Appendix

Prepared for "Positioning Political Texts with Large Language Models by Asking and Averaging" by Gaël Le Mens and Aina Gallego

Appendix 1. Data

Appendix 1.1 Tweets published by members of the US Congress after the training cut-off date of GPT-4 – obtaining crowdsourced position estimates

In November 2023, 597 Prolific participants (US residents, average age 40 years, 52% male, 46% female, 2% others) each rated 30 tweets. We used the same 30 sets of 30 tweets, composed of 15 tweets from each party as in Le Mens et al. (2023a). Participants were randomly assigned to one of these sets. We obtained between 19 and 21 position ratings for each tweet, with an average of 20 ratings per tweet. The crowdsourced position estimate of a tweet is the average of these ratings. All tweets, except one, received at least one position rating (different from 'NA'), leading to a test data set of 899 tweets and their crowdsourced position estimates.

To provide their position ratings, participants responded to the question: "Where does this text stand on the 'left' to 'right' wing scale? If the text does not have political content, select 'Not Applicable'." Participants were not given any instructions as to what we mean by 'left' or 'right.' They reported their scores using a slider with endings 'Extremely Left' (left side, coded 0) to 'Extremely right' (right side, coded 100). The slider was centered when the page appeared on the screen. Participants could use a checkbox to respond 'Not Applicable'.

Importantly, participants were not provided with any information about the author of the tweet — just the text of the tweet. Therefore, they were not informed whether the tweet was written by a Democrat or a Republican.

To ensure response quality, we screened Prolific participants based on the following criteria:

- Fluent languages include English;
- Approval Rate: Minimum 95%;
- Country of Residence: US;
- Political Spectrum (US): they should have provided a response to this question (the response could be anyone of {Conservative, Moderate, Liberal, Other}),
- U.S. Political Affiliation: they should have provided a response to this question (the response could be anyone of {Democrat, Republican, Independent, Other, None}).

The last two items were included to increase the likelihood that participants would have some knowledge of US politics. After providing informed consent, participants were asked to reflect on left- and right-wing tweets. They answered one question about each side of the ideological spectrum:

> What do you expect to see in a tweet that expresses a 'left' ['right'] wing ideology? You could write about possible topics addressed by such tweets, and or opinions you expect the authors to have about these topics (min. response length: 100 characters).

The display order of the two questions was randomized among the participants. The responses to these questions were not analyzed (the purpose of these questions was to make participants think about the left-to-right ideological spectrum before they would respond to the scaling questions.

On the next page, they received short instructions about the position rating task.

> For each of the 30 tweets, you will be asked the following question:
>
> Where does this tweet stand on the 'left' to 'right' wing scale? You will report your response using a continuous slider that goes from 'Extremely left' to 'Extremely right'. There is no right or wrong answer. We are interested in your subjective opinion. If

the text does not have political content, select 'Not Applicable'. Note that the slider will appear on the screen 6 seconds after the text of the tweet.

Then they looped through the 30 tweets and provided their position ratings for each of them.

Finally, the study ended with a short demographic questionnaire and some questions about their political orientation, their political opinions, and their frequency of use of Twitter (not analyzed in this article).

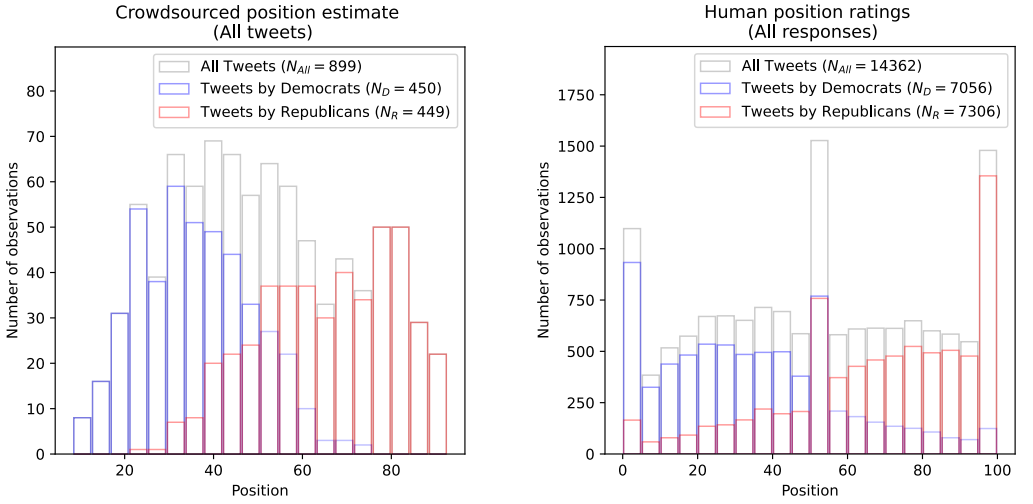

Figure A1. Tweets published by members of the 118th US Congress: Left panel:Histograms of responses by human respondents to the question about the position of each tweet on the Left-Right ideological spectrum (left panel). Right panel: Histogram of crowdsourced position estimates.

Appendix 1.2 Senators of the 117th US Congress

We obtained the list of senators from VoteView.com and the Twitter usernames of most senators from the data published with Le Mens et al. (2023a) (`lattest_legislators_info.csv`). We complemented the list of usernames with a manual search on Twitter. We deposited a table that provides the Twitter usernames we used for each of the senators in the replication folder. (Twitter developer terms prevent us from posting the Tweets we used to position the senators but allow us to post the Tweet IDs of the tweets so that users with access to the API can easily re-download them).

We downloaded the tweets they published during the Congress session via the Twitter API (in 05/2023 and 11/2023).

For each senator, we randomly sampled 100 tweets (with publication dates between January 3 and January 3, 2023). Those who published fewer than 100 tweets during the congressional session were excluded from the analysis, resulting in a set of 97 senators.

Appendix 1.3 British party manifestos

We downloaded the replication materials for Benoit et al. (2016) from https://github.com/kbenoit/CSTA-APSR/tree/master (retrieved September 22, 2023). These consist of a folder `CSTA-APSR-master` from which we obtained the text of the party manifestos and the position estimates provided by expert surveys, sentence scaling by experts, and crowdsourced sentence scaling.

The texts of the party manifestos used for scaling on the economic and social dimension were obtained from the file `CSTA-APSR-master/DataCreated/master.sentences.csv` which consisted of a table of sentences.

We used the same procedure for the texts of the party manifestos used for scaling on the immigration dimension from the file `CSTA-APSR-master/DataCF jobs/immigration/CFjobsresultsImmigration/f354277_immigration1.csv`.

Appendix 1.4 EU legislative speeches

We used the text of the speeches available in the replication package for Benoit et al. (2016) (in folder `07 Additional Texts`). To construct the crowdsourced position estimates of the speech based on translations in English, German, Greek, Italian, Polish, and Spanish, we used the raw crowdsourcing coding files (`fEnglishCombined.csv`, `f655316.csv`, `f656054.csv`, `f656640.csv`, `f658174.csv`, `f658191.csv`).

Appendix 2. Constructing and submitting prompts to LLMs

We used different functions to interact with the different models (see below). Most of these functions use the same parameters and parameter values:

- `messages`: a Python list of dictionaries (with just one 1 element in our case) that contains the prompt (see details below).
- `max_tokens`: 20. This parameter cuts the response of the LLM to a maximum of 20 tokens. It does not affect the nature of the response (it is not an instruction to respond in a few tokens). We used this small value to limit cost (pay-per-use APIs charge per number of tokens in the prompt and the response) and favor speed (the token generation process is relatively slow).
- `temperature`: 0. This implies that when the model generates tokens in response to the prompt, it picks the most likely token, given the posterior distribution of candidate tokens. This ensures that the output is as deterministic as possible (there can be some randomness if there are ties).
- `top_p`: 1. This parameter is essentially irrelevant when temperature is set to 0.
- `n`: 1. This parameter sets the number of responses returned by the OpenAI API in response to the prompt. (Other models do not have this parameter).

In all cases, `messages` is defined as follows:

```
messages=[{
    "role": "user",
    "content": f"""{system_message["pre_text"]}\n{text}\n{system_message["post_text"]}"""}
]
```

where `text` is the text document to be scaled (e.g., tweet, set of tweets, part of a party manifesto or speech) and `system_message` is a Python dictionary with two entries:

- `pre_text`: Text to be included in the prompt before the text to be scaled.
- `post_text`: Text to be included in the prompt after the text to be scaled.

Therefore, `content` is a string variable that contains the text to be scaled and the scaling instructions submitted to the LLM. In some cases, it is further processed by a utility function to add some special tokens necessary for a particular LLM.

Appendix 2.1 OpenAI API

- GPT-4 (`gpt-4-0613`): We used the following command:

```
response_raw = client.chat.completions.create(
  model=llm,
  messages = messages,
  max_tokens= max_tokens,
  temperature=temperature,
       top_p = top_p,
  n = n
)
```

- GPT-4o (`gpt-4o-2024-05-13`), GPT-4 Turbo (`gpt-4-turbo-2024-04-09`), GPT-4o mini (`gpt-4o-mini-2024-07-18`) and GPT 3.5 Turbo (`gpt-3.5-turbo-0125`): We used the following command (a JSON mode is available for these models but not for the earlier models):

```
response_raw = client.chat.completions.create(
    model=llm,
    response_format={ "type": "json_object" },
    messages=messages,
    max_tokens= max_tokens,
    temperature=temperature,
    top_p = top_p,
    n = n
    )
```

Appendix 2.2 Mistral AI API

The variable `messages` is processed by a useful function part of the `mistralai` Python package before submission to the LLM via the API.

```
messages_Mistral_API = [ChatMessage(role="user", content=messages[0]['content'])]
```

- Models published in 2023: `open-mixtral-8x7b`: We used the following command from the `mistralai` Python package:

```
response_raw = client.chat(
    model=llm,
    messages=messages_Mistral_API,
    max_tokens = max_tokens,
    temperature = temperature,
    top_p = top_p,
    )
```

- Models published in 2024: `mistral-large-2402`, `mistral-large-2407`, `open-mixtral-8x22b`, `open-mistral-nemo-2407`: We used the following command from the `mistralai` Python package (a JSON mode is available for those model but not for the models published in 2023):

```
response_raw = client.chat(
    model=llm,
    response_format={ "type": "json_object" },
    messages = messages_Mistral_API,
    max_tokens = max_tokens,
    temperature=temperature,
    top_p = top_p,
    )
```

Appendix 2.3 Running models with the llama-cpp-python Python library

Some LLMs are available for download in .gguf format, in particular quantized models (with compressed weights) that can be run locally on the user's computer (with Apple Silicon or a powerful consumer-grade GPU such as Nvidia RTX3090). These models are executed with the llama-cpp-python Python library. Prompts need to be formatted for each model (by adding some special tokens before and after the main message, and that differ across models). To construct the prompt from the message, we used Huggingface's transformer's library. The Huggingface website contains a vast catalogue of LLMs, with model architecture, weights, and associated utilities. By pointing the transformer library to the URL of a particular LLM (`hf_path` below), the user can access utility functions adapted to that LLM (tokenizer.apply_chat_template). We used the following commands:

```
tokenizer = AutoTokenizer.from_pretrained(hf_path)
prompt = tokenizer.apply_chat_template(messages,
    tokenize=False,
    add_generation_prompt=True
    )
model_local = Llama(model_path = model_path,
    n_ctx = config_LLM.max_tokens_text_scaling,
    n_gpu_layers = -1,
    use_mlock = True,
    logits_all = False,
    )
response_raw = model_local(
    prompt = prompt,
    max_tokens = max_tokens,
    temperature=temperature,
    top_p=top_p,
    )
```

Here `Llama` is a function that loads a model stored locally (at path `model_path` into the function `model_local`. Then the prompt is submitted to the LLM via this function.

Appendix 2.4 Running models on an Apple computer with the mlx-lm Python library

The `mlx-lm` Python library is a deep learning framework optimized for Apple computers with 'Apple Silicon' processors (e.g. variants of M1, M2, M3). It plays the same role for these processors as that of the widely used Pytorch library for the NVidia GPUs.

The LLMs executed with the `mlx-lm` Python library were downloaded from HuggingFace.co. With each model comes a tokenizer that, among other things, can convert a prompt to the format adapted to the particular model used for scaling. The prompt query was then submitted to the LLM via the `generate` function of the `mlx-lm` library.

```
model_mlx, tokenizer = load(mlx_path)
prompt = tokenizer.apply_chat_template(
    messages, tokenize=False, add_generation_prompt=True
    )
response_raw = generate(model_mlx,
    tokenizer,
    prompt=prompt,
    max_tokens = max_tokens,
    temp = temperature,
    top_p=top_p,
    verbose=True)
```

Here the model is stored locally in folder `mlx_path`. The function `load` from the `mlx-lm` library loads the LLM in the `model_mlx` variable and the tokenizer in the `tokenizer` variable. Then the tokenizer is used to pre-process the messages to create the prompt. And finally, the prompt is submitted to the LLM via the `generate` function from the `mlx-lm` library.

Appendix 2.5 Running models on an Huggingface inference endpoint

The Huggingface model sharing platform provides a service called 'Inference endpoints' that allows users to run LLMs on dedicated cloud machines provided by Amazon Web Services (AWS), Microsoft Azure or Google Cloud Platform, a location for the machine (e.g., in the US or in Europe). It offers a graphical interface that allows the user to choose an LLM (e.g., Llama 3 70B Instruct), a GPU infrastructure on which to run it (with different amounts of video memory), and various configuration parameters (such as the maximum number of tokens). Once the setup is completed, the user is provided with a custom URL that is then used to send the prompts to the LLM (stored in the variable `API_URL` in the code snippet below). The cost is per hour (between 1 USD for a machine with 24GB of video memory that can run models such as Llama 3 8B to 16 USD for a machine with 320GB of video memory that can run models such as Llama 3 70B).

```
headers = {
"Accept" : "application/json",
"Authorization": f"Bearer {HF_TOKEN}",
"Content-Type": "application/json"
}

def hf_inference_endpoint_generate(payload):
response = requests.post(API_URL, headers=headers, json=payload)
return response.json()

prompt = tokenizer.apply_chat_template(
    messages, tokenize=False, add_generation_prompt=True
    )

response_raw = hf_inference_endpoint_generate({
"inputs": prompt,
"parameters": {
"top_p": 0.999,
"temperature": 0.001,
```

```
"max_new_tokens": max_tokens,
"do_sample": do_sample,
"return_text": False,
"return_full_text": False,
"return_tensors": False,
"clean_up_tokenization_spaces": True
}
})
```

Here, `HF_TOKEN` is a unique identification token users of Huggingface can create in their account settings. The funcion `hf_inference_endpoint_generate` sends the prompt `prompt` to the dedicated cloud machine.

Appendix 3. Messages submitted to LLMs

In all cases, `messages` is defined as follows:

```
messages=[{
    "role": "user",
    "content": f"""{system_message["pre_text"]}\n{text}\n{system_message["post_text"]}"""}
]
```

Next, we report on the values of `pre_text` and `post_text` in the various settings.

Appendix 3.1 Tweets

Appendix 3.1.1 pre_text

You will be provided with the text of a tweet published by a member of the US Congress. Where does this text stand on the 'left' to 'right' wing scale? Provide your response as a score between 0 and 100 where 0 means 'Extremely left' and 100 means 'Extremely right'. If the text does not have political content, set score to 'NA'. You will only respond with a JSON object with the key Score. Do not provide explanations.

Appendix 3.1.2 post_text

Left empty.

Appendix 3.2 Tweets published by each senator

Appendix 3.2.1 pre_text

You will be provided with the text of a tweet published by a member of the US Congress. Where does this text stand on the 'left' to 'right' wing scale? Provide your response as a score between 0 and 100 where 0 means 'Extremely left' and 100 means 'Extremely right'. If the text does not have political content, set the score to "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

Appendix 3.2.2 post_text

Left empty.

Appendix 3.3 Sets of tweets published by each senators

Appendix 3.3.1 pre_text

You will be provided with the text of a set of tweets published by a member of the US Congress. Tweets are separated by a line break followed by the string of characters <TWEET>.

Appendix 3.3.2 post_text

Where does the author of these tweets stand on the 'left' to 'right' wing scale? Provide your response as a score between 0 and 100 where 0 means 'Extremely left' and 100 means 'Extremely right'. If the text does not have political content, set the score to "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

Appendix 3.4 British party manifestos - scaling without explanations about the policy dimensions.

Appendix 3.4.1 pre_text

You will be provided with a sentence from a party manifesto.

Appendix 3.4.2 post_text

- Economic policy:

  Where does this sentence stand on the 'left' to 'right' wing scale, in terms of economic policy? Provide your response as a score between 0 and 100 where 0 means 'Extremely left' and 100 means 'Extremely right'. If the sentence does not refer to economic policy, return "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

- Social Policy:

  Where does this sentence stand on the 'liberal' to 'conservative' scale, in terms of social policy? Provide your response as a score between 0 and 100 where 0 means 'Extremely liberal' and 100 means 'Extremely conservative'. If the sentence does not refer to social policy, return "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

Appendix 3.5 British party manifestos - scaling with explanations about the policy dimensions.

Appendix 3.5.1 pre_text

You will be provided with a sentence from a party manifesto.

Appendix 3.5.2 post_text

We used `post_text` that included explanations about the scale dimension and the two ends of the scale designed to be as close as possible to the instructions given to crowdworkers in Benoit et al. (2016).

- Economic policy:

  Where does this sentence stand on the 'left' to 'right' wing scale, in terms of economic policy?

  "Economic" policies deal with all aspects of the economy, including:

  Taxation;

  Government spending;

  Services provided by the government or other public bodies;

  Pensions, unemployment and welfare benefits, and other state benefits;

  Property, investment and share ownership, public or private;

  Interest rates and exchange rates;

  Regulation of economic activity, public or private;

  Relations between employers, workers and trade unions.

  "Left" economic policies tend to favor one or more of the following:

  High levels of services provided by the government and state benefits, even if this implies high levels of taxation;

  Public investment. Public ownership or control of sections of business and industry;

  Public regulation of private business and economic activity;

  Support for workers/trade unions relative to employers.

  "Right" economic policies tend to favor one or more of the following:

  Low levels of taxation, even if this implies low levels of levels of services provided by the government and state benefits;

  Private investment. Minimal public ownership or control of business and industry;

Minimal public regulation of private business and economic activity;
Support for employers relative to trade unions/workers.
Provide your response as a score between 0 and 100 where 0 means 'Extremely left' and 100 means 'Extremely right'. If the sentence does not refer to economic policy, return "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

- Social Policy:

Where does this sentence stand on the 'liberal' to 'conservative' scale, in terms of social policy?
"Social" policies deal with aspects of social and moral life, relationships between social groups, and matters of national and social identity, including:
Policing, crime, punishment and rehabilitation of offenders;
Immigration, relations between social groups, discrimination and multiculturalism;
The role of the state in regulating the social and moral behavior of individuals.
"Liberal" social policies tend to favor one or more of the following:
Recording the contents of political text on economic and social scales;
Policies emphasizing prevention of crime, rehabilitation of convicted criminals;
The right of individuals to make personal moral choices on matters such as abortion, gay rights, and euthanasia;
Policies penalizing discrimination against particular social groups and/or favoring a multicultural society.
"Conservative" social policies tend to favor one or more of the following:
Policies emphasizing more aggressive policing, increasing police numbers, conviction and punishment of criminals, building more prisons;
The right of society to regulate personal moral choices on matters such as abortion, gay rights, and euthanasia;
Policies favoring restriction of immigration, and/or opposing explicit provision of state services for minority cultures.
Provide your response as a score between 0 and 100 where 0 means 'Extremely liberal' and 100 means 'Extremely conservative'. If the sentence does not refer to social policy, return "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

Appendix 3.6 EU legislative speeches - scaling with background explanations.

We used the following `pre_text` that included background information about the context of the legislative debate (especially the nature of the proposal being discussed), the scale dimension and the two ends of the scale designed to be as close as possible to the instructions given to crowdworkers in Benoit et al. (2016):

Appendix 3.6.1 pre_text

Summary This task involves reading a sentence from a debate over policy in the European parliament, and judging whether particular statements were for or against a proposed policy.

Background.

The debates are taken from a debate in the European Parliament over the ending of state support for uncompetitive coal mines. In general, state aid for national industry

in the European Union is not allowed, but exceptions are made for some sectors such as agriculture and energy. At stake here were not only important policy issues as to whether state intervention is preferable to the free market, but also the specific issue for some regions (e.g. Ruhrgebiet in Germany, the north-west of Spain, the Jiu Valley in Romania) where the social and economic impacts of closure would be significant, possibly putting up 100,000 jobs at risk when related industries are considered.

Specifically, the debate concerned a proposal by the European Commission to phase out all state support by 2014. Legislation passed in 2002 that allowed for state subsidies to keep non- profitable coal mines running was due to end in 2010. The Commission proposed to let the subsidies end, but to allow limited state support until 2014 in order to soften the effects of the phase-out. A counter proposal was introduced to extend this support until 2018, although many speakers took the opportunity to express very general positions on the issue of state subsidies and energy policy.

Your Coding Job.

Your key task is to judge individual sentences from the debate according to which of two contrasting positions they supported:

- Supporting the rapid phase-out of subsidies for uncompetitive coal mines. This was the essence of the council proposal, which would have let subsides end while offering limited state aid until 2014 only.
- Supporting the continuation of subsidies for uncompetitive coal mines. In the strong form, this involved rejecting the Commission proposal and favoring continuing subsidies indefinitely. In a weaker form, this involved supporting the compromise to extend limited state support until 2018.

The 'pro-subsidy' to 'anti-subsidy' scale.

The "anti-subsidy" position was the essence of the Commission proposal, which would have let subsidies end while offering limited state aid until 2014 only. Examples of "anti-subsidy" positions:

Statements declaring support for the commission position.

Statements against state aid generally, for reasons that they distort the market.

Arguments in favor of the Commission phase-out date of 2014, rather than 2018.

In the strong form, the "pro-subsidy" position, involved rejecting the Commission proposal and favoring continuing subsidies indefinitely. In a weaker form, this involved supporting the compromise to extend limited state support until 2018.

Examples of "pro-subsidy" positions: Arguments that keeping the coal mines open to provides energy security.

Arguments that coal mines should be kept open to provide employment and other local economic benefits.

Preferences for European coal over imported coal, for environmental or safety reasons.

Sentence:

Appendix 3.6.2 post_text

Instructions.

Where does this sentence stand on the 'anti-subsidy' to 'pro-subsidy' scale? Provide your response as a score between 0 and 100 where 0 means 'Extremely anti-subsidy' and 100

means ‘Extremely pro-subsidy.’ If the sentence does not refer to subsidy policy, return "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

Appendix 4. Other Scaling Methods

Appendix 4.1 British party manifestos - fine-tuned BERT classifier

Note: We used a Python Jupyter notebook executed on Google Colab adapted from that made available in the companion OSF folder to the paper on using machine learning to measure typicality by Le Mens et al. (2023b) (https://osf.io/ta273/).

We implemented a BERT classifier that would predict the sentence coding crowdworkers in Benoit et al. (2016). For the manifestos used for scaling on the Economic and Social policy dimensions, we specified an 11 class sentence classifier in which the classes were the 5 positions on the Economic policy dimension [-2,-2,0,1,2] the 5 positions on the Social policy dimension [-2,-2,0,1,2], and 'nan' in case a crowdworker judged a sentence to be neither about economic nor social policy. We divided the data into training, validation and prediction sets, ensuring that all ratings for a given sentence were in just one of these three sets. We used 45% of the sentences for training, 5% for validation, and 50% for prediction. This produced a training set with 98K human ratings, a validation set with 10K human ratings, and a test set with 107K human ratings.

We implemented our BERT model using the `bert-base-uncased`, the TensorFlow machine-learning library and its higher-level wrapper Keras. We used the Adam optimizer to minimize the cross-entropy loss function to fine-tuning the model. This amounts to trying to find the parameters that maximize the likelihood of the true categories in the training data. We used Google Colab with a TPU-equipped virtual machine to train the model.

We used the following (standard) parameter values:

- Batch size: 256
- Max number of tokens: 512 (this is the maximum possible with `bert-base-uncased`)
- Optimizer: adam
- Loss function: categorical_crossentropy
- Learning rate: 2e-5
- Epochs: 200 (we used a training routine that would treat this parameter as the maximum possible number of epochs - it automatically stopped the training when the validation loss started to increase)

We applied the fine-tuned model on the sentences in the prediction set. This gave a vector of categorization probabilities in the 11 candidate classes. To obtain the policy area of the focal sentence, we first added the categorization probabilities for the 5 possible dimensions of the economic policy dimension, and did the same for the Social Policy dimension. This gives a vector of 3 probabilities for Economic Policy, Social Policy and 'Other'. We then assigned the focal sentence to the most likely policy area (including 'Other').

To obtain the position on the economic policy dimension, of a sentence about Economic policy (as per the result of the classification in policy areas described above), we took the weighted average of the 5 possible positions on that dimension, where the weights were the categorization probabilities in these 5 classes (rescaled to sum to 1). From this, we constructed a position estimate for each manifesto as the average position of its sentences.

We followed the same procedure to obtain position estimates of sentences and then manifestos on the Social policy dimension.

Appendix 4.2 Tweets - fine-tuned BERT classifier

Note that for tweets, we did not train the models on data produced by human coders, but instead trained the classifiers to predict the party of the Congress member who published the tweet (Democratic Party or Republican Party), following the approach proposed by Le Mens et al. (2023b) and implemented on these Twitter data in Le Mens et al. (2023a). We used the approach used in this latter paper by slightly adapting the jupyter python notebooks made available at on the companion OSF

folder (https://osf.io/ta273/). We used the data from that paper. The training set consisted in close to 1M tweets published by US Congress members during the 116th and 117th Congress sessions.

For the tweets published by the senators of the 117th US congress, we used as a training set the tweets published by senators of the 116th and 117th Congress sessions, ensuring that none of the tweets in the prediction set (100 tweets per senator) was in the training or validation set. This resulted in a training set of about 246K tweets. We used the categorization probabilities in the two parties to compute the typicality of each tweet in each party (following Le Mens et al. (2023b)) and took the difference in typicalities in the Republican and Democratic parties as the position of the tweet on the left-right ideological spectrum. The position estimate of a senator is the average position of the 100 tweets in the prediction set.

For tweets published after the training cut-off date of GPT-4, we used the same training set as Le Mens et al. (2023a) and simply computed the position of each tweet following the approach explained in the previous paragraph.

We used the following (standard) parameter values:

- Batch size: 256
- Max number of tokens: 512 (this is the maximum possible with `bert-base-uncased`)
- Optimizer: adam
- Loss function: categorical_crossentropy
- Learning rate: 2e-5
- Epochs: 200 (we used a training routine that would treat this parameter as the maximum possible number of epochs – it automatically stopped the training when the validation loss started to increase)

Appendix 4.3 Tweets - fine-tuned GloVe classifier

Note: We used a Python Jupyter notebook executed on Google Colab adapted from that made available in the companion OSF folder to the paper on using machine learning to measure typicality by Le Mens et al. (2023b) (https://osf.io/ta273/). The following text is a minor adaptation of the description of the fine-tuned GloVe classifier in that paper.

We adapted the approach described in Le Mens et al. (2023b) to the present empirical setting. We used the same model structure as for the BERT classifier, but we used GloVe word embeddings as a language representation instead of the BERT language representation. More precisely, we used a word embedding layer with pre-trained weights in our classifiers instead of the BERT language representation. We used GLOVE word embeddings (Pennington, Socher, and Manning 2014) to transform text documents into vectors. GLOVE is a *word*-embedding model, not a text-embedding model. Consequently, we needed to combine word positions in the embedding space to create a unique position for text documents. We used the average position of the words in the text as the position of the tweet in semantic space.

We selected the top 20,000 most frequent words in the training data, then we used the Glove embedding model "glove.6B.300d" to transform each of the 20,000 words into vector positions. With this embedding, we created a basic deep-learning model consisting of:

1. Embedding layer
2. Pooling 1D layer
3. Dense layer (with softmax activation)

We trained the model for up to 1,000 epochs with a $2e-3$ learning rate using a categorical cross entropy loss function and a batch size of 16,384.

Appendix 4.4 Tweets - fine-tuned Naive Bayes classifier with tf-idf language representation

Note: We used a Python Jupyter notebook executed on Google Colab adapted from that made available in the companion OSF folder to the paper on using machine learning to measure typicality by Le Mens et al. (2023b) (https://osf.io/ta273/). The following text is a minor adaptation of the description of the naive Bayes classifier in that paper.

We also used a standard machine-learning text classifier based on a version of the Bag-of-Words representation that weighs word frequencies by diminishing the importance of words that occur in many text documents. This approach is known as 'TF-IDF', or 'term frequency-inverse document frequency', (Jones 1972). The classifier model is the Naive Bayes classifier (Maron 1961). This machine-learning classifier produces categorization probabilities based on word co-occurrences. It is computationally undemanding, but its representation of text documents is not sensitive to the order of words in sentences. Also, the representation of words does not depend on their semantic similarity.

To train the classifier applied on the tweets of the senators of the 117th US Congress (Section 3.2), we selected the 5,000 more frequent words from the training data (this number is constrained by the RAM of the computer on which we ran the model training, which in this case was a virtual machine on Google Colab with 50GB of RAM). We assigned an ID to each word and transformed all the book descriptions using this ID dictionary. Finally, we fit a multinomial Naive Bayes on all the book descriptions in the training data.

To train the classifier applied to the tweets published after the GPT-4 training cut-off date (Section 3.1), we selected the 2,000 more frequent words from the training data (this number is lower than in the previous paragraph because the training data is larger in this case and thus running the model training requires more RAM for the same vocabulary size).

For the construction of the dictionary, we used the sklearn package "CountVectorizer" and to fit the model we used sklearn package "MultinomialNB."

Appendix 5. Additional Results - Tweets published by members of the US Congress after the training cut-off date of GPT-4

Visual comparison of the performance of the various scaling methods (LLMs, fine-tuned BERT classifier, fine-tuned GloVe classifier, and naive Bayes classifier with tf-idf representation).

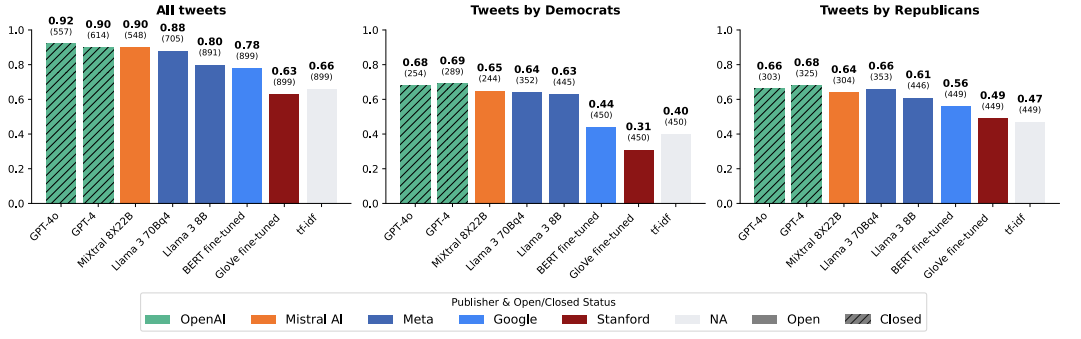

Figure A2. Positioning tweets published by members of the US Congress after the training cut-off date of GPT-4 on the left-right ideological spectrum (N = 899). Pearson correlations between the benchmark (crowdsourced position estimates) and position estimates obtained with LLMs. The numbers in parentheses indicate the number of documents for which a position estimate was obtained (position estimates are missing if the LLM did not return a position score for any of the parts of manifestos submitted to it). The data are the same as for Figure 1.

The LLMs failed to return position scores for some of the tweets, possibly because these did not include enough political content. To verify that this pattern of non-response is reasonable, we computed the average number of human ratings for the two sets of tweets, for each LLM (survey respondents had the option to respond 'Not applicable' to the question about the position of each tweet on the Left-Right ideological spectrum, if they felt the tweet did not contain enough information for them to return a position rating). Under the hypothesis that an LLM fails to return a numeric score for tweets that did not contain enough political content, but would do so for tweets than contain enough political content, we expected that the average number of human position ratings would be larger for tweets that received a position score by the LLM than for tweets that did not receive a position score. These quantities are reported in the last two columns of table A1. For all the LLMs, the pattern is consistent with the hypothesis.

Table A1. Positioning tweets published by members of the US Congress after the training cut-off date of GPT-4 on the left-right ideological spectrum (N = 899). The last two columns report the average number of human position ratings for tweets that received a position score, and the same quantity for those for which the LLM did not return a position score. This table includes the LLMs from the body of the paper as well as a set of additional LLMs (from the previous generation). 'Open' LLMs can be downloaded on the user's machine and run locally if the machine has enough video memory (under more or less restrictive licenses such as Apache 2.0, the Meta Llama 3 community license agreement, or the Mistral AI Research License). 'Local' execution means that the model was run on our laptop after downloading the model 'API' execution means that the model was run for a per-token fee on a machine managed by the provider of the model. 'Cloud' execution means that the model was run via a cloud computing service (Google Colab or Huggingface Inference Endpoints) on a machine equipped with a TPU or GPUs (payment per hour / compute unit). Models in .gguf format are versions of the model with compressed ('quantized') weights and were run locally with the `llama-cpp-python` package. In the full names of these models,'QX' indicates the number of bits used to represent the model weights. We used minimal compression that would allow us to run the model on our laptop (an Apple Macbook Pro with an M1 Max processor with 64GB that can run models as large as 40GB). Llama 3.1 8B, Llama 3 8B, Llama 2 7B, Llama 2 13B, Gemma 1.1 7B, Gemma 2 9B, Aya 23 8B were downloaded from Huggingface.co and were run locally without any compression with the `mlx-lm` Python package. For Gemma 2 27B, we used the `mlx-lm` Python package to apply an 8-bit quantization convertion and to run it locally.

				All tweets (N=899)		Democratic tweets		Republican tweets		Average number of human ratings over	
Model	Model (full name)	Open / Closed	Execution (Local / Cloud / API)	*r*	Number of position scores	*r*	Number of position scores	*r*	Number of position scores	Tweets for which the LLM returns a position score	Tweets for which the LLM does NOT return a position score
GPT-4o	gpt-4o-2024-05-13	Closed	OpenAI API	.92	557	.68	254	.66	303	18.7	11.6
GPT-4 Turbo	gpt-4-turbo-2024-04-09	Closed	OpenAI API	.94	469	.72	201	.67	268	19.1	12.6
GPT-4	gpt-4-0613	Closed	OpenAI API	.90	614	.69	289	.68	325	18.4	10.7
GPT-4o mini	gpt-4o-mini-2024-07-18	Closed	OpenAI API	.85	626	.66	310	.58	316	18.4	10.5
GPT-3.5 Turbo	gpt-3.5-turbo-0125	Closed	OpenAI API	.76	840	.27	424	.68	416	16.6	7.0
Mistral Large (07/24)	mistral-large-2407	Open	Mistral AI API	.91	515	.65	235	.62	280	18.9	12.1
Mistral Large (02/24)	mistral-large-2402	Closed	Mistral AI API	.87	433	.59	169	.50	264	19.2	13.0
MiXtral 8X22B v0.1	open-mixtral-8x22b	Open	Mistral AI API	.90	548	.65	244	.64	304	18.6	11.9
MiXtral 8X7B v0.1	open-mixtral-8x7b	Open	Mistral AI API	.71	385	.29	134	.34	251	19.2	13.5
MiXtral 8X7B v0.1	mixtral-8x7b-instruct-v0.1.Q6_K.gguf	Open	Local (llama-cpp-python)	.78	422	.44	154	.38	268	19.1	13.2
MiXtral 8x7B v0.1	mixtral-8x7b-instruct-v0-1-bav	Open	HF Cloud	.81	406	.39	145	.45	261	19.2	13.3
Mistral Nemo (07/24)	open-mistral-nemo-2407	Open	Mistral AI API	.68	704	.63	345	.33	359	17.8	9.3
Mistral Nemo (07/24)	Mistral-Nemo-Instruct-2407	Open	Local (mlx-lm)	.62	664	.64	321	.25	343	18.1	10.0
Mistral 7B v0.2	Mistral-7B-Instruct-v0.2	Open	Local (mlx-lm)	.42	524	.22	242	.22	282	18.8	12.0
Mistral 7B v0.2	mistral-7b-instruct-v0-2-kmz	Open	HF Cloud	.37	534	.18	249	.22	285	18.8	11.9
Llama 3 70B	Meta-Llama-3-70B-Instruct-Q4_K_M.gguf	Open	Local (llama-cpp-python)	.88	705	.64	352	.66	353	17.9	9.1
Llama 3 70B	meta-llama-3-70b-instruct-amm	Open	HF Cloud	.87	773	.64	380	.72	393	17.4	7.4
Llama 3.1 8B	Meta-Llama-3.1-8B-Instruct	Open	Local (mlx-lm)	.78	893	.57	445	.58	448	16.0	6.0
Llama 3 8B	Meta-Llama-3-8B-Instruct	Open	Local (mlx-lm)	.80	891	.63	445	.61	446	16.1	4.9
Llama 3 8B	Meta-Llama-3-8B-Instruct_hf_inference	Open	HF Cloud	.80	893	.57	445	.61	448	16.1	4.8
LLama 2 70B	llama-2-70b-chat.Q4_K_M.gguf	Open	Local (llama-cpp-python)	.71	847	.51	433	.55	414	16.3	10.1
Llama 2 13B	Llama-2-13b-chat-hf	Open	Local (mlx-lm)	.37	165	-.26	66	.54	99	13.5	16.5
Llama 2 7B	Llama-2-7b-chat-hf	Open	Local (mlx-lm)	.18	899	-.36	450	.39	449	16.0	-
Aya 23 35B	aya-23-35b-qr	Open	HF Cloud	.83	644	.54	305	.71	339	18.2	10.4
Aya 23 8B	Aya-23-8b	Open	Local (mlx-lm)	.39	625	.04	291	.33	334	17.5	12.5
Gemma 2 27B	gemma-2-27b-it	Open	Local (mlx-lm) 8bit quantization	.90	586	.73	265	.66	321	18.6	11.1
Gemma 2 9B	gemma-2-9b-it	Open	Local (mlx-lm)	.71	667	.42	324	.56	343	18.1	9.8
Gemma 1.1 7B	gemma-1.1-7b-it.gguf	Open	Local (llama-cpp-python)	-.06	772	-.02	394	-.02	378	16.9	10.2
BERT fine-tuned	bert-base-uncased	Open	Cloud (Google Colab)	.78	899	.44	450	.56	449	16.0	-
GloVe fine-tuned	glove.6B.300d.txt	Open	Cloud (Google Colab)	.63	899	.31	450	.49	449	16.0	-
tf-idf	sklearn.naive_bayes MultinomialNB	Open	Cloud (Google Colab)	.66	899	.40	450	.47	449	16.0	-

Appendix 5.1 Distribution of position estimates produced by LLMs and other scaling methods

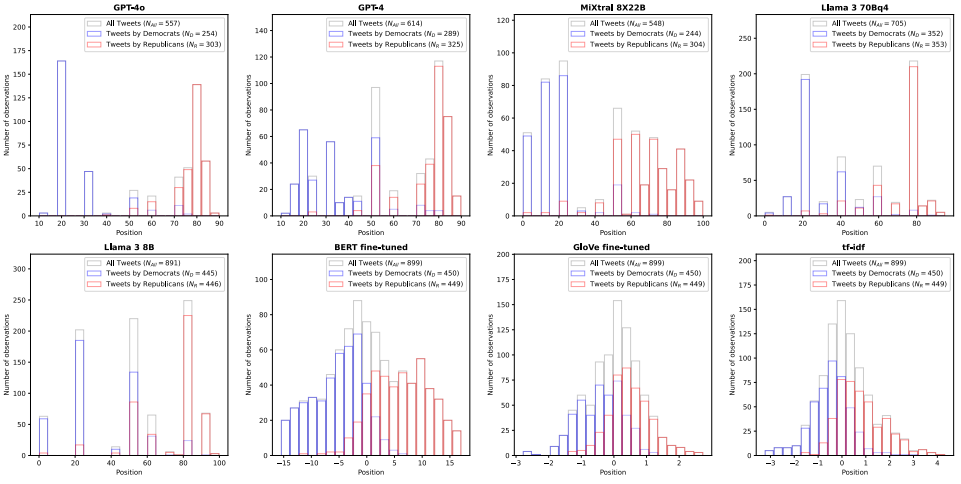

Figure A3. Tweets published by members of the 118th US Congress: Histograms of position estimates produced by the LLMs and other scaling methods.

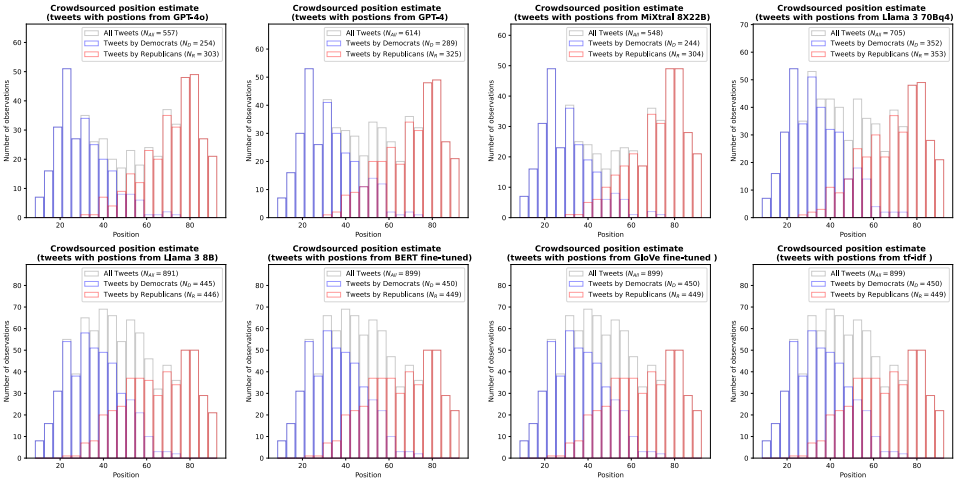

Figure A4. Tweets published by members of the 118th US Congress: Histograms of crowdsourced position estimates for the subset (out of the total of 899 tweets) for which each scaling method produced a position estimate).

Appendix 5.2 Scaling tweets on the left-right ideological spectrum by using LLMs to obtain their typicalities in the Democratic and Republican parties

We implemented the approach advocated in Le Mens et al. (2023a) to obtain typicality measures of tweets in political parties (and illustrated in that paper with GPT-4). This involves directly asking the LLMs for the typicality of each tweet in each party. We then construct a position estimate of each tweet on the left-right ideological spectrum as the difference between its typicality in the Republican party and its typicality in the Democratic party. Le Mens et al. (2023a) had found that the typicality measures obtained with GPT-4 had higher correspondence with human typicality ratings than those produced with other methods, such as a BERT classifier or classifiers based on text embeddings (ADA2 embeddings published by OpenAI) or text embeddings (GloVe) that were trained on vast amounts of data.

Appendix 5.2.1 Messages submitted to LLMs.

We used the same message structure as explained in Appendix 3 except for a change to the '`pre_text`' part of the user messages.

Appendix 5.2.2 pre_text for typicality in Democratic Party.

> You will be provided with the text of a tweet published by a member of the US Congress. How typical is this tweet of the Democratic Party? Provide your response as a score between 0 and 100 where 0 means 'Not typical at all' and 100 means 'Extremely typical'. You will only respond with a JSON object with the key Score. Do not provide explanations.

Appendix 5.2.3 pre_text for typicality in Republican Party.

> You will be provided with the text of a tweet published by a member of the US Congress. How typical is this tweet of the Republican Party? Provide your response as a score between 0 and 100 where 0 means 'Not typical at all' and 100 means 'Extremely typical'. You will only respond with a JSON object with the key Score. Do not provide explanations.

Appendix 5.2.4 post_text.

Left empty.

Appendix 5.2.5 Results.

The correlations with crowdsourced position estimates are similar to those obtained with the baseline approach reported in the body of the article, with some gains in terms of within-party correlation. Another key difference is that, with this approach, LLMs return position estimates for all tweets. It should be noted that there is no need for a tweet to have explicitly political content for an artificial or human judge to evaluate its typicality in a political party. This is because of a fundamental difference between this approach of constructing a position estimate and the baseline approach used in the body of the paper: the baseline approach instructs the LLM to focus on a particular dimension. The approach consists in taking the difference between typicalities in two concepts does not require the specification of a particular dimension: it amounts to a sort of projection on the direction of the vector that connects that centers of the two concepts in multidimensional semantic space.

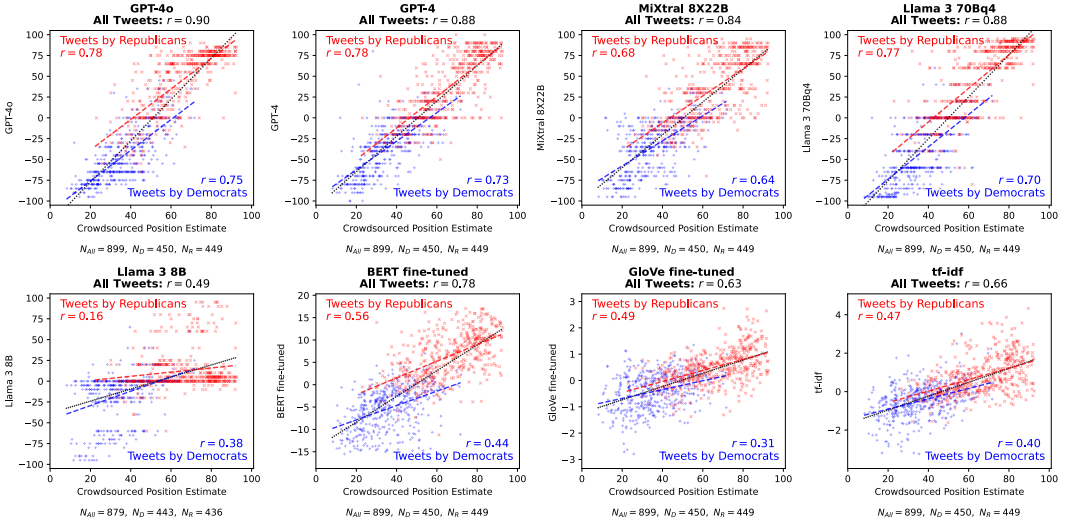

Figure A5. Positioning tweets published by members of the US Congress after the training cut-off date of GPT-4 on the left-right ideological spectrum ($N = 899$) by asking for the typicality in the Republican party, their typicality in the Democratic party and defining the position estimate as the difference between these typicalities. The y-axes report the position of each tweet on the left-right ideological spectrum obtained with the various models. The x-axes reports crowdsourced estimates as the average position provided by participants in an online survey of US residents.

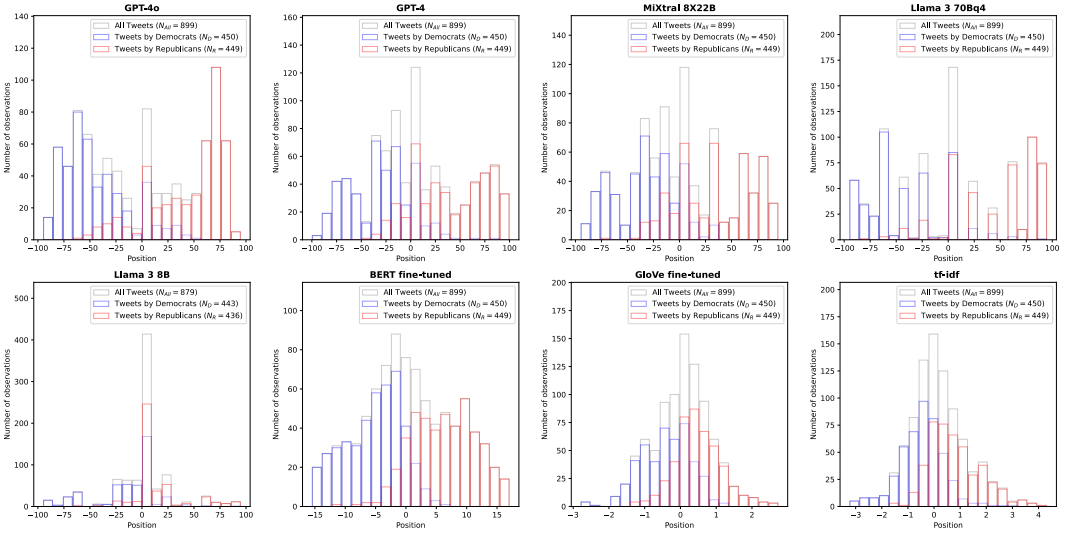

Figure A6. Tweets published by members of the 118th US Congress: Histograms of position estimates produced by the LLMs by taking the difference between the typicality in the Republican party and typicality in the Democratic party.

Appendix 5.3 Predicting the position estimates given by one human respondent to the survey of tweet positions: ENO scores

One existing characterization of the predictive performance of models of human judges focuses on predicting the ratings of one human respondent (Erev et al. 2007). It asks how many observations of other human participants are necessary to make a prediction as good as the model. The resulting number is the model's equivalent number of observations (ENO).

Here, we predict the position ratings of one human participant based on the average of *N* position ratings of other participants. We compute the Pearson correlation between these two quantities. The resulting correlation will depend on the specific ratings selected as criterion and predictor variables. So, for each tweet, we randomly sample one human rating as the criterion variable and treat the remaining ratings as potential predictors. We repeat this procedure 100 times and report the average correlation in Table A2. We do this for tweets that received at least 15 human position ratings (at least 15 respondents in our survey did not select 'NA' for that tweet), which leads to a set of 598 tweets. We repeat the procedure for values of *N* that range from 1 to 14 (the number of human typicality ratings minus 1), and over three sets of tweets: tweets from Democratic and Republican politicians, tweets from Democratic politicians and tweets from Republican politicians.

It should be noted that the column for N=1 reports the pairwise correlation between the position ratings given by two randomly picked human judge, computed over the tweets for which the focal model returned a position score. Because the set of tweets for which each model returned a position score differs across model, this measure of pairwise correlation differs across rows in the column. The same observation applies to the other values of N.

The ENO scores for the tweet position estimates provided by the LLMs are all higher than 1 and go up to 5 to 7 (for GPT-4, MiXtral 8X22B and Llama 3 70Bq4, over all tweets). In the latter case, this means that the position scores provided by these models are as good predictors as the average position ratings given by 5 independent human participants in our survey of Prolific participants who reside in the US.

Table A2. Tweet data: Predicting one human position rating rating with position ratings by other human coders and with the position estimates provided by the LLMs)

					Average of N human position ratings with N equal to													
	Model	**r(model, one human)**	**Number of tweets**	**ENO**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Democratic and Republican Tweets	GPT-4o	.70	524	7	.55	.63	.66	.68	.69	.70	.70	.71	.71	.71	.72	.72	.72	.72
	GPT-4	.68	554	5	.54	.61	.65	.67	.68	.69	.69	.70	.70	.71	.71	.71	.71	.71
	MiXtral 8X22B	.69	507	5	.55	.63	.66	.68	.69	.70	.70	.71	.71	.71	.72	.72	.72	.72
	Llama 3 70Bq4	.67	586	5	.53	.61	.64	.66	.67	.68	.69	.69	.70	.70	.70	.70	.71	.71
	Llama 3 8B	.63	598	3	.53	.61	.64	.66	.67	.68	.69	.69	.70	.70	.70	.70	.71	.71
	BERT fine-tuned	.63	598	3	.53	.61	.64	.66	.67	.68	.69	.69	.70	.70	.70	.70	.71	.71
	GloVe fine-tuned	.50	598	1	.53	.61	.64	.66	.67	.68	.69	.69	.70	.70	.70	.70	.71	.71
	tf-idf	.52	598	1	.53	.61	.64	.66	.67	.68	.69	.69	.70	.70	.70	.70	.71	.71
Democratic Tweets	GPT-4o	.34	240	6	.20	.25	.29	.31	.33	.34	.35	.36	.36	.37	.38	.38	.38	.39
	GPT-4	.35	257	5	.22	.28	.31	.34	.35	.36	.38	.38	.39	.40	.40	.41	.41	.42
	MiXtral 8X22B	.31	230	5	.19	.24	.27	.30	.32	.33	.34	.35	.36	.36	.37	.37	.38	.38
	Llama 3 70Bq4	.32	284	4	.20	.26	.30	.32	.34	.35	.36	.37	.38	.38	.39	.39	.40	.40
	Llama 3 8B	.31	287	4	.20	.26	.30	.32	.34	.35	.36	.37	.38	.38	.39	.39	.40	.40
	BERT fine-tuned	.26	287	2	.20	.26	.30	.32	.34	.35	.36	.37	.38	.38	.39	.39	.40	.40
	GloVe fine-tuned	.13	287	1	.20	.26	.30	.32	.34	.35	.36	.37	.38	.38	.39	.39	.40	.40
	tf-idf	.22	287	1	.20	.26	.30	.32	.34	.35	.36	.37	.38	.38	.39	.39	.40	.40
Republican Tweets	GPT-4o	.34	284	4	.22	.28	.32	.34	.35	.37	.38	.39	.39	.40	.40	.41	.41	.41
	GPT-4	.36	297	3	.25	.31	.35	.37	.39	.40	.41	.42	.42	.43	.43	.44	.44	.44
	MiXtral 8X22B	.31	277	3	.21	.28	.31	.33	.35	.36	.37	.38	.39	.40	.40	.41	.41	.41
	Llama 3 70Bq4	.33	302	2	.25	.32	.36	.38	.40	.41	.42	.43	.44	.44	.45	.45	.45	.46
	Llama 3 8B	.32	311	2	.26	.33	.37	.40	.42	.43	.44	.45	.45	.46	.46	.47	.47	.47
	BERT fine-tuned	.31	311	2	.26	.33	.37	.40	.42	.43	.44	.45	.45	.46	.46	.47	.47	.47
	GloVe fine-tuned	.27	311	1	.26	.33	.37	.40	.42	.43	.44	.45	.45	.46	.46	.47	.47	.47
	tf-idf	.24	311	1	.26	.33	.37	.40	.42	.43	.44	.45	.45	.46	.46	.47	.47	.47

Appendix 6. Additional Results - Senators of the 117th US Congress

Visual comparison of the performance of the various scaling methods (LLMs, fine-tuned BERT classifier, fine-tuned GloVe classifier, and naive Bayes classifier with tf-idf representation).

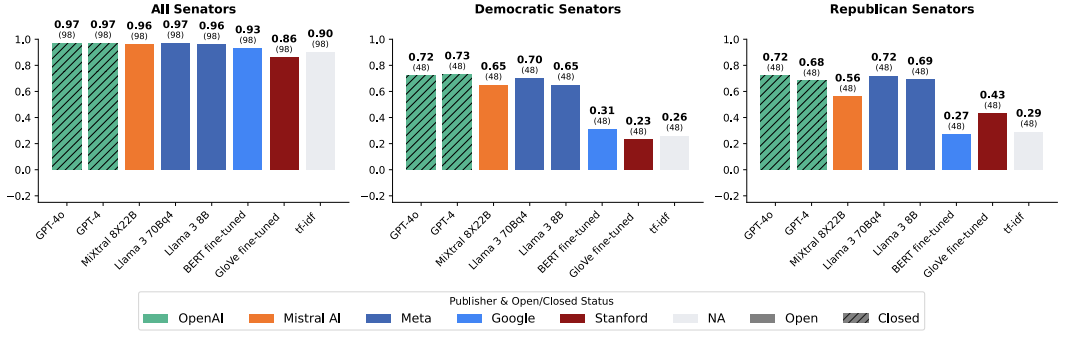

Figure A7. Positioning Senators of the 117th Congress on the left-right ideological spectrum based on their tweets (N = 98). Pearson correlations between the benchmark (First dimension Nokken-Poole period-specific DW-NOMINATE score) and position estimates obtained with LLMs and other scaling methods. The numbers in parentheses indicate the number of senators for which a position estimate was obtained. The data are the same as for Figure 2.

Appendix 6.1 Correlation between the position estimates of the senators (using the baseline approach described in the body of the article) and a position estimate measure based on campaign funding.

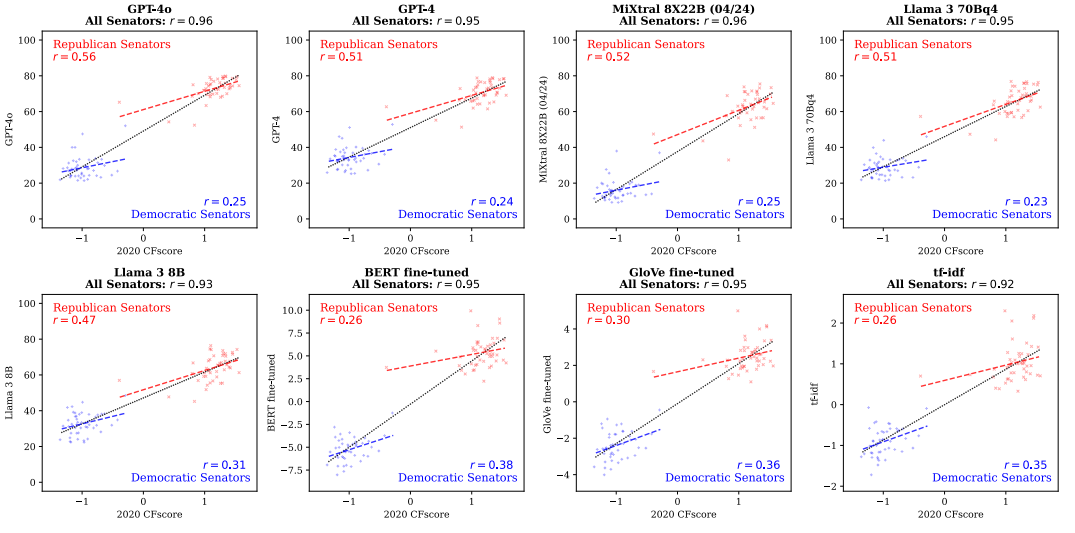

Figure A8. Replication of Figure 2 using the 2020 CFscore from the Dime database (Bonica 2014) instead of the period-specific DW-NOMINATE score. The 2020 CFscore is a period-specific estimate of ideology based on donation received.

Appendix 6.2 Scaling the aggregated text of the 100 tweets in one prompt

We used the user message to obtain the position of each senator based on the text of the 100 tweets:

> You will be provided with the text of a set of tweets published by a member of the US Congress. Tweets are separated by a line break followed by the string of characters <TWEET>.
>
> <TWEET> ≪ Text of tweet 1 ≫
>
> ...
>
> <TWEET> ≪ Text of tweet N ≫
>
> Where does the author of these tweets stand on the 'left' to 'right' wing scale? Provide your response as a score between 0 and 100 where 0 means 'Extremely left' and 100 means 'Extremely right'. If the text does not have political content, set the score to "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

For the models we used, this resulted in a prompt that could fit the maximum context window. This means the prompt could be processed by the model in a single query

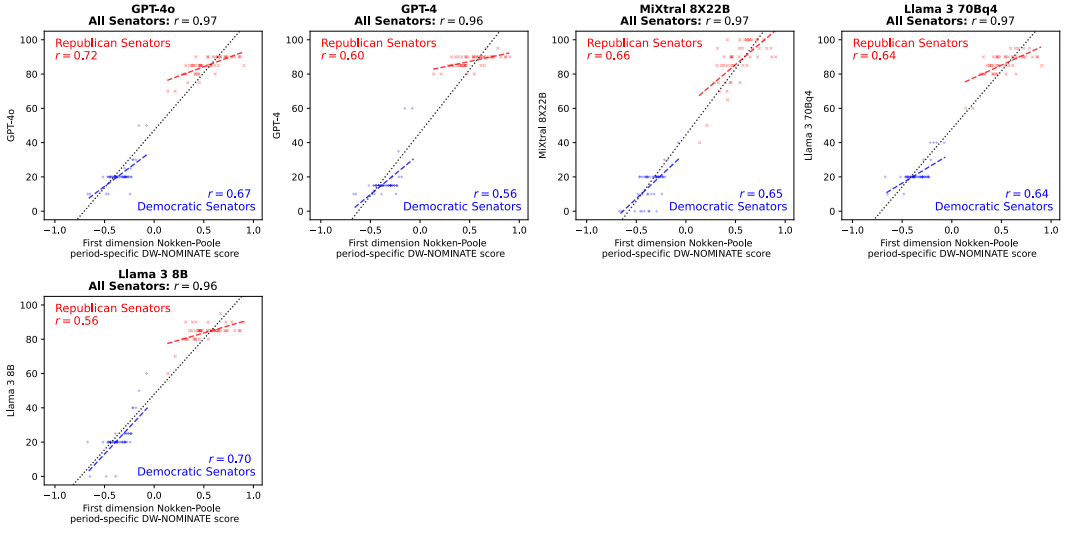

Figure A9. Replication of Figure 2 by submitting sets of tweets a single prompt. Each dot represents a senator ('+': Democrats, 'x': Republicans, 'o': others). The Y-axis reports, for each senator, the position estimates of senators based on random samples of 100 of their tweets published during the Congress session. The X-axis reports the first dimension Nokken-Poole period-specific DW-NOMINATE score.

Appendix 7. Additional Results - British Party Manifestos

Appendix 7.1 Performance of position estimates obtained by including descriptions of policy dimensions in the prompts submitted to the LLMs.

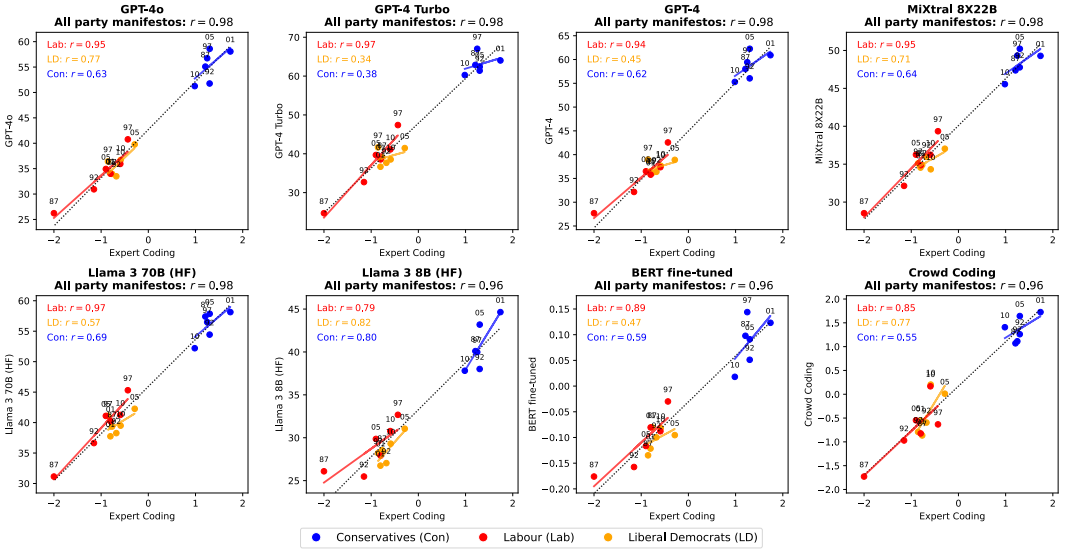

Figure A10. Positioning British party manifestos on the Economic policy dimension (left to right wing scale, partial replication of figure 3). A description of the policy dimension and of its extremities as similar as possible to the instructions given to crowdworkers by Benoit et al. (2016) was included in the prompts submitted to the LLMs (see Appendix 3.5).

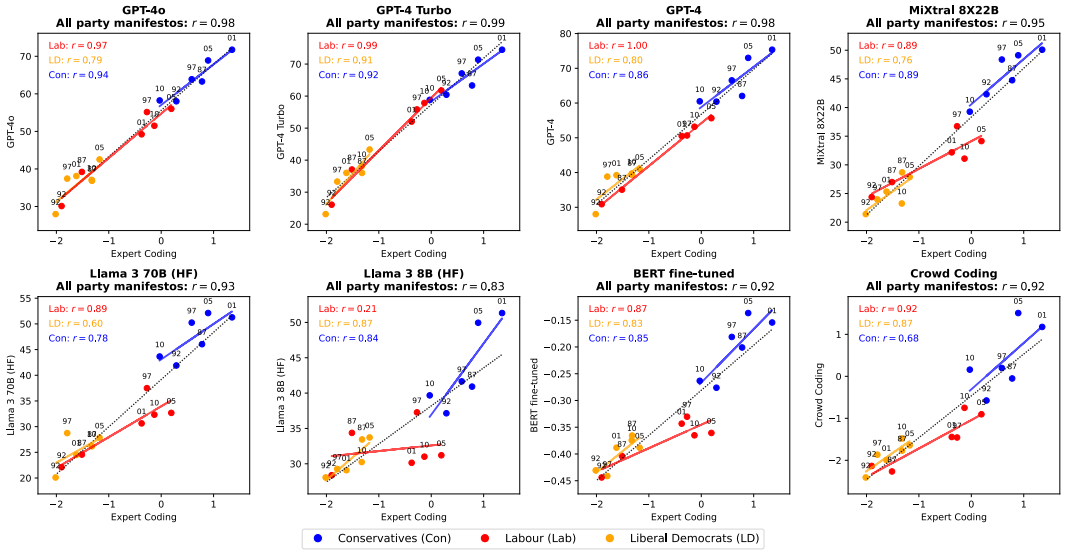

Figure A11. Positioning British party manifestos on the Social policy dimension (liberal to conservative scale, partial replication of figure 4). A description of the policy dimension and of its extremities as similar as possible to the instructions given to crowdworkers by Benoit et al. (2016) was included in the prompts submitted to the LLMs (see Appendix 3.5).

Appendix 7.2 Performance of position estimates obtained by submitting the full text of manifestos in a single prompt to the LLMs.

In the body of the article, we report the results obtained by submitting each sentence of the party manifestos to the LLMs, mimicking the approach adopted by Benoit et al. (2016) with the crowds of experts and workers. This method is somewhat slow (it takes about one or two seconds to obtain a position estimate for each sentence, and sometimes longer, for large LLMs executed locally on our laptop). Therefore, we also experimented with an approach that consists in submitting the full text of the manifesto in a single prompt. Among the LLMs we have used, the only ones that have a context window long enough to accept the manifestos in one prompt are GPT-4o, GPT-4-Turbo, and MiXtral 8X22B.

We used a prompt similar to that used for obtaining position estimates of sentences, slightly adapted to the fact that we did not ask for the position of a sentence.

> You will be provided with a text from a party manifesto.
>
> ≪ Text from the party manifesto ≫
>
> Where does this text stand on the 'left' to 'right' wing scale, in terms of economic policy? Provide your response as a score between 0 and 100 where 0 means 'Extremely left' and 100 means 'Extremely right'. To compute the score, start by first identifying the parts of the text that are about economic policy. Then, compute the score based on these parts of text. If the text does not refer to economic policy, return "NA". You will only respond with a JSON object with the key Score. Do not provide explanations.

The results are reported in figures A12 and A13. In terms of overall correlations, the results are similar to those obtained with the sentence-by-sentence scaling for economic policy, and not as good for social policy. The within-party correlations are not as good as with the sentence-by-sentence approach. We conjecture that this performance difference could come from the similarity between the LLM-based and human-based approaches: when both approaches rely on sentence-by-sentence coding, the results obtained by both approaches correlate more highly than if the two approaches

code text of different lengths. This suggests that using LLMs in a way that mimics the procedure used with human coders might lead to a higher match with the position estimates based on human coding. Further studying this conjecture is an interesting avenue for future research.

It is also worth noting that, even though GPT-4o and MiXtral 8X22B returned position estimates for all party manifestos, GPT-4 Turbo failed to do so for two manifestos (for social policy). In ancillary analyses, we observed this failure to return position estimates for long texts in other settings and with other models as well.

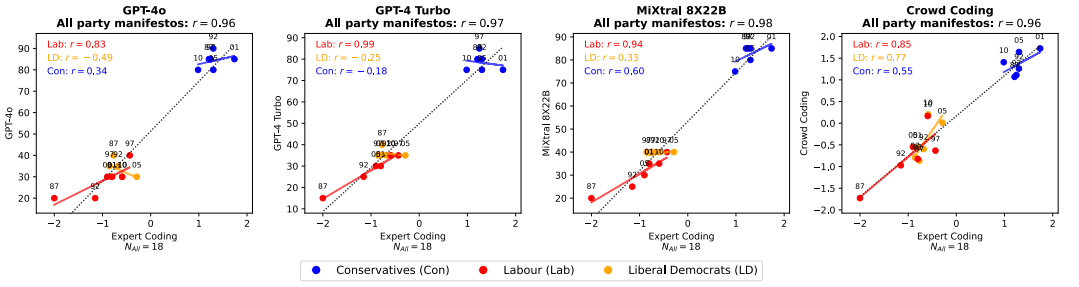

Figure A12. Positioning British party manifestos on the Economic policy dimension (left to right wing scale) in a single prompt: Pearson correlations between the benchmark (expert survey placement analyzed in Benoit et al. (2016)) and position estimates obtained with LLMs or crowdsourced position estimates reported in Benoit et al. (2016).

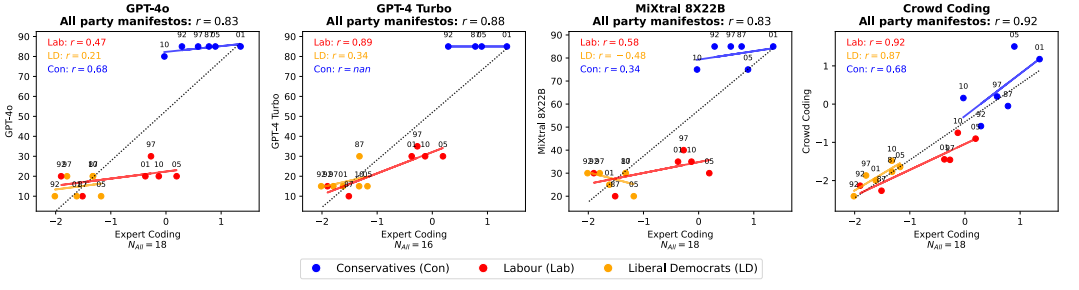

Figure A13. Positioning British party manifestos on the Social policy dimension (liberal to conservative scale) in a single prompt: Pearson correlations between the benchmark (expert survey placement analyzed in Benoit et al. (2016)) and position estimates obtained with LLMs or crowdsourced position estimates reported in Benoit et al. (2016).

Appendix 8. Additional Results - EU legislative speeches in 10 Languages

Appendix 8.1 Result summary

crowd crowd	English	German	Greek	Italian	Polish	Spanish
English	1.00	0.93	0.94	0.95	0.96	0.94
German		1.00	0.94	0.92	0.92	0.92
Greek			1.00	0.92	0.94	0.95
Italian				1.00	0.96	0.94
Polish					1.00	0.94
Spanish						1.00

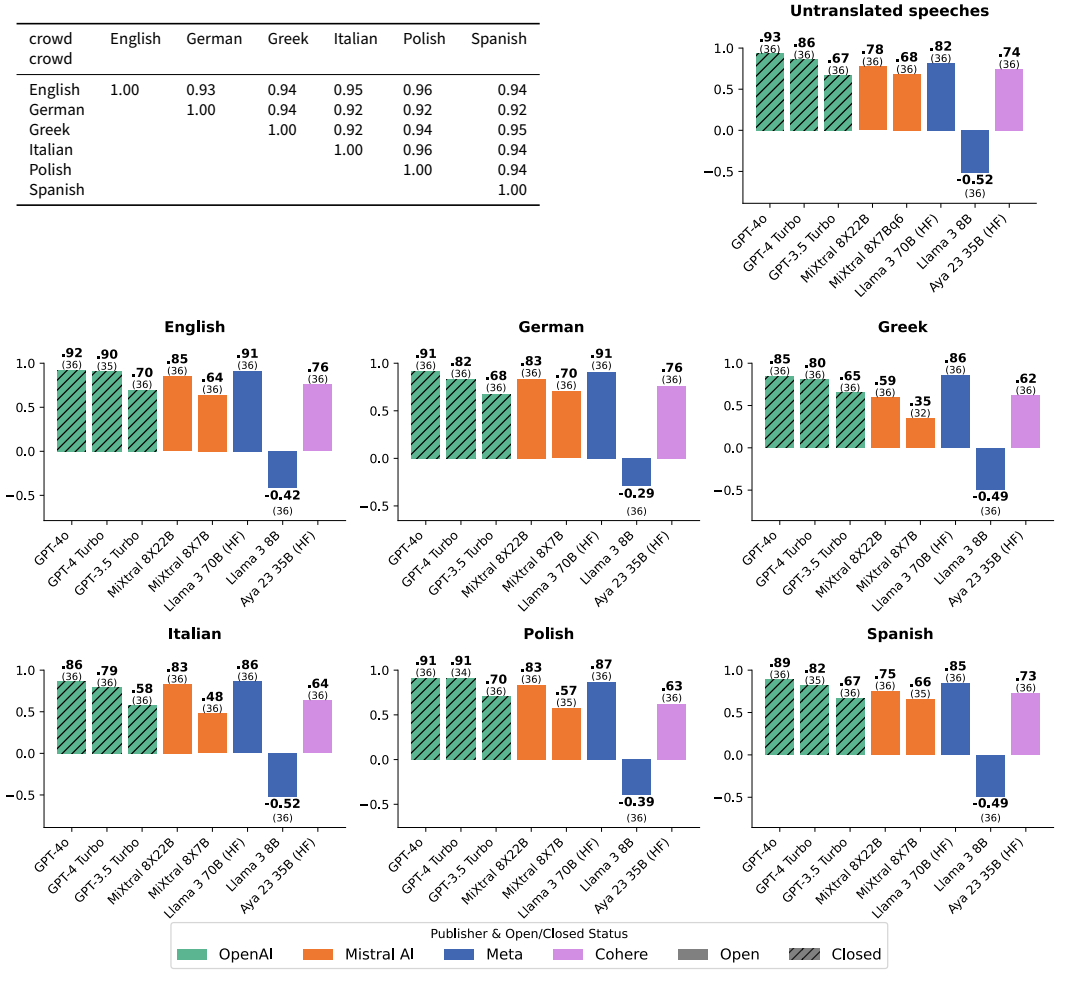

Figure A14. Positioning EU legislative speeches in 10 languages on the 'anti-subsidy' to 'pro-subsidy' dimension: Pearson correlations between the benchmark (crowdsourced position estimates reported in Benoit et al. (2016)). The numbers in parentheses indicate the number of documents for which a position was obtained. As a performance benchmark, we report the pairwise correlations between crowdsourced estimates obtained from the translations of speeches in the different languages (upper left panel). These correlations are of similar magnitude.

Appendix 8.2 Scatter plots for scaling of translations of speeches by LLMs and crowd workers

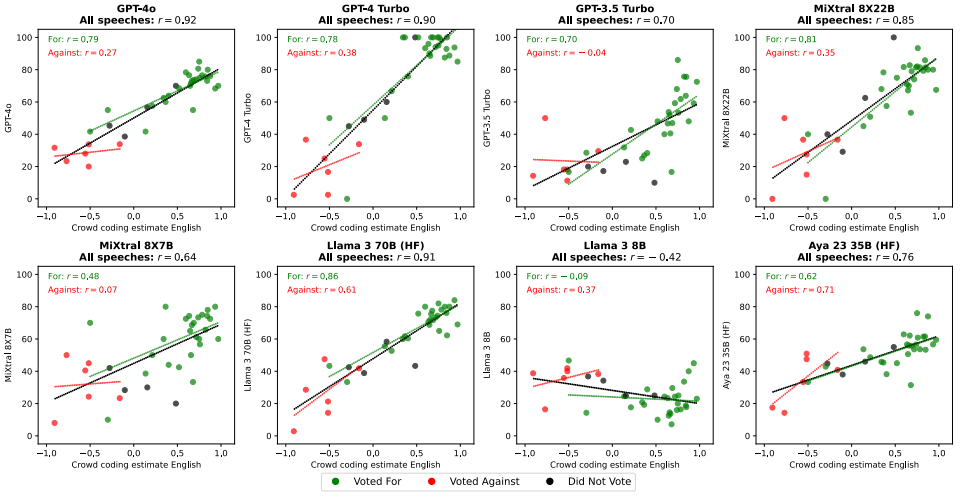

Figure A15. Positioning EU legislative speeches in 10 languages on the 'anti-subsidy' to 'pro-subsidy' dimension. Positioning using the official translation in **English**.

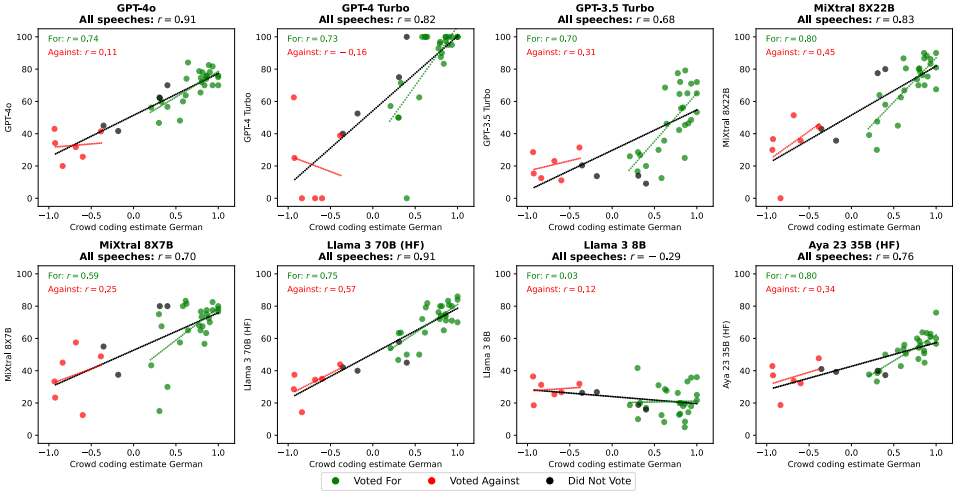

Figure A16. Positioning EU legislative speeches in 10 languages on the 'anti-subsidy' to 'pro-subsidy' dimension. Positioning using the official translation in **German**.

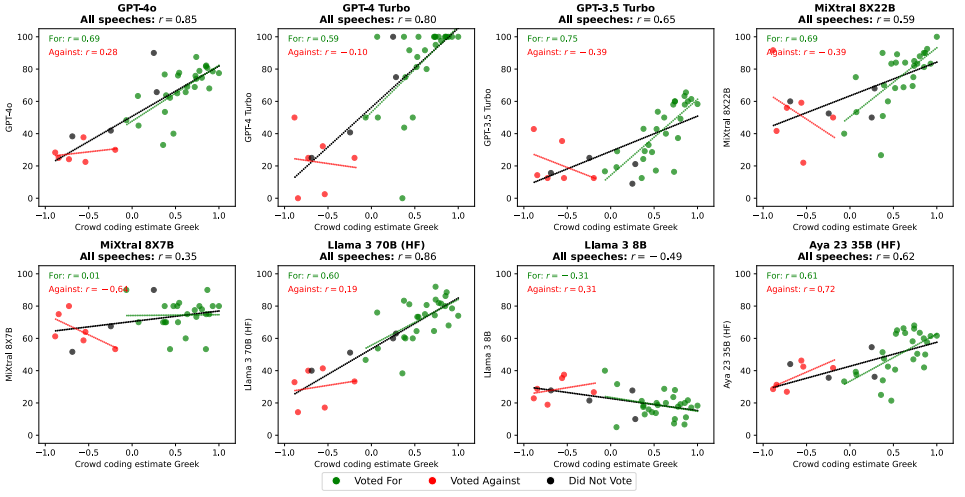

Figure A17. Positioning EU legislative speeches in 10 languages on the 'anti-subsidy' to 'pro-subsidy' dimension. Positioning using the official translation in **Greek**.

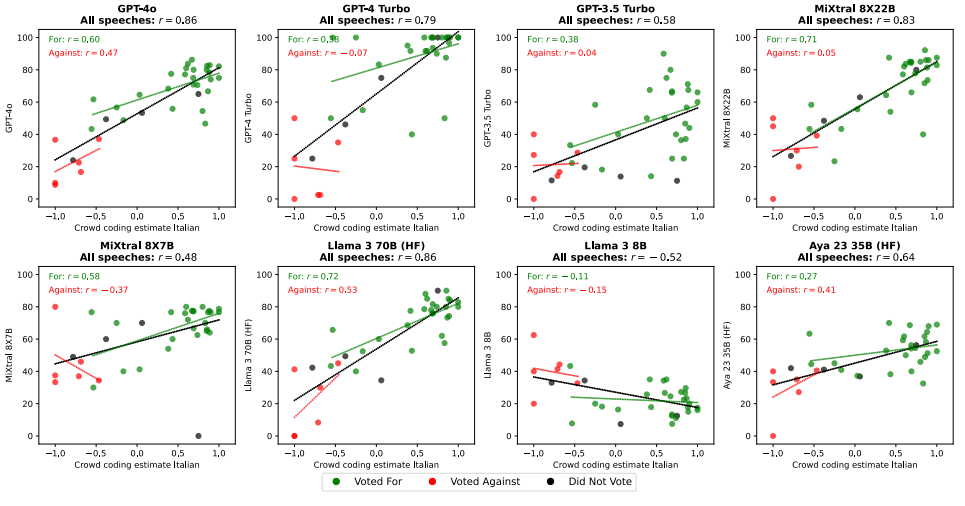

Figure A18. Positioning EU legislative speeches in 10 languages on the 'anti-subsidy' to 'pro-subsidy' dimension. Positioning using the official translation in **Italian**.

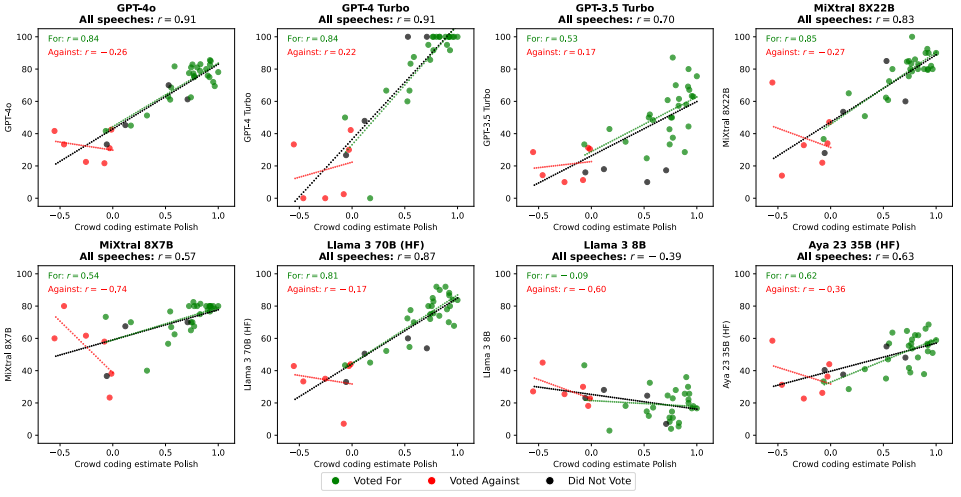

Figure A19. Positioning EU legislative speeches in 10 languages on the 'anti-subsidy' to 'pro-subsidy' dimension. Positioning using the official translation in **Polish**.

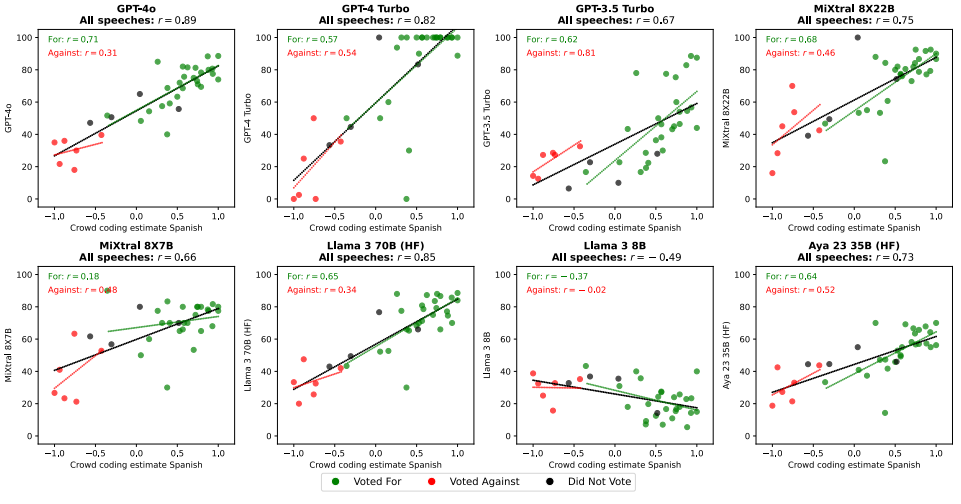

Figure A20. Positioning EU legislative speeches in 10 languages on the 'anti-subsidy' to 'pro-subsidy' dimension. Positioning using the official translation in **Spanish**.