Text To Data: Adapting Large Language Models for Data Extraction

Joseph T. Ornstein*
University of Georgia

August 29, 2023

Abstract

Large language models (LLMs) can be adapted to parse unstructured text into structured datasets, but it remains an open question whether this approach can outperform human coders in terms of cost and reliability. In this letter, I present a pre-analysis plan to validate an LLM approach to data extraction, and provide an open-source software package for researchers.

Keywords: text as data, large language models

*Thanks to Amanda Heideman, Bryant Moy, Alexander Sahn, and Kaylyn Jackson Schiff for helpful comments on earlier drafts.

1 Introduction

The purpose of this paper is to introduce and validate a new approach to extracting data from unstructured text. I take advantage of recent advances in large language models (LLMs) – next-word prediction models trained on massive corpus of text from the Internet (Ornstein et al. 2022) – by constructing prompts that express the data extraction task as a next-word prediction task. The result is a general-purpose tool that can be applied to a wide array of unstructured texts. In a set of pre-registered analyses, I plan to test whether this approach performs as well as non-expert human coders at a fraction of the time and cost.

To date, there are three approaches that social scientists have taken to convert text to structured datasets. The first is human hand-coding. Although considered the most reliable, it is costly, time-consuming, and frequently inconsistent across multiple coders. Automated computational approaches, by contrast, are inexpensive and fast, but have traditionally been restricted to applications where the text is *structured* in some known fashion. For example, the leading sentences of newspaper articles tend to contain complete event summaries, making automated events coding in international relations much more feasible (King & Lowe 2003). Or a researcher may focus on coding documents from a single political jurisdiction where the formatting of public records is known and consistent over time (Sahn 2022).

But converting *unstructured* text to data, where each document may have a completely different format, has long been a task that only human coders are capable of reliably performing. Supervised machine learning approaches, though promising, tend to require large sets of hand-labeled data to train the models (Harrison et al. 2023), and the feature space that such models use to represent texts is a poor substitute for the contextual understanding that human coders bring to reading a text. Systematic measurement error from such proxies can bias subsequent causal analyses (Knox et al. 2022).

The development of large language models (LLMs) creates an opportunity for improvement. Instead of creating a bespoke approach to data extraction for each type of document, researchers can take a general-purpose "foundation model" (Bommasani 2021) and adapt it

to their particular task. This approach takes advantage of the fact that LLMs are trained on far more data – and encode a far more nuanced understanding of human language – than any model that an individual social scientist could train.

For an example of how this approach works in practice, consider the following passage from a Planning Commission meeting in Acton, Massachusetts, held on January 19, 2017.

> Jeff Barry, of 21 Martin Street, was concerned that the applicant did not look closely at the groundwater and feared that the new houses in the development may be subject to flooding.
>
> Anne Forbes, of 25 Martin Street, asked that the north boundary houses have a landscape privacy screen, asked for the applicant to confirm that the setbacks complied, wants concrete reason why the third building couldn't be saved, and mentioned that there is already a Haley Lane in Acton.
>
> Anne Cochrane, of 13 Martin Street, expressed concerns about the increased traffic on an already narrow street and wants to see sidewalks for the development.
>
> David Honn, of the Acton Design Review Board, expressed concerns on who would be moving the houses. He wanted to know why the applicant could not save the third house, and he mentioned the applicant needing to submit reports to the state.

Figure 1: Excerpt from Acton Planning Commission meeting minutes, January 19, 2017

Einstein et al. (2019) laboriously hand-coded published meeting minutes like this from 813 meetings of local planning and zoning boards in 97 Massachusetts towns. The purpose of the research was to determine who participates in local politics, and how participatory inequalities are likely to drive housing and land use policy choices. Unfortunately, the labor-intensive nature of the work meant that the authors had to restrict the scope of their research to a relatively small geographic area (suburbs around Boston). An automated approach could significantly increase the scope and generalizability of this type of research.

If we are interested in documenting the names of commenters, their position on the proposed projects, and a summary of the reasons given for their position, we can structure a prompt as follows, which expresses the problem as a next-word prediction task (prompt in black, GPT-3 output in green).

Create a table summarizing the public comments made at the Acton Planning Board meeting.

Jeff Barry, of 21 Martin Street, was concerned that the applicant did not look closely at the groundwater and feared that the new houses in the development may be subject to flooding.

Anne Forbes, of 25 Martin Street, asked that the north boundary houses have a landscape privacy screen, asked for the applicant to confirm that the setbacks complied, wants concrete reason why the third building couldn't be saved, and mentioned that there is already a Haley Lane in Acton.

Anne Cochrane, of 13 Martin Street, expressed concerns about the increased traffic on an already narrow street and wants to see sidewalks for the development.

David Honn, of the Acton Design Review Board, expressed concerns on who would be moving the houses. He wanted to know why the applicant could not save the third house, and he mentioned the applicant needing to submit reports to the state.

Name of Commenter	Position (Support, Neutral, or Oppose)	Reasons, separated by semicolons
Jeff Barry	Oppose	Groundwater not looked at closely; fear of flooding
Anne Forbes	Neutral	North boundary houses need landscape privacy screen; confirm setbacks comply; concrete reason why third building couldn't be saved; already a Haley Lane in Acton
Anne Cochrane	Oppose	Increased traffic on narrow street; need sidewalks for development
David Honn	Neutral	Who will move the houses; why third house couldn't be saved; need to submit reports to state

Figure 2: GPT-3 prompt for extracting data from meeting minutes

The result is a data table structured according to the researcher's specifications, which can be combined with data tables produced from other documents and used as an input to a statistical analysis. The process of converting hundreds of PDFs into structured data, which would otherwise take hours of work for human coders, can be accomplished in seconds using this approach.

In this letter, I present a some preliminary results validating this approach to extracting data from unstructured text. In particular, I will focus on applications in the subfield of American local politics, where there is tremendous variation in the formatting and availability of public records, and the process of hand-coding datasets has limited the scope of what research projects are feasible. Throughout, I describe best practices for validating outputs and minimizing measurement error. All the methods described in the paper will be made available through an open-source software package.[1]

[1]The `text2data` package in `R`. See the Appendix for installation instructions and vignettes.

2 Methods

Any automated method for extracting data from unstructured text will generally require two steps (illustrated in Figure 3), both of which are computationally demanding. The first is to convert the document into plain text. For non-digitized documents that are available only as images, this requires some form of Optical Character Recognition. The next step is to take the unstructured plain text and convert it to a structured dataset with the desired information. Fortunately, LLMs can aid in both steps of the process.

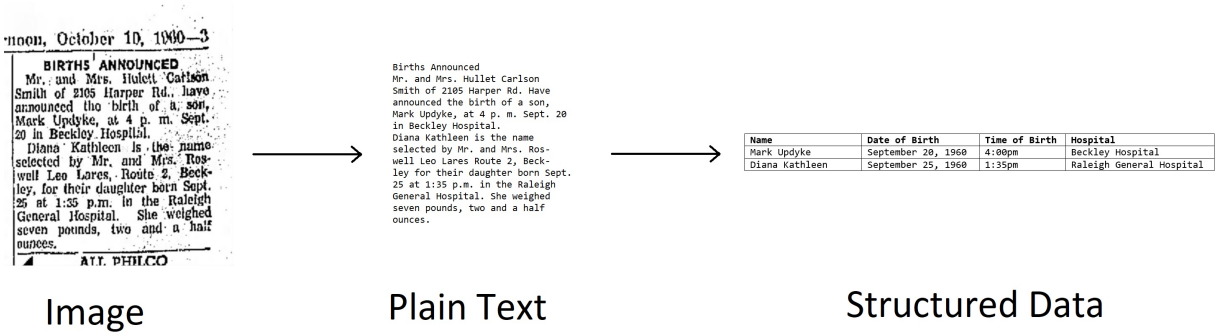
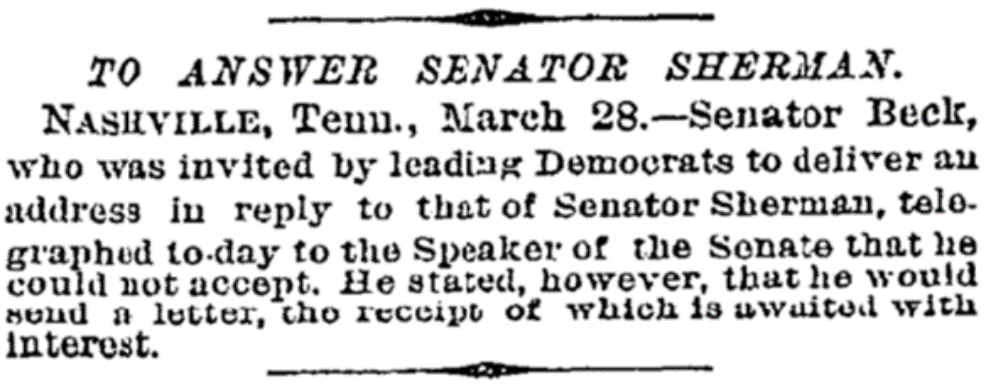


Figure 3: Automated data extraction workflow

2.1 Optical Character Recognition

Social scientists often rely on texts from archival or other non-digitized sources. The process of converting these primary sources to text is often quite difficult; human transcriptionists are costly, and optical character recognition (OCR) can be error-prone when text is low-resolution or hand-written. As an example, consider the image in Figure 4 of a 1887 article from *The New York Times*.



TO ANSWER SENATOR SHERMAN.

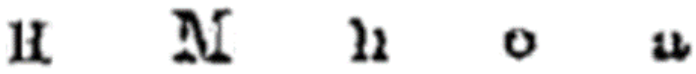
NASHVILLE, Tenn., March 28.—Senator Beck, who was invited by leading Democrats to deliver an address in reply to that of Senator Sherman, telegraphed to-day to the Speaker of the Senate that he could not accept. He stated, however, that he would send a letter, the receipt of which is awaited with interest.

Figure 4: Article from the front page of the New York Times (March 29, 1887).

Though a human can easily read and interpret this text, converting the image to plain text is a non-trivial problem in the field of computer vision. Note that the text is slightly tilted, and the penultimate line is cut off at the top. A standard OCR algorithm[2] yields the following output:

```
                20 ANSWER SENATOR SHERMAN.
      NasuviLte, Tenn., Narch 28.-Senator Beck,
  who was invited by leading Democrats to deliver au
  address in reply to that of Senator Sherman, tolo-
  graphed to-day to the Speaker of the Senate that ne
  could not accept. He stared, however, thut he would
  send a lutter, tho receipt of which fo uwaited with
                      Interest.
```

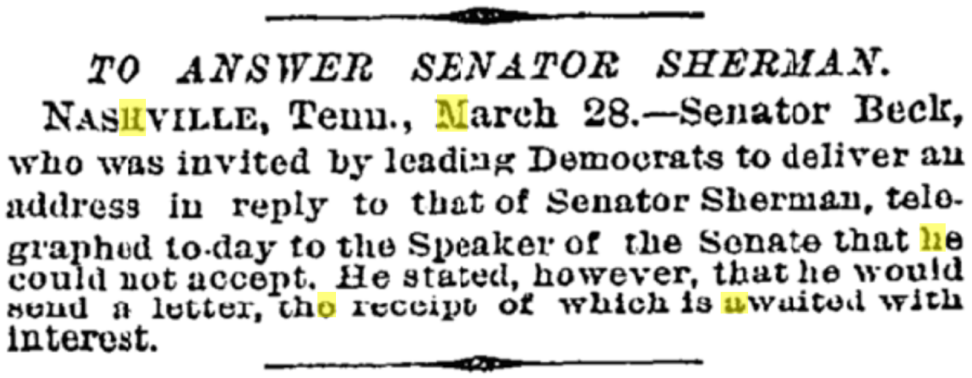
Roughly one out of every five words in the article are transcribed incorrectly. To see why there are so many errors, consider the following images of five letters from the article:



When isolated from their context, the typeface of each letter could easily be mistaken for another letter. But humans do not read letters in isolation. Instead, fluent readers learn to recognize letters in parallel, using their knowledge of the language to "predict" what the constituent letters of a word must be (e.g. McClelland & Johnston 1977). Figure 5 highlights the letters in their context, where it is straightforward to see that (for instance)

[2]The `tesseract` package in `R`.

the first letter of "March" is an "M"; a human reader would correctly judge the phrase "Narch 28" to be unlikely.



TO ANSWER SENATOR SHERMAN.

NASHVILLE, Tenn., March 28.—Senator Beck, who was invited by leading Democrats to deliver an address in reply to that of Senator Sherman, telegraphed to-day to the Speaker of the Senate that he could not accept. He stated, however, that he would send a letter, the receipt of which is awaited with interest.

Figure 5: New York Times article with ambiguous letters highlighted

To achieve state-of-the-art performance when digitizing documents, scholars often rely on costly proprietary software (e.g. the ABBYY FineReader, as in Hopkins et al. (2022)). Though cheaper than human transcriptionists, this is a non-negligible research cost. Large language models suggest an alternative, less costly, method for achieving the same level of performance. Just as humans interpret ambiguous sequences of text based on what would be the most likely next word in a sequence, we can make use of the fact that large language models are adept at predicting the next word in a sequence to perform a similar task. Figure 6 presents an example of a prompt that adapts GPT-3 to this purpose (black is the prompt, green highlighted text is the output).

Correct the OCR errors (removing characters denoting carriage returns).

NasuviLte, Tenn., Narch 28.—Senator Beck,\nwho was invited by leading Democrats to deliver au\naddress in reply to that of Senator Sherman, tolo-\ngraphed to-day to the Speaker of the Senate that ne\ncould not accept. He stared, however, thut he would\nsend a lutter, tho receipt of which fo uwaited with\nInterest.\n\na\n

---

Nashville, Tenn., March 28.—Senator Beck, who was invited by leading Democrats to deliver an address in reply to that of Senator Sherman, telegraphed to-day to the Speaker of the Senate that he could not accept. He stated, however, that he would send a letter, the receipt of which is awaited with interest.

Figure 6: GPT-3 prompt to clean up OCR errors

Every transcription error in the body of the article is flawlessly corrected. When combined

with image pre-processing tools and modern methods for image analysis, like convolutional neural networks (Torres & Cantú 2021), this approach could significantly improve the accuracy of automated image-to-text conversions. Post-processing OCR transcriptions using large language models has been proposed elsewhere by computer scientists (Li et al. 2022, Ströbel et al. 2022), and, though promising, the approach has yet to be fully explored or validated in a social science context.

2.2 Parsing Data

Once a document has been converted to plain text, we require a method that can convert the unstructured text into structured data. Fortunately, one can adapt LLMs to this task as well – as in the example from the introduction – by expressing the data extraction task as a next-word prediction task.

The challenge in this stage of the workflow is prompt design. For any given data extraction task, there are numerous ways that one could structure a prompt, and seemingly innocuous differences in wording and formatting can affect an LLM's responses (Zhao et al. n.d.). Because these models are black boxes with billions of parameters, it is essential that researchers validate their choice of prompt by comparing the output against a hand-coded "gold standard" dataset. Only if the LLM can reproduce the results from this validation set should it be applied to the rest of the text corpus.

3 Application: Neighborhood Defenders

What follows are some preliminary results from a subset of the Einstein et al. (2019) meeting minutes. To validate the prompt design described in the introduction, I retrieved all the meeting minutes in the corpus from the town of Acton MA, converted the PDFs to plain text using OCR, and extracted data using a ChatGPT prompt with the following format:

```
Create a data table summarizing the public comments made by
citizens in the following meeting minutes.  Only include in
the table comments about residential construction projects that
create at least one new unit of housing.  Omit comments by members
of the Board, developers, or the applicants themselves.  Include
comments by attorneys representing members of the public.  Leave
```

```
missing fields blank.

---

{TEXT OF MINUTES}

---

| --- | --- | --- | --- | --- | --- |

| Date of Comment (yyyy-mm-dd) | Name of Commenter | Most Likely
Gender of Commenter | Street Address of Commenter (if available)
| Position on Construction Project (Support, Neutral, or Oppose)
| Reasons for Position, separated by semicolons |

| --- | --- | --- | --- | --- | --- |
```

This is a deceptively difficult task for a general-purpose model. To perform adequately, the language model must be able to determine which parts of the text reflect comments by members of the public (omitting comments by members of the Board, developers, or applicants), deduce their positions about the proposed project based on their comments, and summarize the reasons for their position. It must also keep track of the date the meeting was held (reported in a specific format), and infer the gender of the commenters based on their name. This would be a daunting task for a human coder, much less for a model trained on next-word prediction.

And yet the preliminary results from this validation exercise are quite promising. Across 18 pages of documents, Einstein et al.'s human coders identified 22 distinct comments made by members of the public in reference to new residential development – 14 opposed, 6 neutral, and 2 in support. By comparison, ChatGPT identified 34 public comments – 18 opposed, 13 neutral, and 3 in support. The analysis took less than a minute and cost 6 cents for the use of the OpenAI API.

Conducting a careful audit of these two datasets reveals errors made by both human coders and the LLM. The mistakes made by human coders are largely errors of omission. There are 10 comments identified by ChatGPT that are missing from the original human-coded dataset — nearly one-third of all the public comments in these meeting minutes. The human coders also make several transcription errors, incorrectly recording dates for 5 out of 22 comments.

By comparison, ChatGPT's mistakes tended to be errors of commission. The resulting dataset omitted only one comment identified by human coders, but included three comments that should not have been included. One was a comment made by a member of the Acton Planning Board, and two of the public commenters had duplicate entries (likely because they spoke multiple times during the meeting, and my instructions did not explicitly ask to consolidate multiple comments by the same speaker into one row). As the following table reports, ChatGPT and the human coders agreed on the stance of the comments in 76% of cases. In the five cases where the methods disagreed, the human coders are obviously correct in one, obviously incorrect in another, and the remaining three cases are ambiguous.

Table 1: Human-coded stance vs. ChatGPT-coded stance for 21 comments that both methods identified

Human Coder / ChatGPT	Neutral	Oppose	Support
Neutral	4	3	0
Oppose	1	11	1
Support	0	0	1

In future work, I plan to further refine the prompt, ensuring that it passes validation tests on multiple random documents before applying it to the larger meeting minutes corpus. To assess the quality of the data produced, I plan to use a two-stage validation process. For the "position" column, I will compare the labels assigned by the model with the labels assigned by human coders. When the two labels differ, I will ask a second set of crowd-coders (undergraduate research assistants and/or online crowd workers) to judge whether the human-coded label or LLM-coded label is a better reflection of the text. Validating the "reasons" column is somewhat more difficult, because different human coders are likely to summarize the text in somewhat different ways. For this column, I plan to ask human crowd-coders, blinded to which is the human and which is the GPT-3 column, to report which set of reasons better summarizes the text (or if they are both equally good).

4 Discussion

The capabilities of large language models exceed what many thought was possible just five years ago. Using the approach described in this letter, a researcher can extract data from roughly 100 pages of text (200,000 characters) in a few minutes for less than 15 cents. This is a tool worth exploring and understanding. Yet caution is warranted before adopting these methods, particularly the sorts of closed-source, proprietary models that currently dominate the field (Spirling 2023). Any new application of LLMs requires carefully assessing the resulting measures for reliability and validity – using the same best practices one would use for non-expert human coders.

Another limitation of currently available LLMs is the amount of text that they can process at one time. As of August 2023, the most capable variant of ChatGPT has a "context window" of 16,000 tokens (roughly four characters per token), the maximum size of input and output that the model can process at once. Longer texts (e.g. 50 page PDFs from interminable New England planning commission meetings) cannot be processed in this way without being split into multiple documents, which impedes performance (e.g. ChatGPT won't remember who is a Planning Commission member and who is a public commenter if that information is at the beginning of a document that was split in two).

Based on the preliminary results in this paper, my tentative advice for researchers is to use LLMs as a supplement, rather than a replacement, for human coding, to identify items in a large text corpus that inattentive coders may have missed. This recommendation is in line with other research on combining human-labeled and surrogate measures using ensemble methods (Ornstein et al. 2022) or design-based estimators (Egami et al. n.d.). How best to combine judgments from humans and LLMs is likely to be an active area of research in the coming years.

Appendix A: The `text2data` package

The package is currently available as a GitHub repository. To install, first make sure you have the `devtools` package available, then you can install the package with the following line of code:

```
devtools::install_github('joeornstein/text2data')
```

You will also want to make sure you have the `reticulate` package installed, which allows R to talk to Python.

```
install.packages('reticulate')


reticulate::install_miniconda()
```

Once `reticulate` is installed, you can install OpenAI's Python module, which will allow you to submit prompts to the GPT-3 API. I've included a convenience function in the package to handle this installation:

```
library(text2data)


setup_openai()
```

Next, you will need an account with OpenAI. You can sign up here, after which you'll need generate an API key here. I recommend adding this API key as a variable in your environment called `OPENAI_API_KEY`; that way you won't risk leaking it by hard-coding it into your R scripts. The `text2data` package will automatically look for your API key under that variable name, and will prompt you to enter the API key manually if it can't find one there. If you're unfamiliar with setting Environment Variables, here are some helpful instructions for various operating systems. Note that you may need to restart your computer after completing this step (I did).

Completing Prompts

The workhorse function of the `text2data` package is `complete_prompt()`. This function submits a sequence of words (the prompt) to the OpenAI API, and returns a dataframe with the five highest-probability next word predictions and their associated probabilities.[3]

```
library(text2data)


complete_prompt(prompt = 'My favorite food is')
```

```
  response       prob
1    pizza 0.08614531
2        a 0.03913327
3    sushi 0.03705063
4  chicken 0.01834140
5    pasta 0.01819866
```

If you prefer the model to auto-regressively generate sequences of text instead of outputting the next-word probabilities, set the `max_tokens` argument greater than 1. The function will return a character object with the most likely completion at each point in the sequence.

```
complete_prompt(prompt = 'My favorite food is',
                max_tokens = 6)
```

```
[1] " pizza. I love pizza."
```

For more on using this function, including how to adapt it for classification problems, see the tutorial here. The following two functions are wrappers around the `complete_prompt()` function that allow you to clean up OCR output and extract data from unstructured text.

[3]Note that the default model used by `complete_prompt()` is "davinci-002", the 175-billion parameter base GPT-3 model. You can prompt different model variants using the `model` argument.

Clean Up OCR

The `clean_ocr()` function submits a prompt that produces a cleaned up version of garbled text.

```
clean_ocr('John wenf to the sfore. Rosie ran ercands at the mal.')
```

```
[1] "John went to the store. Rosie ran errands at the mall."
```

Parse Text

The `parse_text()` function formats and submits a data extraction prompt like the ones in the paper. It takes three inputs:

- `text`: the text
- `instructions`: a set of instructions that will be appended to the beginning of the prompt, formatted like instructions you would provide to a human research assistant
- `col_names`: a list of column names for the desired dataframe; they do not have to follow standard naming conventions for R columns, and can be as detailed as you like

Here is an example of this function simultaneously cleaning up and parsing a garbled text string from OCR.

```
parse_text(
  text = "John wenf to the sfore. Rosie ran ercands at the mal.",
  instructions = "Create a dataset from the following passage with the names
  of each character and the place they went.",
  col_names = c("Name of Character", "Where They Went"))
```

```
# A tibble: 2 x 2
  `Name of Character` `Where They Went`
  <chr>              <chr>
1 John               store
2 Rosie              mall
```

References

Bommasani, R. e. a. (2021), 'On the opportunities and risks of foundation models', *arXiv:2108.07258 [cs]* . arXiv: 2108.07258.

Egami, N., Jacobs-Harukawa, M., Stewart, B. M. & Wei, H. (n.d.), 'Using large language model annotations for valid downstream statistical inference in social science: Design-based semi-supervised learning'.

Einstein, K. L., Palmer, M. & Glick, D. M. (2019), 'Who participates in local government? evidence from meeting minutes', *Perspectives on Politics* **17**(1), 28–46.

Harrison, J. S., Josefy, M. A., Kalm, M. & Krause, R. (2023), 'Using supervised machine learning to scale human-coded data: A method and dataset in the board leadership context', *Strategic Management Journal* .

Hopkins, D. J., Schickler, E. & Azizi, D. L. (2022), 'From many divides, one? the polarization and nationalization of american state party platforms, 1918–2017', *Studies in American Political Development* **36**(1), 1–20. Publisher: Cambridge University Press.

King, G. & Lowe, W. (2003), 'An automated information extraction tool for international conflict data with performance as good as human coders: A rare events evaluation design', *International Organization* **57**(3), 617–642.

Knox, D., Lucas, C. & Cho, W. K. T. (2022), 'Testing causal theories with learned proxies', *Annual Review of Political Science* **25**, 419–441.

Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z. & Wei, F. (2022), 'Trocr: Transformer-based optical character recognition with pre-trained models', *arXiv* .

McClelland, J. L. & Johnston, J. C. (1977), 'The role of familiar units in perception of words and nonwords', *Perception & Psychophysics* **22**(3), 249–261.

Ornstein, J. T., Blasingame, E. N. & Truscott, J. S. (2022), 'How to train your stochastic parrot: Large language models for political texts'.

Sahn, A. (2022), 'Public comment and public policy', *Working Paper* .

Spirling, A. (2023), 'Why open-source generative ai models are an ethical way forward for science', *Nature* **616**(7957), 413–413.

Ströbel, P. B., Clematide, S., Volk, M. & Hodel, T. (2022), 'Transformer-based htr for historical documents', *arXiv* .

Torres, M. & Cantú, F. (2021), 'Learning to see: Convolutional neural networks for the analysis of social science data', *Political Analysis* pp. 1–19.

Zhao, T. Z., Wallace, E., Feng, S., Klein, D. & Singh, S. (n.d.), 'Calibrate before use: Improving few-shot performance of language models'.