6. Estimate probability that words in unlabeled data predict each class.
7. Re-estimate classifier from step 4 with results from steps 5 and 6.
8. Repeat steps 5-7 until model converges.
9. Introduce the remaining 20% of human coded data to *Newsmap* for classification.

Model 2: BERT

We compare *Newsmap's* performance against models with different architectures, namely *transformer* models. Although bag-of-word models are more transparent and computationally less expensive, they may have less predictive power for large or complex texts – thus making a good comparison for more context-focused transformer models. While bag-of-words models only consider the frequency of terms in a document without regard for context, transformer models weigh the importance of certain words in context. The mechanism for weighing the predictive power of words in context is called *self-attention*, and it allows for both increased speed and precision when completing complex classification tasks.

The first transformer model we evaluate is BERT, which comes pre-trained on a large corpus of documents for a general understanding of language in context [2]. Addressing concerns with zero-shot applications, researchers can then fine-tune the model to solve specific tasks, e.g., providing additional labeled texts for classification. We describe the fine-tuning process in the appendix. BERT is effective at several tasks including classification; there is also evidence that it outperforms many other similar models on NLP tasks. Table 2 outlines the modeling process when using BERT. We run BERT in Python via the *transformers* package.

Table 2. Overview of the BERT Model Process

1. Obtain corpus of relevant documents for analysis.
2. Load BERT model pre-trained on a general corpus of texts.
3. Fine-tune BERT model for specific coding project, using a sample (80%) of human-coded data from the corpus obtained in step 1.
4. Introduce remaining 20% of human coded data to BERT for classification.

[7] See supplemental materials for descriptive statistics on the final news articles data set.

Model 3: GPT

We also consider OpenAI's GPT models. Like BERT, the GPT family of models is pre-trained on a very large corpus that provides them with a general understanding of language in context. GPT is therefore well-suited for classification tasks, including identifying topics or events in news articles. The literature variously tests on different generations including 3.5 [4, 12] and 4 [16, 10], complicating comparisons. For clarity, we test four generations of GPT models but only discuss the results of GPT-4 here, leaving comparisons to earlier generations for the supplementary appendix. The modeling process for GPT and other openAI models is presented in Table 3.

Currently researchers can only fine-tune GPT models up to version 3.5 [8] to perform specific classification tasks. Therefore, we focus on prompt engineering – arguably considered by the literature as the most important procedure when using GPT models – to address reliability and validity concerns with respect to zero-shot applications [21, 12, 19].

To engineer prompts, we simulated text about protest and non-protest events, varying levels of detail and specificity (see appendix for details). The best-performing engineered prompt – both in precision and recall for classifying our data – was:

Engineered prompt: *Classify protest events based on contextual cues. Consider keywords like "protest," "demonstration," "rally," "strike", "march", and "sit in". Protests could be violent or symbolic forms of resistance. Examine contextual information such as location, participants (groups, organizations, activists, advocacy groups, specific communities), event date, and time. Check for motivations, demands, grievances, and the presence of law enforcement when large groups gather for a cause. Use 'yes' or 'no' to indicate if the article describes a protest event.*

We use the Chat Completions API with this prompt on the same five folds of data used for the other models. Because GPT models can respond differently over multiple iterations to the same prompt, we run the GPT-4 model three times over the five folds (altogether 15 runs) using the same prompt, take the modal output value for each fold, and compare that to the human-coded output. We set the temperature hyperparameter for this set of runs at 0 (no randomness in output). For robustness, we also vary the temperature as recommended in the literature [12] as explained below. We use the *httr2* package [20] in R to interact with the API.

Table 3. Overview of the GPT Model Process

1. Obtain corpus of relevant documents for analysis.
2. Construct prompts that accurately define coding task for the GPT model to perform.
3. Test accuracy of prompt responses on a small set of data.
3. Pick prompt that performs best and run the OpenAI API request for test sample of articles in corpus.
4. Run final prompt multiple times on test data.

Additionally, as discussed below we compare the performance of our engineered prompt to the following simple baseline prompt: *"Identify with 'yes' or 'no' whether the following article (delimited in XML tags) mentions a protest event: <article>text</article>"*. (For detailed results, see supplementary appendix).

Next, we explain the collection and coding process for the data. We use this data to compare the results of Newsmap, BERT, and GPT-4 classification to human coders.

Data

Our initial set of labeled news articles contained 1,718 unique documents. Of the 1,718 documents, only 454 (26%) observed protest. Such class imbalances are problematic for both training and assessing model performance. This is because the resulting metrics can be biased towards majority-represented classes. To account for this, we down-sample the data when performing fine-tuning and testing so that all 454 observations of protest are present alongside a random sample of 454 "not protest"-labeled texts. The resulting sample contains 908 unique documents evenly split by labels that we use to train and test *Newsmap*, fine-tune and test BERT, and run our prompts on GPT-4.

To compensate for the loss in observations when assessing model accuracy via down-sampling, we run all the models using five-fold cross-validation. K-fold cross-validation takes k sub-samples of the data and then trains and tests the model on said sub-sample (fold). Each model uses the same indices for each fold to improve result comparability – i.e., observations are the same across models. As is the case when using cross-validation techniques, the final model performance metrics are the average prediction results across the five test sets created for each fold.

Results

Five-fold Cross Validated Results

Table 4 presents the average precision, recall, and F1 scores across the three models. *Newsmap* performed the worst, with lower mean recall and precision scores than the other models. As expected, *BERT* had greater reliability, with a mean precision of 0.737 and recall of 0.717. Thus, the F1 score reflects the model's higher level of precision. The GPT-4 model with the engineered prompt outperforms the bag-of-words model when considering overall performance (F1); however, the fine-tuned BERT model has a slightly higher precision score.

Table 4. Performance metrics across three models for identifying *protest* label in news articles.

Model	Precision	Recall	F1
Newsmap	0.620	0.533	0.568
BERT	**0.759**	0.789	0.765
GPT-4	0.739	**0.938**	**0.821**

Metrics represent average results for models using five-fold cross-validation. Bold numbers indicate model with the highest score for the given metric.

Figure 1 shows the ROC/AUC curves for each model's performance across each fold of the five-fold cross-validations. The y-axis shows the rate of true positives; and the x-axis, the false positive rate. Some folds in the assessment received low accuracy results across all three models (Newsmap, Bert, GPT). This suggests the results are likely a lower-bound for all models because some of the data was harder to classify. The curves suggest similar results to the metrics in Table 4, but they also highlight differences between the approaches. First, BERT has the highest true positive rate of the models – arguably the most important statistic when coding rare events.

Prompt engineering, temperature, text complexity

We also compare the performance of GPT-4 when using our baseline and our engineered prompt. Interestingly, and contrary

Fig. 1. AUC graphs for three models predicting protest in news articles. Black lines are ROC results from individual cross-folds when testing using five-fold cross-validation. Red lines are overall ROC curves. Results generated in *R* (v 4.2.1) using the *cvAUC* package (v 1.1.4)

to our expectations, Table 5 shows that GPT-4's ability to classify only true protest events (precision) was consistently better when using the baseline prompt, whereas the model's ability to detect all protest events in the data (recall) was consistently better with the engineered prompt.

Table 5. GPT-4 Performance After Prompt Engineering (Average five folds)

	Precision	Recall	F1
Engineered prompt	0.74	0.94	0.82
Base prompt	0.78	0.91	0.83

To test the effect of temperature on model performance [12], we varied the temperature hyperparameter for GPT-4 with the engineered prompt on a sample of articles. The accuracy metrics when the temperature hyperparameter is set at 2 is comparable to those when the temperature hyperparameter is set at 0 – thus demonstrating temperature variance did not affect our modeling substantially. However, because this analysis was run on a sample and there is a slight improvement with higher temperature setting, we conclude that this is an important topic for further research.

Finally, to allow for a more direct comparison between humans and the machines in the above test, we fed the machines the exact same articles that the humans were asked to code. To test GPT-4 performance across more or less complex text, which may matter to performance [3], we heavily pre-processed a balanced sample of articles of median length. We did not find evidence that substantial pre-processing improved LLM's performance. For further details on all robustness checks, see supplementary appendix.

GPT-4 Performance

Why did GPT-4's overall accuracy (F1) fail to consistently exceed that of humans – per earlier work [4, 16]? We suggest several explanations. First, our human coders received iterative training – setting a higher, and arguably more realistic, bar for comparison.

Second, because the data are new, we need to rethink the notion of human coded data as ground truth. We can think of the human coder ICC as measuring the consistency across coders. In contrast, machine accuracy is measured by comparing how well the machines replicate human coding. While ICC scores measure consistency, the variance in scores can indicate the complexity of a coding task, which likely affects the accuracy of human coding. The machines can be internally consistent in their own coding (with GPT-4 Cronbach's alpha being above 0.90 across iterations). However, when replicating human coding the best they do is capturing around 80% (F1) of cases coded by the humans.

There is a caveat to the presumed quality of human coding captured by the ICC scores. Specifically, humans are only highly consistent when coding well-known categories such as reports about African-Americans. Once we shift to less well-known categories such as classifying reports about Dalits, ICC scores drop as low as 0.48. With improvements in model precision – potentially achieved through multi-shot applications and fine-tuning (once this feature becomes available for GPT-4) – transformers may improve in their overall accuracy. Furthermore, the consistency in transformer coding metrics across all cases suggests they may rival human coders when coding cases where humans do not have area expertise [13, p. 28].

Third, task variance has a substantial effect on LLM performance [10] for reasons not yet well articulated in the literature. Our results showed that the recall metric (false negatives) of LLMs approached the human coding, while the precision metric (false positives) fared worse. As such, the performance of LLMs varies not only in overall accuracy as noted in the literature, but also with respect to the specific task at hand – e.g., whether the goal is to classify only true positives or to find all positives. Moreover, we found that the baseline (simple) prompt produced fewer false positives (precision) in classifying protests, while the engineered prompt consistently produced fewer false negatives (recall). Addressing calls in the literature for better prompt engineering [21], we add that the need for prompt engineering is likely task dependent. In contrast to the simple classification performed in this article, prompt engineering may be more important for complicated classifications such as conditional statements or multistage classification. Furthermore, the need for prompt engineering likely depends on the desired outcome and is probably more important in early stages of data collection where completeness outweighs concerns with mis-classification.

Conclusion

These results demonstrate that human validation is still important and necessary to ensure construct validity. However, LLMs may prove useful for helping coders make decisions about new data, especially when coding rare events and/or less familiar contexts. The GPT models, in particular, demonstrated strong performance without fine-tuning. Thus, startup costs are low. Moreover, if coding records already exist, they can be used to calibrate machine coding for improved reliability and validity. In spite of this discussion, there are important caveats to consider.

Notably the internal processing of the best performing models BERT and GPT-4 is opaque; moreover, these models are still prone to error. Even so, the LLMs demonstrate sufficient internal consistency and accuracy with respect to human coding to be an important step in making human coding of data more efficient. Human coders simply cannot process and annotate the same volume of material as the machine. Therefore, researchers may want to consider a reverse workflow where model results are used as a "screener test" to highlight evidence of protest subject to human verification. Furthermore, machine output can be set to reference sources – thus allowing coding decisions to be easily verified. In this way, machine-coding can help improve the transparency of output.

Importantly, we remain agnostic about whether we should consider human coding or LLM classification as the "ground truth." For example, human coders can possess similar biases that lead to similar coding decisions, and thus bolster accuracy metrics. As such, we contend that the two methods are complementary. Together, they should be used for triangulation. When the classifications of the two methods align, we should have greater confidence in their results.

Conversely, when classifications diverge, we should proceed with caution and ask why.

Furthermore, variations in human coder reliability indicate that spatial and temporal [11] contextual knowledge is highly relevant for expert-coding accuracy. In contrast, machine coding showed greater consistency across cases. One drawback is, as noted, that the best-performing algorithm – GPT – is also the most opaque. Thus, best practices for a particular task may be subject to experimentation and not necessarily applicable across tasks – at least not in a straightforward manner. All this suggests that the comparison of machine-coding to human-coding across tasks remains a topic for further study.

Finally, machine-coding methods are easily adaptable. They can address new questions or subject matters not just from different places and time, but also across different units and outcomes. This can potentially facilitate broad spatial and time-series analyses of politics – including at the sub-national level, where definitions and boundaries of groups and concepts change over time [11].

Funding

The author(s) received no financial support for the research, authorship, or publication of this article.

Data Availability

All data and code used for analysis and creating visualizations will be included in publicly available replication files. The replication files will be made available on Harvard Dataverse as "Replication files for Coding with the machines". Additionally the files will be available on Github at www.github.com/j-dewar/amar_and_the_machine.

References

1. Jóhanna K. Birnir, David D. Laitin, Jonathan Wilkenfeld, David M. Waguespack, Agatha S. Hultquist, and Ted R. Gurr. Introducing the AMAR (All Minorities at Risk) Data. *Journal of Conflict Resolution*, 62(1):203–226, 2018.
2. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805v2 [cs.CL], 2019.
3. Zican Dong, Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. Bamboo: A comprehensive benchmark for evaluating long text modeling capacities of large language models, 2023.
4. Alizadeh M. Gilardi, F. and M. Kubli. Chatgpt outperforms crowd-workers for text-annotation tasks. arXiv:2303.15056v2 [cs.CL], 2023.
5. Justin Grimmer and Brandon M. Stewart. Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, 21(3):267–297, 2013.
6. Clotilde Napp. Gender stereotypes embedded in natural language are stronger in more economically developed and individualistic countries. *PNAS Nexus*, 2, 2023.
7. R Shen A Macanovic A Chatelain Ollion, E. Chatgpt for text annotation? mind the hype! *osf.io*, 2023.
8. OpenAI. Open AI developer platform: Fine tuning. *https://platform.openai.com/docs/guides/fine-tuning*, 2023.
9. OpenAI. Open AI developer platform: Prompt engineering. *https://platform.openai.com/docs/guides/prompt-engineering*, 2023.
10. Samuel Wolken Neil Fasching Pangakis, Nicholas. Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning. arXiv:2306.00176v1 [cs.CL], 2023.
11. Baekkwan Park, Kevin Greene, and Michael Colaresi. Machine learning human rights and wrongs: How the successes and failures of supervised learning algorithms can inform the debate about information effects. *Political Analysis*, 27:223–230, 2019.
12. M.V. Reiss. Testing the reliability of chatgpt for text annotation and classification: A cautionary remark. arXiv:2304.11085v1 [cs.CL], 2023.
13. Philip Schrodt and David Van Brackle. Automated coding of political event data. In V.S. Subrahmanian, editor, *Handbook of Computational Approaches to Counterterrorism*, pages 23–49. Springer, New York, NY, 2013.
14. Philip A. Schrodt, Shannon G. Davis, and Judith L. Weddle. Political science: Keds—a program for the machine coding of event data. *Social Science Computer Review*, 12(4):561–587, 1994.
15. Usman Naseem Mehwish Nasim Thapa, Surendrabikram. From humans to machines: Can chatgpt-like llms effectively replace human annotators in nlp tasks? *Association for the Advancement of Artificial Intelligence*, Workshop Proceedings of the 17th International AAAI Conference on Web and Social Media, 2023.
16. Petter Törnberg. Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning. arXiv:2304.06588v1 [cs.CL], 2023.
17. Sebastián Vallejo Vera. Rage in the Machine: Activation of Racist Content in Social Media. 65(1):74–100, 2023.
18. Kohei Watanabe. Newsmap: A semi-supervised approach to geographical news classification. *Digital Journalism*, 6(3):294–309, March 2018.
19. Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023.
20. Hadley Wickham. *httr2: Perform HTTP Requests and Process the Responses*, 2022. R package version 0.2.2.
21. Zhou K. Li J. Tang T. Wang X. Hou Y. Min Y. Zhang B. Zhang J. Dong Z. Zhao, W.X. and Y. Du. A survey of large language models. arXiv:2303.18223v13 [cs.CL], 2023.
22. Peixian Zhang Ehsan-Ul Haq Pan Hui Gareth Tyson Zhu, Yiming. Can chatgpt reproduce human-generated labels? a study of social computing tasks. arXiv:2304.10145v2 [cs.CL], 2023.