Political Analysis (2025), **0**, 1–14
doi:10.1017/pan.2024.31

ARTICLE

Synthetically generated text for supervised text analysis

Andrew Halterman

Michigan State University, East Lansing, MI, USA

E-mail: ahalterman0@gmail.com

(Received 9 April 2024; revised 26 August 2024; accepted 26 August 2024)

Abstract

Large language models are a powerful tool for conducting text analysis in political science, but using them to annotate text has several drawbacks, including high cost, limited reproducibility, and poor explainability. Traditional supervised text classifiers are fast and reproducible, but require expensive hand annotation, which is especially difficult for rare classes. This article proposes using LLMs to generate synthetic training data for training smaller, traditional supervised text models. Synthetic data can augment limited hand annotated data or be used on its own to train a classifier with good performance and greatly reduced cost. I provide a conceptual overview of text generation, guidance on when researchers should prefer different techniques for generating synthetic text, a discussion of ethics, a simple technique for improving the quality of synthetic text, and an illustration of its limitations. I demonstrate the usefulness of synthetic training through three validations: synthetic news articles describing police responses to communal violence in India for training an event detection system, a multilingual corpus of synthetic populist manifesto statements for training a sentence-level populism classifier, and generating synthetic tweets describing the fighting in Ukraine to improve a named entity system.

Key words: large language models; text analysis; document classification; synthetic data

Edited by: Jeff Gill

1. Introduction

Researchers in political science are rapidly adopting generative large language models (LLMs) to assist in the analysis of text. Generative LLMs, such as GPT-3, Llama, and many others, show great promise in labeling or extracting information from documents (Ornstein, Blasingame, and Truscott 2022; Wu *et al.* 2023; Ziems *et al.* 2024), tasks that researchers have traditionally done by hand-labeling documents and training supervised learning models. A common approach to using LLMs for text analysis is to provide a prompt describing the task and available categories, along with the text to be analyzed. The LLM generates a response, including the category label it assigns to the document or the other information the prompt elicits.

While using LLMs to label documents directly is simple to implement and performs well on some tasks (Gilardi, Alizadeh, and Kubli 2023; Rytting *et al.* 2023; Ziems *et al.* 2024), it faces several drawbacks. The most capable LLMs are closed source models behind commercial APIs. These models change rapidly and can be deprecated with little warning, making it difficult to reproduce previous results (Spirling 2023). This has implications beyond reproducibility: researchers who update text-derived datasets over time with new documents cannot apply a consistent coding scheme if the original LLM is not available. Second, LLMs are difficult to combine with hand labeled data without a separate and technically challenging fine-tuning step (Longpre *et al.* 2023). Third, while LLMs are clearly cheaper than hiring human annotators (Ornstein *et al.* 2022), running them over a large corpus can be expensive in time, equipment, or API fees. Finally, the process by which the LLM provides a document label is inscrutable, particularly in the case of closed-source commercial LLMs, and even state-of-the-art LLMs fail on specific tasks (Peskoff and Stewart 2023).

The traditional approach to supervised text analysis—hand-labeling text and training a relatively small supervised classifier—avoids many of these issues. Traditional supervised models are well understood, can be run on local hardware, and can be iteratively improved through well understood processes of hyperparameter tuning and model selection. However, it depends on extensive efforts to collect labels. The annotation process is especially onerous if researchers are annotating a rare class, which requires that a large number of documents be annotated to obtain a sufficient number of positive examples.

This article suggests an alternative use of LLMs for supervised text analysis tasks: to generate synthetic text as training data to fit a traditional supervised learning model. The synthetic text can augment a small set of hand-labeled data, be used on its own to train a classifier, or, in some cases, be hand annotated. Using synthetic training text addresses some of the problems with using LLMs to directly analyze text. Generating text, as opposed to analyzing it, is a much more straightforward task for LLMs, allowing researchers to use smaller local LLMs, greatly improving reproducibility and lowering costs. Researchers can publish their synthetic training data and models, allowing others to reproduce their results. The synthetic text can be inspected for validation and used alongside real text to fit efficient local classifiers that researchers have more control over.

To use synthetic text as training data, researchers need to take three steps. First, they must be able to guide or control the content of synthetic text. By *prompting*—changing the input to the LLM to guide its generation—*adapting*—updating the model weights using a specific text corpus—or changing how tokens are *sampled* from the LLM—researchers can guide the style and content of the synthetic text the LLM produces. Second, they validate the quality and coverage of the synthetic text. Finally, the synthetic training text is used to fit a supervised model, either by *augmenting* an existing set of hand-labeled data to improve a classifier's performance, or, in some cases, used without hand labeling to train a document classifier. In the latter case, this use of synthetic text can be seen as a form of *model distillation*—using a larger, more capable model to train a smaller, narrowly focused model. Using LLMs to generated synthetic training data uses them for their strengths—generating text—while avoiding their weaknesses—opaque classification, high computational costs, a lack of reproducibility, and difficulty in combining with hand-labeled data.

This article describes how researchers can use LLMs to lower the costs of supervised text analysis by using synthetic text. It provides guidance on generating and using synthetic text, when synthetic text can augment existing data and when it can be used on its own as a form of model distillation, introduces techniques for measuring and improving the quality of synthetic text, and discusses the ethical pitfalls inherent in using synthetic text. It presents three short validations, illustrating how the approach can create synthetic news articles for training a model to classify police responses to violence in India, a multilingual sentence-level populism classifier with no hand-labeled examples, and synthetic tweets describing the war in Ukraine for a named entity recognition model. It shows that augmenting hand-labeled text with synthetic data improves performance and that high-quality synthetic on its own can be used to train a model without any hand labeling. It also shows that synthetic text can differ from real text in quantitatively measurable ways that a marginal synthetic example is usually less informative than a marginal real example. However, the near-zero cost of producing a marginal synthetic example offers major benefits for applied researchers developing text classifiers. The method does not depend on a particular LLM architecture, making it applicable even as LLMs rapidly improve.

1.1. Previous Work on Synthetic Data

This approach, generating synthetic training data from LLMs, differs from the common use of LLMs in political science to directly annotate documents discussed above (e.g., Ornstein *et al.* 2022; Wu *et al.* 2023). Instead, it builds more directly on synthetic data techniques from computer science and

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

NLP. Using state-of-the-art LLMs to generate synthetic data to train smaller models is an increasingly common technique in machine learning. Large, capable LLMs can generate training data for smaller LLMs, including by writing synthetic "textbooks" to train on (Gunasekar *et al.* 2023) or by writing instruction–output pairs to train smaller instruction fine-tuned models (Meng *et al.* 2022; Taori *et al.* 2023; Wang *et al.* 2023). The success of these approaches suggests that the abilities of larger models can be "distilled" into smaller models *via* synthetic text generated by the large models. While the computer science literature provides some practical advice for generating synthetic training text from LLMs (e.g., He *et al.* 2022; Yu *et al.* 2024), these techniques are largely focused on training LLMs, as opposed to training supervised models, and the approach has not yet been adopted in political science.

Earlier data augmentation techniques produce semi-synthetic data from real text to expand a limited set of training data. Rule-based systems modify text by randomly inserting/deleting words or replacing words with their synonyms (Wei and Zou 2019), or augmentation can be done by using LLMs to rewrite text (Edunov *et al.* 2018; Schick and Schütze 2021). These two existing concepts, model distillation and data augmentation, motivate this article's use of synthetic text in training supervised classifiers.

2. Using Language Models to Generate Synthetic Text

To be useful as training data, synthetic text must have the content and style that researchers require. To preview the first validation, a researcher training a classifier to identify reports of Indian police making arrests requires a set of positive examples (stories containing arrests) and negative examples. To use synthetic text as training data, a researcher needs a way to reliably produce documents with the desired content. A brief formalization of LLMs illustrates the three available options.

2.1. Formalizing Text Generation

Generative language models learn to produce text by optimizing a language modeling objective: conditional on a sequence of tokens (words), they predict which token is likely to follow. Formally, given a set of tokens $W = \{w_1 \ldots w_n\}$, a language model decomposes probability of the sequence into the probability of each token given the previous sequence of tokens: $p(W) = \prod_{i=1}^{n} p(w_i | w_{i-1}, \ldots, w_2, w_1)$. The conditional probability of the next token is modeled with the previous tokens, trainable parameters θ, and a function f:

$$\hat{p}(w_i) = f(w_{i-1}, w_{i-2}, \ldots, w_1, \theta). \tag{1}$$

To generate text from a language model, a token $\hat{w}_i$ is sampled from the predicted distribution over the next word $\hat{p}(w_i)$, using a set of generation parameters γ:

$$\hat{w}_i \sim \hat{p}(w_i), \gamma. \tag{2}$$

The generation parameters γ control how words are sampled from the probability distribution over the next word. For transformer-based models (e.g., GPT, Llama), common parameters include the "temperature", "top K", and "top P", which control whether to sample a high-probability next token (leading to simple, repetitive text) or favor low probability next tokens (leading to more creative but potentially nonsensical text).[1]

[1] Briefly, temperature modifies the softmax function

$$p(w_i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

to concentrate more probability in high probability words (low temperature) or to smooth the probability distribution (high temperature). Top-K limits sampling to the K highest probability words, while top-P limits to the words that together include proportion P of the total probability. See Platen (2020).

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

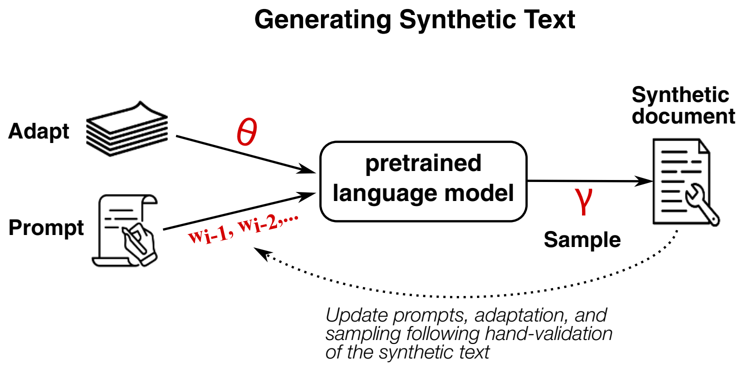
Generating Synthetic Text
Adapt
θ
Prompt
wi-1, wi-2,...
pretrained language model
γ
Sample
Synthetic document
Update prompts, adaptation, and sampling following hand-validation of the synthetic text

Figure 1. Overview of options for controlling synthetic text generation. Researchers can affect the content and style of synthetic documents by changing language model parameters (θ), by providing new prompts ($w_{i-1}, w_{i-2}, \ldots$), or by changing the sampling parameters (γ). Researchers then decide how to use the synthetic text as training data.

Table 1. Overview of the three approaches to controlling synthetic text generation.

	Advantages/Uses	Limitations	Examples
Prompting			
$p(w_i \vert w_{i-1}\ldots,\theta)$	• No existing corpus needed • Simple to implement • Easily controllable	• Local, open source models struggle with abstract prompts • Requires manual experimentation	• News stories about police responses to communal violence (Val. 1) • Populist sentences (Val. 2)
Adaptation			
$p(w_i \vert w_{i-1}\ldots,\theta)$	• Provides text very similar to existing corpus • Useful for expanding rare document classes	• Requires existing corpus • Computationally expensive/requires a GPU	• Ukraine war tweets (Val. 3)
Sampling			
$\hat{w}_i \sim \hat{p}(w),\gamma$	• Can improve text quality • Used in conjunction with adaptation or prompting • No training of the model required	• Controls style more than content • Requires existing corpus for adversarial method • If no corpus is available, requires manual tuning	• Ukraine war tweets (Val. 3)

2.2. Step 1: Controlling Synthetic Text Generation

Thus, applied researchers who would like to influence which token $\hat{w}_i$ is produced next have three options: they can *prompt* by changing the previous sequence of tokens ($w_{i-1}, w_{i-2}, \ldots$), they can *adapt* the parameters θ used to change the distribution $\hat{p}(w_i)$, or they can vary γ to change how the next token is sampled from the distribution over the next token. Figure 1 provides an overview, and Table 1 summarizes each approach. These techniques are general, working on current transformer-based neural networks, but also on older technologies such as recurrent neural networks and likely on future language models as well.

If a researcher *prompts* an LLM with the beginning of a document or an instruction ($w_{i-1}, \ldots w_n$), an LLM can generate a plausible continuation of the document. For example, a researcher can generate a

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

news story describing police making arrests by hand-writing a headline related to arrests. A prompting approach is illustrated in the first validation to generate news stories describing police actions in India by providing manually written headlines to elicit stories with the desired event types.

Recent "instruction tuned" language models can generate text from general prompts that describe the desired output, rather than simply the starting tokens of some desired text. For instance, GPT-4 or Llama can be provided with a prompt such as "write a political manifesto supporting free trade" and obtain a plausible output without the need to provide the first sentence of the manifesto. Prompting with an explanation of the desired text is used in the second validation to generate populist party manifestos in 22 languages for 27 European countries.

Prompts are easy to write and tweak and can be published, allowing other researchers can assess whether prompts accurately describe the concept being prompted (see, for example, the definition of populism used in the second validation). However, good prompts require manual experimentation and prompting will fail if the task is outside the LLM's pretraining data.

Second, *adaptation* updates the weights θ of a pretrained model to affect the content or style of generated text. Off-the-shelf pretrained language models may not reflect a researcher's desired style or content, especially if it was not present in the original pretraining data. Adapting an LLM consists of providing it with additional unlabeled text from a specific domain and updating the weights θ in the model to guide the text that the model produces. I use the term "adaptation" rather than "fine-tuning" to avoid ambiguity in how the term "fine-tuning" is used.[2] The adapted model can then generate additional synthetic training examples. An adaptation approach is used in the third validation to generate synthetic tweets about the war in Ukraine that closely resemble real tweets.

Adaptation may be necessary when a researcher's text is outside the domain of the training data of the pretrained model, can expand a small set of training documents into a larger synthetic corpus without the need for prompts, and provide them with a version of their dataset that they can share freely. However, adaptation is more technically challenging, requiring an existing corpus of text and GPU infrastructure.

Third, varying the generation parameters γ changes how the next token is sampled from the distribution over the next token. This generally changes the style of the text, rather than its content, and is useful in conjunction with either adaptation or prompting to produce more realistic text. Little theoretical guidance exists on how to select hyperparameters (Fu *et al.* 2021), but a technique introduced below helps select better hyperparameters. The final validation varies sampling hyperparameters to generate more realistic tweets about the war in Ukraine.

2.3. Step 2: Evaluating the Quality of Synthetic Text

After generating synthetic text, a researcher needs to validate that it is similar to real text, reflects the desired labels, and covers the semantic space of the real text. As with all text analysis, manual validation of the (synthetic) corpus is important. Synthetic augmentation can be considered another hyperparameter in a classification pipeline, similar to other text preprocessing decisions. Selecting the amount and type of synthetic text and avoiding overfitting can be accomplished with standard train/eval/test splits or cross-validation.

I also introduce two quantitative approaches to improving the quality of the synthetic text. First, good synthetic text is difficult to distinguish from real text. A simple "adversarial" procedure helps select the best generation hyperparameters γ for generating text if a reference corpus is available. The adversarial procedure builds on the intuition that the less a classifier can distinguish between real and synthetic text,

[2] The NLP literature uses several terms to describe the process of updating a pretrained model's weights using new text, including "fine-tuning", "domain adaptation", or "additional pretraining". "Fine-tuning" often refers to training the model to perform a new task, such as instruction following on prompt-answer pairs or supervised fine-tuning for classification tasks, rather than the language modeling objective.

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

the higher the quality of synthetic text, and thus the more useful the synthetic text will be for training supervised learning models.

For each set of hyperparameters γ, a researcher generates n synthetic documents from γ and samples n real documents from the existing corpus. They then train a classifier on a training set of both synthetic and real documents with the objective of predicting whether the document is real or synthetic. The set of hyperparameters that results in the lowest classification accuracy is the set that generates the most realistic synthetic documents and should be used to generate the final synthetic text. Further details are provided in Validation 3 and Section D.1 of the Supplementary Material.

Second, a synthetic training corpus should cover the semantic space of the real corpus. Document embeddings have a useful property of placing semantically similar documents near to each other in embeddings space. A good synthetic training corpus should have a similar distribution in embedding space to the real documents. Areas of poor overlap can yield misclassification, as the classifier does not have examples to estimate a decision boundary. A simple technique for checking the coverage of the synthetic corpus, demonstrated in Validation 1 and expanded in Section B.3 of the Supplementary Material, consists of embedding the real documents and plotting their distribution in two dimensional space. Overlaying the embedded synthetic text in the same space provides a quick visual check of its coverage. If gaps are present, new prompts can be added to fill in the poorly covered areas.

The two new quantitative approaches introduced here—the adversarial procedure and the embedding space coverage check—address two of the specific requirements of using synthetic text to train supervised models and compliment existing techniques in NLP for synthetic text detection (e.g., Ippolito, Duckworth, and Eck 2020; Zellers *et al.* 2019).

2.4. Step 3: Using Synthetic Text as Training Data

After generating synthetic text, researchers have three options for how to use it in a supervised learning pipeline. First, they can use the synthetic text as a form of *data augmentation*. Adding synthetic data (with inferred labels) to a limited set of hand labeled data can increase the size and diversity of the training set. Validation 1 demonstrates that synthetic augmentation produces police event classifiers that are more accurate than those trained on hand-labeled data alone. The augmentation approach is especially valuable when the concept of interest is rare, as it allows researchers to generate a large number of positive synthetic examples.

Second, they can use synthetic text for a form of *model distillation*, training a supervised model entirely on synthetic text. Researchers do not need the full capabilities of an LLM, but only the portion that encodes the desired information about the political science document annotation task. By generating synthetic text from the LLM and training a smaller supervised model on the synthetic text, researchers distill the relevant knowledge of the LLM into a smaller, simpler model that can be used to annotate real documents. This approach is used in Validation 2: the knowledge that GPT 3.5 has of populist rhetoric is transferred using synthetic text to a specialized, efficient, and accurate sentence-level model.

Third, in some limited circumstances, researchers can hand-label synthetic text directly to use as training data. Validation 3 shows that this approach carries costs in accuracy. A hand-labeled synthetic tweet is less informative to the model than a labeled real tweet. In some cases, this approach might be useful, for instance, to produce a hand-labeled, copyright-free benchmark dataset for other researchers, in situations where real text is too sensitive or restricted to show annotators, or when researchers do not have direct access to the raw text data.

In all three uses, the hyperparameters of the local classifier can be tuned to perform better on the development set. In contrast, tuning an LLM to label text directly requires rerunning the entire (expensive) LLM step. When labeling text in social science, LLMs are very sensitive to the prompt and instructions (Atreja *et al.* 2024), and experimenting with alternative instructions is much more costly than tuning local classifiers.

Once the final model is trained, the synthetic training data is discarded and the newly trained classifier is applied to real text to obtain predicted labels. As with all labels produced by machine learning

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

models, the labels from a classifier trained on synthetic text will be noisy. Recent work has shown that label error can cause severe bias and confidence interval coverage issues (Egami *et al.* 2024; Fong and Tyler 2020). Egami *et al.* (2024) show that calibrating a model's predictions using a set of expertly labeled examples can mitigate the problem, underscoring that while synthetic data can reduce labeling costs for training data, some expert hand labeling will still be required. Future work is needed to understand the effects of different models and prompts on the accuracy of classifiers trained on their output.

2.5. Ethics and Limitations

Generating synthetic text presents serious ethical concerns. Synthetic text can include factual errors, conspiracy theories, or offensive statements and will reproduce the societal biases of the model's training data (Caliskan, Bryson, and Narayanan 2017). To avoid any possibility of synthetic text being mistaken for real text, researchers working with synthetic text should always attach a disclaimer directly to any synthetic text any time it is saved or stored, clearly indicating that the text is synthetic. Annotators should be briefed on the use of synthetic text. Any synthetic training text reported in published work must be clearly marked.

LLMs raise environmental concerns over resource usage. Here, the synthetic approach has an advantage over using LLMs to label text directly. As the validations show, even relatively small LLMs can generate high quality synthetic text, and tuning a local classifier on synthetic text does not require rerunning an LLM step, as direct labeling requires.

Any use of LLMs also raises data provenance and copyright concerns. LLMs are trained on unlicensed, copyrighted text and whether this training is "fair use" is an open question. As the validations show, even relatively small non-instruction tuned models can generate high quality synthetic text, raising the possibility of using smaller LLMs trained entirely on public domain text (Langlais 2024).

Despite some recent proposals to use language models to approximate survey responses (Argyle *et al.* 2023), synthetically generated text should *never* be used to draw substantive conclusions. While synthetic text may be difficult to distinguish from real text and thus useful for training a model to recognize certain linguistic patterns, its factual content will be imaginary and thus completely unsuitable for answering substantive questions on its own. Instead, synthetic text should only be used to train a model, then discarded. Only real text should be used for analysis after the model is trained.

3. Validations

The following section presents three short validations to show the effectiveness of the approach and illustrate the decisions researchers face in generating synthetic text. Where possible, I use local open-weight LLMs to enhance reproducibility and to demonstrate the efficacy of even small or "obsolete" models in generating synthetic text. The validations cover a range of subfields and tasks and show that the best approach to generating synthetic text and which supervised model to fit depend on the application. Table 2 summarizes the validations, including the LLM and supervised model used for each.

Table 2. Overview of the validations.

Dataset	Task	LLM	Guidance	Usage	Classifier
Police events in India	Document classification	gpt-2-xl	Prompt	Augmentation	Sentence transformers + logit
Populism in manifestos	Document classification	GPT-3.5	Prompt	Distillation	Sentence transformers + SetFit
Weapons in Ukraine tweets	Information extraction	gpt-2-large	Fine-tune	Hand label	SpaCy named entity recognition

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

3.1. Validation 1: Labeling Rare Event Types with Synthetic Augmentation: Identifying Police Actions in the 2002 Gujarat Violence

Following the death of 59 Hindu pilgrims in a train fire in late February 2002, the Indian state of Gujarat experienced widespread communal violence, directed primarily at Muslims. The qualitative research on the violence emphasizes the role of the police in the violence, including their failure to respond to ongoing violence (Subramanian 2007; Wilkinson 2006). Halterman *et al.* (2021) introduce a new annotated corpus, the IndiaPoliceEvents dataset, focusing on police actions taken during this period. The dataset includes 1,257 articles (21,391 sentences) from the *Times of India* covering the period after the train fire. Each sentence is labeled by two or three trained annotators for a set of actions taken by police. The dataset reflects the common rare class problem in social science document classification: only 9.7% of sentences involve any police action, and specific actions are rarer (only 0.45% of sentences describe police making arrests).

I conduct a set of experiments to evaluate the benefit of using synthetic data to augment hand-labeled data. To augment the real annotated documents, I generate around 1,600 synthetic news stories by prompting GPT-2 (xl) with a set of hand-written headlines to elicit police actions, other mentions of police, and non-police events. I write 2-6 headlines for each event type in the India Police Events event schema (see Section A.1 of the Supplementary Material) and generate multiple stories for each headline. The synthetic stories are labeled according to the class of the headline (e.g., story from a headline describing arrests receives the ARREST label).

To ensure coverage of the semantic space, I plot the real and synthetic text embeddings in 2D PCA space as described in Section 2.3. Doing so revealed that the initial set of synthetic text was missing discussions of party politics and discussions of a disputed temple site in Gujarat (see Section B.3 of the Supplementary Material). I then generate additional synthetic examples to discuss those issues (Figure SI 5 in the Supplementary Material).

I then train a binary classifier for each event class. I embed each document using a sentence transformer (`paraphrase-mpnet-base-v2`, Reimers and Gurevych 2019) and fit a logistic regression model with balanced class weights on the (frozen) embeddings. This model provides a good balance of computational efficiency and performance. As a baseline, I also include a traditional data augmentation technique, which adds/deletes/swaps words from the real text to create additional training documents (Wei and Zou 2019). Section A.2 of the Supplementary Material provides more details and an alternative experiment using active learning.

Figure 2 shows how classification F1 changes with varying amounts of human annotation and synthetic augmentation. Synthetic augmentation outperforms no augmentation, traditional augmentation, and synthetic text used on its own. Increasing the number of synthetic augmentation examples does not improve performance on all classes and offers only slight benefits when 1,000 human annotations are available. However, synthetic augmentation provides better classification accuracy with only 100–500 hand-labeled examples than 1,000 hand-annotated stories on their own. Given the reported annotation time in Halterman *et al.* (2021), annotating an additional 500 sentences with 2–3 annotators would take around 3.3–5 hours of work, excluding annotator training and overhead.

This validation demonstrates that even a small set of synthetic augmentation data, generated using a relatively old LLM, improves the performance of a supervised model trained on hand-annotated text alone. In applied work, a researcher would then apply the best classifier to their entire corpus, along with a separate labeled test set for final evaluation.

3.2. Validation 2: Synthetic Data Without Hand Labeling–Training a Sentence-Level Populist Classifier

As attention to populist parties has grown, so too has the methodological work on identifying populism in text, including in party manifestos (Dai and Kustov 2022; Di Cocco and Monechi 2021; Jankowski and Huber 2023). A key challenge has been to identify populism in short text, specifically sentences, to estimate the degree or amount of populism in a longer document. This validation trains a sentence-level classifier to identify populism across 27 European countries in 22 languages. It use a *prompting* approach

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

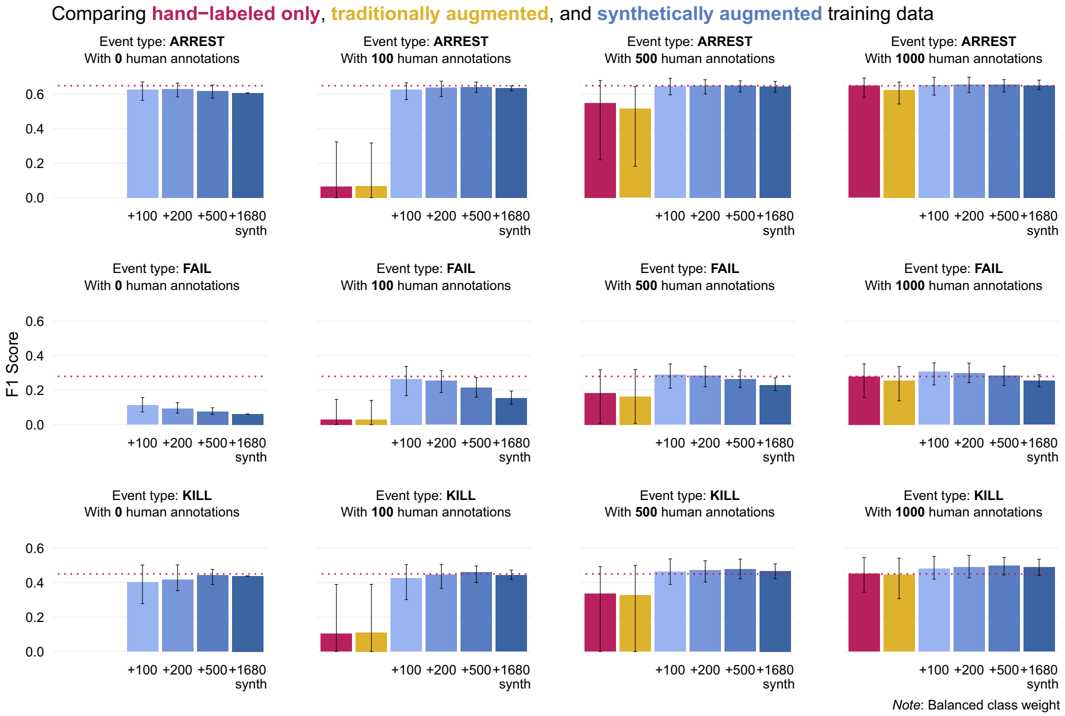

Figure 2. Mean F1 performance on an evaluation set for three classes (ARREST, FAIL, KILL) with increasing sizes of the hand-annotated training set (0, 100, 500, 1,000). The red bar shows performance with hand-labeled data only. Orange shows performance with traditional data augmentation (Wei and Zou 2019). The four blue columns show the performance with different numbers of synthetic training examples added to the hand-labeled data. Maximum human annotation performance is shown as a horizontal line. Error bars show the empirical 90% range across 50 samples.

to generate synthetic populist manifesto statements with GPT-3.5-turbo. It then fits a binary classifier on the synthetic data alone to identify populist statements in real manifesto text. I find evidence that the classifier trained on synthetic text alone reliably identifies populist rhetoric in text, but performance is limited by the difficulty of the task.

3.2.1. Measuring Populism

I employ a conceptualization of populism drawing on Mudde's (2004) "thin" definition of populism, which focuses on its rhetorical aspects and worldview. Because GPT-3.5 was trained on a multilingual corpus, a prompt can specify a country and language to obtain non-English training text, even with an English language prompt. By inserting each country and its associated language(s) into the prompt and varying the sampling hyperparameters (γ), I generate a set of 5,357 synthetic populist sentences (Table 3).

I also generate 36,509 negative examples—instances of non-populist text for the 22 countries—by prompting the LLM with policy position descriptions from the Manifesto Project codebook (Volkens *et al.* 2021) and an additional set of ten hand-written prompts to cue criticism of other political parties and dissatisfaction with current policies.

Putting all 1.1 million Manifesto Project sentences through a commercial LLM could cost several thousand dollars and its classification performance would depend on the ability of the model to precisely apply technical definitions from the codebook (Halterman and Keith 2024).[3]

[3] 1.1 million documents, with around 1,000 tokens per document+codebook prompt at 0.005 dollars per 1,000 tokens equals $5,500. Generating the synthetic training data cost around $2.

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

Table 3. Prompts used to generate populist text with language and country placeholders.

Description	Prompt
Populist (thin definition)	Populist rhetoric sees politics as a conflict with good, common, or “real” people on one side, and out-of-touch, evil, or self-serving elites on the other. Write ten statements that a populist party in {country} might make (in {language}): Example output: [SYNTH] *“We’re committed to giving voice to those who have been ignored or left behind by mainstream politics.”*
Populist (style prompt)	A populist party in {country} believes that politics is corrupted by self-interested elites, unelected bureaucrats, croynism, and big business. It wants to take power back for ordinary people. Write 12 statements that a {country_adjective} populist party might make (in the {language} language) in the style of a political manifesto: Example output: [SYNTH] *“We believe that the people of Ireland are sovereign, and that the government should be accountable to them.”*

Table 4. Performance of classifiers trained on synthetic documents and real labeled documents (1,136 of each) and evaluated on real Manifesto Project text with gold-standard labels.

Code	Description	Synth F1	Hand F1
201	Freedom and human rights	0.60	0.66
304	Political corruption	0.66	0.75
402	Incentives: Positive	0.47	0.59
403	Market regulation	0.37	0.48
414	Economic orthodoxy	0.60	0.70
416.2	Sustainability: Positive	0.62	0.75
502	Culture: Positive	0.77	0.78
504	Welfare state expansion	0.54	0.60
703	Agriculture and farmers: Positive	0.71	0.75
705	Underprivileged minority groups	0.44	0.67

I then train a supervised text classifier on the synthetic sentences to predict binary populism status. I use SetFit, an efficient technique for short text classification (Tunstall *et al.* 2022) to fine-tune a multilingual sentence transformer model (Reimers and Gurevych 2020). In training the model, I assume that the statements generated with the populist prompt are indeed examples of populist rhetoric, and that non-populist prompts generate non-populist rhetoric.

3.2.2. Validating the Populism Classifier—Performance on Known Manifesto Labels

As a first validation, I examine how well a classifier trained on synthetic text can recover known labels from the Manifesto Project. For each of the ten most commonly used policy codes, I train a classifier using the same architecture on the synthetic text generated from the Manifesto Project codebook descriptions discussed above and evaluate against the Manifesto Project’s gold-standard labels. I compare the results to the performance of a classifier trained on the same number of real manifesto statements, finding that the synthetic-only classifier performs almost as well on each category (Table 4). This provides evidence that the model can accurately predict the labels of manifesto sentences, even when trained on purely synthetic text.

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

3.2.3. Validating the Populism Classifier—Hand Labeling

I then validate the populism scores directly using a small set of hand-annotated statements. I apply the newly trained populism classifier to each of the million sentences or phrases in the Manifesto Project corpus, producing a continuous predicted $[0,1]$ populism score for each. I then collect hand annotations on a sample of 450 English-language manifesto statements and evaluate the predictions of the model. It achieves an accuracy of 0.87 and a macro F1 score of 0.76 (see Section C of the Supplementary Material). Validating the model's cross-lingual performance is more difficult, given the lack of a labeled dataset of populist sentences in languages other than English and is left for future work. However, because the model is trained on text in all 22 languages, it does not need to do the challenging task of classifying languages outside the training set.

Examining the scored sentences by hand provides some further validity for the method. Of all manifesto sentences from the United Kingdom Independence Party, a populist party, the model identifies the following three sentences as having the highest populism scores:

- "Politics is corrupted by self-interest and big business."
- "These professional politicians don't want us to run our own country or control our own lives."
- "An unaccountable elite revels in mutual back-scratching and cronyism."

Here, the model successfully identifies elements of populist rhetoric, such as anti-elitism and the portrayal of a corrupt political class.

This validation demonstrates the potential of using synthetic data to train a multilingual sentence-level populism classifier without any hand-labeled examples. The classifier's performance on both known manifesto labels and hand-annotated samples, along with its ability to identify highly populist sentences, suggests that sythentic text is a viable source of training data for applied researchers.

3.3. Validation 3: Generating Synthetic Tweets with Adapted Language Models to Identifying Weapons in the Ukraine War

A major component of U.S. and European policy responses to the invasion of Ukraine centers on the provision and use of specific weapons systems in Ukraine and the escalation risks associated with providing advanced weapons. Open source intelligence from social media data provides granular data on where and how the Ukrainian military is using specific weapons. This validation shows that the accuracy of a named entity recognition (NER) model trained to identify specific weapons systems on hand-labeled synthetic text can match the accuracy of a model trained on real tweets. However, it highlights a limitation of synthetic data: a marginal hand-annotated synthetic tweet is less useful for training a supervised model than an annotated real tweet. This gap in performance is lessened by tuning the generation parameters to produce more realistic text, but suggests that hand annotation of synthetic text should only be used in limited circumstances, such as when real training text is unavailable, is too sensitive to show to annotators, or when reproducibility of a specific methodological approach is a top priority.

I collect a set of around 20,000 real tweets from four Twitter accounts that report detailed information on the fighting in Ukraine from the first two months after the invasion. Because the synthetic tweets should closely match the actual tweets, I opt for an adaptation approach to text generation. I use GPT-2 (large) because it can be adapted and run with standard hardware, and performs well on the task of generating tweets. Further details on data and training in Section D.2.1 of the Supplementary Material.

To improve the quality of the synthetic tweets, I apply the adversarial method introduced in Section 2.3. Across 56 combinations of hyperparameters, the classification accuracy of a BERT-based classifier ranges from 0.99 to 0.67, indicating a large effect of hyperparameters of tweet generation (see Figure SI 6 in the Supplementary Material). An ideal set of synthetic tweets would be indistinguishable from real tweets: the fact that they are distinguishable indicates that a marginal synthetic tweet is less useful for training a supervised model than an annotated real tweet.

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

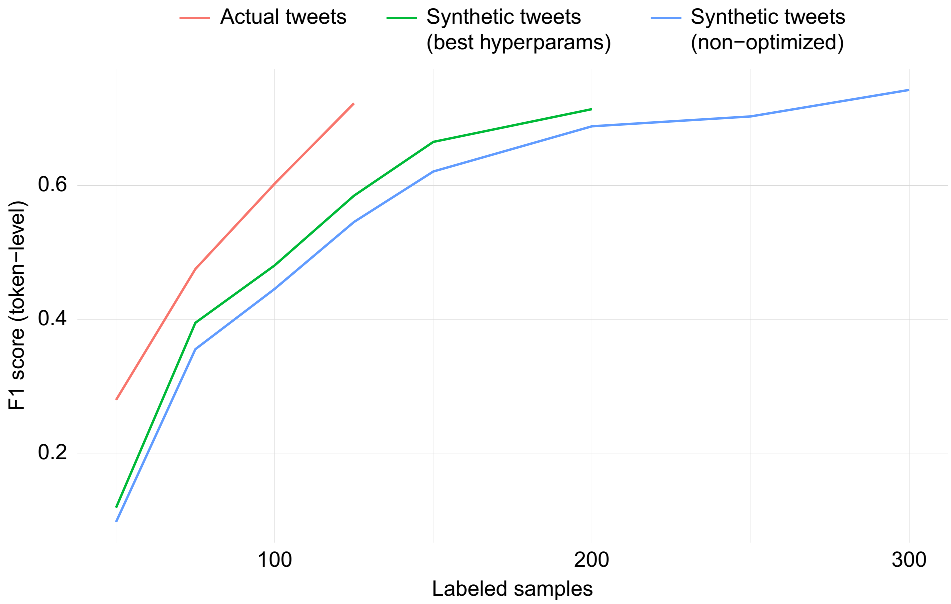

Figure 3. Test set performance of a named entity recognition model detecting a WEAPON class, trained on annotated actual tweets and annotated synthetic tweets.

3.3.1. Comparing Performance of Real and Synthetic Tweets

How well do these synthetic tweets work in practice to train a supervised model? I hand annotate 1,600 tweets with span-level labels on the specific, named weapons systems described in the tweets (see Section D of the Supplementary Material). The annotated set includes 200 real tweets, 600 synthetic tweets generated from non-optimized parameters, 600 tweets generated using the parameters selected by the adversarial method, with an additional 200 real tweets as evaluation data. I train an NER model to identify mentions of specific, named weapons in the text (e.g., "Bayraktar TB2").[4] Figure 3 reports the accuracy for the NER model trained on actual and synthetic tweets at different training set sizes and evaluated on labeled actual tweets. A marginal labeled non-optimized synthetic tweet is much less valuable than a real labeled tweet: the model requires 300 labeled non-optimized synthetic training examples to reach the performance it can attain with 200 labeled real tweets. Applying the adversarial technique to select the optimal generation hyperparameters reduces the gap: only 200 synthetic tweets are required to reach the same performance as the real tweet model.

4. Conclusion

As political scientists increasingly turn to large language models as useful tools for analyzing text, they face decisions about how to use these models in a transparent, reproducible, and accurate way. This paper shows that using LLMs for their original strength—generating text—can help researchers overcome the challenges of labeling text for supervised text analysis. Training a smaller model on synthetic text allows researchers to use the full capabilities of a large language model to generate synthetic text, but then use a smaller, simpler, reproducible model to make predictions. The small model is faster and cheaper to run than the large language model, its hyperparameters can be cheaply tuned, it can be used in the future to classify new documents, and can be shared with other researchers.

[4] I use spaCy's small `en_core_web_sm` model as a base and the default training values set by Prodigy (Montani and Honnibal 2018).

https://doi.org/10.1017/pan.2024.31 Published online by Cambridge University Press

By controlling the content and style of synthetic text through prompting, adaptation, and sampling, researchers can generate synthetic text that represents the domain of text they are interested in. This synthetic text can be used to augment hand-labeled data or used directly as a form of model distillation. In limited circumstances where privacy or reproducibility concerns necessitate it, researchers may wish hand-label synthetic text, although this approach carries a cost in accuracy.

The size and capabilities of LLMs are growing at a rapid rate, and using them to generate synthetic training text to fit small, tailored models will remain relevant. Indeed, using synthetic text to train smaller LLMs is driving many of the improvements in efficient LLMs (Gunasekar *et al.* 2023). Even as LLMs improve, controlling their generation will still operate either by modifying the conditioning sequence of text ($w_{i-1} \dots w_1$), modifying the language model's parameters (θ), or changing how words are sampled from the probability distribution (γ), and researchers will still face the same decisions about how to use it as training data.

Acknowledgments. Thank you to Benjamin Bagozzi, Jill Irvine, Michael Jankowski, Adam Lauretig, Aidan Milliff, Erin Rossiter, Brandon Stewart, and the anonymous reviewers for helpful comments. This work was first presented at PolMeth 2022 and New Directions in Text as Data 2022. MIT's Lincoln Laboratory provided computing resources (Reuther *et al.* 2018).

Funding. This work was supported by the Political Instability Task Force (PITF). The PITF is funded by the Central Intelligence Agency. The views expressed in this paper are the author's alone and do not represent the views of the US Government.

Data Availability Statement. All code and uncopyrighted text used in this paper is available in the Harvard Dataverse at https://doi.org/10.7910/DVN/JJ5BBX (Halterman 2024).

Supplementary Material. For supplementary material accompanying this paper, please visit https://doi.org/10.1017/pan.2024.31.

References

Argyle, L. P., E. C. Busby, N. Fulda, J. R. Gubler, C. Rytting, and D. Wingate. 2023. "Out of One, Many: Using Language Models to Simulate Human Samples." *Political Analysis* 31 (3): 337–351.

Atreja, S., J. Ashkinaze, L. Li, J. Mendelsohn, and L. Hemphill. 2024. "Prompt Design Matters for Computational Social Science Tasks but in Unpredictable Ways." arXiv:2406.11980v1, Preprint.

Caliskan, A., J. J. Bryson, and A. Narayanan. 2017. "Semantics Derived Automatically from Language Corpora Contain Human-Like Biases." *Science* 356 (6334): 183–186.

Dai, Y., and A. Kustov. 2022. "When Do Politicians Use Populist Rhetoric? Populism as a Campaign Gamble." *Political Communication*, 39 (3): 1–22. https://doi.org/10.1080/10584609.2022.2025505.

Di Cocco, J., and B. Monechi. 2021. "How Populist Are Parties? Measuring Degrees of Populism in Party Manifestos Using Supervised Machine Learning." *Political Analysis*, 30 (3): 1–17.

Edunov, S., M. Ott, M. Auli, and D. Grangier. 2018. "Understanding Back-Translation at Scale." In *EMNLP*, 489–500.

Egami, N., M. Hinck, B. Stewart, and H. Wei. 2024. "Using Imperfect Surrogates for Downstream Inference: Design-Based Supervised Learning for Social Science Applications of Large Language Models." In *NeurIPS* 36.

Fong, C., and M. Tyler. 2020. "Machine Learning Predictions as Regression Covariates." *Political Analysis*, 29 (4): 1–18.

Fu, Z., W. Lam, A. M.-C. So, and B. Shi. 2021. "A Theoretical Analysis of the Repetition Problem in Text Generation." *Proceedings of the AAAI Conference on Artificial Intelligence*, 35: 12848–12856, 14.

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

Gunasekar, S. et al. 2023. "Textbooks Are All You Need." arXiv:2306.11644, Preprint.

Halterman, A. 2024. "Replication Data for: Synthetically Generated Text for Supervised Text Analysis." Harvard Dataverse. https://doi.org/10.7910/DVN/JJ5BBX10.7910/DVN/JJ5BBX.

Halterman, A., and K. A. Keith. 2024. "Codebook LLMs: Adapting Political Science Codebooks for LLM Use and Adapting LLMs to Follow Codebooks." arXiv:2407.10747, Preprint.

Halterman, A., K. A. Keith, S. M. Sarwar, and B. O'Connor. 2021. "Corpus-Level Evaluation for Event QA: The IndiaPoliceEvents Corpus Covering the 2002 Gujarat Violence." In *Findings of the Association for Computational Linguistics*.

He, X., I. Nassar, J. Kiros, G. Haffari, and M. Norouzi. 2022. "Generate, Annotate, and Learn: NLP with Synthetic Text." *TACL* 10: 826–842.

Ippolito, D., D. Duckworth, and D. Eck. 2020. "Automatic Detection of Generated Text Is Easiest When Humans Are Fooled." In *ACL*, 1808–1822.

Jankowski, M., and R. A. Huber. 2023. “When Correlation Is Not Enough: Validating Populism Scores from Supervised Machine-Learning Models.” *Political Analysis*, 31 (4): 1–15. https://doi.org/10.1017/pan.2022.32.

Langlais, P.-C. 2024. “Releasing Common Corpus: The Largest Public Domain Dataset for Training LLMs.” https://Huggingface.co/Blog/Pclanglais/Common-Corpus.

Longpre, S., et al. 2023. “The Flan Collection: Designing Data and Methods for Effective Instruction Tuning.” In *ICML*, 22631–22648.

Meng, Y., J. Huang, Y. Zhang, and J. Han. 2022. “Generating Training Data with Language Models: Towards Zero-Shot Language Understanding.” In *NeurIPS* 35: 462–477.

Montani, I., and M. Honnibal. 2018. “Prodigy: A New Annotation Tool for Radically Efficient Machine Teaching.” Online article: https://explosion.ai/blog/prodigy-annotation-tool-active-learning. Last accessed: August 19, 2024.

Mudde, C. 2004. “The Populist Zeitgeist.” *Government and Opposition* 39 (4): 542–563.

Ornstein, J. T., E. N Blasingame, and J. S. Truscott. 2022. “How to Train Your Stochastic Parrot: Deep Language Models for Political Texts.” *PolMeth Conference Paper*.

Peskoff, D., and B. M. Stewart. 2023. “Credible Without Credit: Domain Experts Assess Generative Language Models.” In *ACL (Short Papers)*, 427–438.

Platen, P. von. 2020. “How to Generate Text: Using Different Decoding Methods for Language Generation with Transformers.” Hugging Face Blog.

Reimers, N., and I. Gurevych. 2019. “Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks.” In *EMNLP*.

Reimers, N., and I. Gurevych. 2020. “Making Monolingual Sentence Embeddings Multilingual Using Knowledge Distillation.” In *EMNLP*.

Reuther, A., et al. 2018. “Interactive Supercomputing on 40,000 Cores for Machine Learning and Data Analysis.” In *HPEC*, 1–6. IEEE.

Rytting, C. M., et al. 2023 “Towards Coding Social Science Datasets with Language Models.” arXiv:2306.02177, Preprint.

Schick, T., and H. Schütze. 2021. “Generating Datasets with Pretrained Language Models.” In *Proceedings of EMNLP*, 6943–6951.

Spirling, A. 2023. “Why Open-Source Generative AI Models Are an Ethical Way Forward for Science.” *Nature* 616 (7957): 413.

Subramanian, K. S. 2007. *Political Violence and the Police in India*. New Delhi: SAGE Publications India.

Taori, R., et al. 2023. “Alpaca: A Strong, Replicable Instruction-Following Model.” *Stanford Center for Research on Foundation Models* 3 (6): 7.

Tunstall, L., et al. 2022 “Efficient Few-Shot Learning Without Prompts.” In *NeurIPS*.

Volkens, A. et al. 2021 *The Manifesto Data Collection. Manifesto Project (MRG/CMP/MARPOR, Version 2021a*. Berlin: Wissenschaftszentrum Berlin für Sozialforschung (WZB). https://doi.org/10.25522/manifesto.mpds.2021a.

Wang, Y., et al. 2023 “Self-Instruct: Aligning Language Models with Self-Generated Instructions.” *ACL (Long Papers)*, 13484–13508.

Wei, J., and K. Zou. 2019. “EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks.” In *EMNLP-IJCNLP*.

Wilkinson, S. I. 2006. *Votes and Violence: Electoral Competition and Ethnic Riots in India*. Cambridge: Cambridge University Press.

Wu, P. Y., J. A. Tucker, J. Nagler, and S. Messing. 2023. “Large Language Models Can Be Used to Estimate the Ideologies of Politicians in a Zero-Shot Learning Setting.” arXiv:2303.12057, Preprint.

Yu, Y. et al. 2024 “Large Language Model as Attributed Training Data Generator: A Tale of Diversity and Bias.” In *NeurIPS* 36.

Zellers, R. et al. 2019 “Defending Against Neural Fake News.” *NeurIPS* 32.

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. (2025). Synthetically generated text for supervised text analysis. *Political Analysis*, 1–14. https://doi.org/10.1017/pan.2024.31