Paired completion: quantifying issue-framing at scale with LLMs

Simon D Angus and Lachlan O'Neill

SoDa Laboratories Working Paper Series
No. 2024-02

REF

Simon D Angus and Lachlan O'Neill (2024), SoDa Laboratories Working Paper Series No. 2024-02, Monash Business School, available at http://soda-wps.s3-website-ap-southeast-2.amazonaws.com/RePEc/ajr/sodwps/2024-02.pdf

PUBLISHED ONLINE

16 June 2024

Paired completion: quantifying issue-framing at scale with LLMs

Simon D. Angus[*]
SoDa Laboratories & Dept. of Economics
Monash Business School;
Data Futures Institute, Monash University
simon.angus@monash.edu

Lachlan O'Neill[*]
SoDa Laboratories
Monash Business School
lachlan.oneill@monash.edu

Abstract

Detecting and quantifying issue framing in textual discourse - the slant or perspective one takes to a given topic (e.g. climate science vs. denialism, misogyny vs. gender equality) - is highly valuable to a range of end-users from social and political scientists to program evaluators and policy analysts. Being able to identify statistically significant shifts, reversals, or changes in issue framing in public discourse would enable the quantitative evaluation of interventions, actors and events that shape discourse. However, issue framing is notoriously challenging for automated natural language processing (NLP) methods since the words and phrases used by either 'side' of an issue are often held in common, with only subtle stylistic flourishes separating their use. Here we develop and rigorously evaluate new detection methods for issue framing and narrative analysis within large text datasets. By introducing a novel application of next-token log probabilities derived from generative large language models (LLMs) we show that issue framing can be reliably and efficiently detected in large corpora with only a few examples of either perspective on a given issue, a method we call 'paired completion'. Through 192 independent experiments over three novel, synthetic datasets, we evaluate paired completion against prompt-based LLM methods and labelled methods using traditional NLP and recent LLM contextual embeddings. We additionally conduct a cost-based analysis to mark out the feasible set of performant methods at production-level scales, and a model bias analysis. Together, our work demonstrates a feasible path to scalable, accurate and low-bias issue-framing in large corpora.

1 Introduction

The advent of large language models (LLMs) has changed the landscape of NLP research. State-of-the-art LLMs are generally trained as chat-bots that can then perform most NLP tasks through a conversational, prompt-based approach where one simply asks the model for the answer. Besides unlocking previously inconceivable use cases with their logical reasoning skills and deep knowledge of the world, LLMs have demonstrated state-of-the-art performance in many domains. We focus on the particular problem of **textual alignment**, which is the problem of determining the likelihood that a person or entity who said some text a might then say some other text b. This is similar to, but distinct from, classification in that, rather than separating the texts into classes, we instead have a high-level conceptual map of views of the world ("framings" (Entman, 1993)), and wish to determine the alignment of texts to these framings relative to other (potentially overlapping) framings.

We propose that the current conversational, prompt-based approaches to using LLMs for traditional NLP tasks (i.e. "asking the LLM") are sub-optimal when other, more direct methods are available. We demonstrate an example of this experimentally through 192 independent experiments across several methods, including traditional NLP, contextual embeddings, conversational/prompting approaches, and a novel method called "paired completion". We demonstrate that, at least in this problem domain, practitioners are generally better off using large language models as direct, probabilistic *language models*, rather than relying on chat- and instruct-based fine-tuning.

Paired completion takes advantage of the log-probability (logprob) outputs of an LLM[1] to find conditional probabilities of a text given a series of conditioners from different conditioning sets.

[*]Both authors contributed equally to this work and are listed alphabetically.

[1]Note: logprobs are available as outputs through the OpenAI API for "babbage-002" and "davinci-002" (OpenAI, 2024a), and can be gathered by running the "vLLM OpenAI-compatible API" (Kwon et al., 2023) on a local machine, for a wide variety of open-source models.

We use the relative differences in probabilities to establish a baseline metric that (at least theoretically) is resilient to the model's prior probabilities of both the conditioning text and the text being aligned to the conditioning sets. We demonstrate empirically that this method is successful, and that one achieves superior performance from using this method with raw base models compared to "asking" instruct-fine-tuned AI-models the "question" at hand.

We conduct rigorous evaluation of our proposed method by comparing it to four framing classification approaches over three diverse, synthetic textual datasets, including two baseline approaches (traditional tf-idf vectors (Sparck Jones, 1972; Salton, 1983), and fasttext sentence embeddings (Bojanowski et al., 2017)) and three LLM-based methods (contextual embeddings (Peters et al., 2018; Devlin et al., 2018; OpenAI, 2024b) LLM chat token probabilities (Radford et al., 2018, 2019), and our novel *paired completion* method). We demonstrate that the LLM-based approaches are, in general, far superior to the alternatives. The LLM-embedding approach is powerful with enough training data, but with small amounts of data (e.g. five sentences for each conditioning set) the LLM methods easily outperform -LLM-embeddings. We also demonstrate that paired completion with LLMs is generally superior to the LLM prompting approach. We discuss why this might be the case in Section 2.1.1, from a theoretical perspective. We also conduct a cost comparison analysis at current gated API pricing to assess any trade-offs in performance.

1.1 Contributions

We introduce paired completion as a solution to the problem of textual alignment. We construct a series of high-quality synthetic datasets using a novel method which captures nuances of discourse on complex topics, and use these datasets to evaluate the performance of several approaches to textual alignment. We demonstrate that paired completion is a novel, efficient, and more effective method for performing textual alignment (compared to a chat-based LLM baseline).

1.2 Related Literature

Traditional NLP classification techniques have been in use for decades. We consider two variants of such "traditional" techniques (where traditional techniques are those which were used before the advent of Large Language Models). We consider the class of methods which seek to transform text into a vector of information, in some sense, which can then be fed into a machine learning model such as a logistic regression (Hosmer Jr et al., 2013) to create a classification model.

One traditional approach we consider applies frequency counting through the TF-IDF measure (Sparck Jones, 1972; Salton, 1983) to vectorize the texts, and then feed the vectors into a logistic regression model (Hosmer Jr et al., 2013) to create a simple baseline classifier.

One can alternatively generate these vectors by using word embeddings (Mikolov et al., 2013), which assign words with a vector in some n-dimensional vector space where (ideally) dimensions correlate to some sense of semantic meaning. We use the FastText embeddings (Mikolov et al., 2013), through their Python library. To create document embeddings, we use their sentence embedding method, which performs an averaging operation on the word vectors to create an overall sentence vector[2].

Whereas word embeddings just discussed encode words first in high-dimensional vector space, which are then averaged across words in a sentence, in contrast, **contextual embedding methods** such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2018), and more recently, OpenAI's "text-embedding-3-small" and "text-embedding-3-large" (OpenAI, 2024b), are contextually-aware, meaning they take an *entire document* into account when creating the embedding vector. While the specifics of OpenAI's embedding implementation is unknown, the introductory blog post refers to Matryoshka Representation Learning (Kusupati et al., 2022). We make use of these contextually aware embeddings in a similar way to TF-IDF and word-embeddings as inputs to a logistic regression model for text classification.

The recent advance of **Large Language Models (LLMs)** stems from the development of sequence-to-sequence models (Sutskever et al., 2014) and the Transformer (Vaswani et al., 2017) in particular, which makes use of the attention mechanism. BERT (Devlin et al., 2018), and its derivatives such as ALBERT (Lan et al., 2020) and RoBERTa (Liu

[2]Note that the details can depend on the model used, as discussed here `https://github.com/facebookresearch/fastText/issues/323#issuecomment-353167113`. We use the default model "cc.en.300.bin", which is a "CBOW with position-weights in dimension 300" as per `https://fasttext.cc/docs/en/crawl-vectors.html`.

et al., 2019) continued to introduce new techniques and analyze training methods for such models. The Generative Pre-Training model architecture (Radford et al., 2018, 2019)(i.e. GPT) is a decoder-only Transformer architecture that has become the de-facto standard. All LLMs in our experiments use a GPT-like architecture, albeit with some advances and optimisations, such as Mixture of Experts (Jacobs et al., 1991), and improvements in training methodologies such as Chincilla-optimal training (Hoffmann et al., 2022).

One common measure of the capability of an LLM is **perplexity** (Jelinek et al., 1977), which is a statistical measure of the model's "surprise" at a given completion under the logic that a model which is less surprised by correct answers is better (similar to the maximum likelihood principle). The paired completion approach developed in this work is a measure similar to perplexity, but instead of seeking the estimated likelihood of a particular completion we instead calculate and compare the likelihoods of multiple completions of a given text.

2 Textual Alignment & Paired Completion

In broad terms, we define the "textual alignment" between two texts as a measure of the likelihood (in some sense) that the two texts might be spoken by the same entity. This implies the statements come from the same theoretical outlook, model of the world, and/or causal structure. It is important that the expressive entity is generally defined. For we will be, at times, leveraging generative AI LLMs to play the role of $\mathfrak{E}$, alongside human expression, to quantify the degree of alignment.

Definition 1 (Textual alignment)**.** *Given two conditioning texts a and b, and an expressive entity, $\mathfrak{E}$ (e.g. a person, a generative AI LLM), text x is said to be more textually aligned with a versus b if it is more likely that x would be expressed by some $\mathfrak{E}'$ who previously expressed a, than the alternate case where $\mathfrak{E}'$ had previously expressed b.*

Importantly, Def 1 is not the same as *similarity*. Consider the texts, 'Getting a dog will improve your life' and, 'Getting a dog will ruin your life'. Whilst these are very similar (in fact, an LLM-powered contextual similarity score would be close to 1 for these texts), they are not *textually aligned*. If someone holds the view that dogs *improve* your life (framing A), it is highly unlikely that they would say that dogs *ruin* your life (framing B). Yet these texts are highly similar on sentiment (both are neutrally posed) and share an almost identical vocabulary. However, consider the third text, 'Pets help to keep you fit and healthy'. It is clear that this text is strongly textually aligned with framing A, but strongly dis-aligned with framing B. Yet, this text is perfectly dissimilar in vocabulary, and is of middling similarity in an LLM-powered contextual embedding space. These examples demonstrate that issue-framing, formalised as *textual alignment*, is both 'simple' for a human to perceive, yet difficult for existing computational methods (based on similarity, sentiment, vocab, embeddings) to detect.

As such, we desire a new set of tools to *quantify textual alignment*. We consider these tools in the context of the "Issue-Framing" task, where a user wishes to detect and quantify texts from a large corpus which share the same framing, via textual alignment. Suppose the user has a small set of texts which together lay out a given framing position A, as compared to an opposing framing position B with a similar number of texts. We then formalise this task as follows:

Definition 2 (The Issue-Framing Task)**.** *Given a corpus of texts X (target texts) and a set of priming (or framing) texts $S = \{A, B\}$ comprising texts which represent framing A and B, for each $x \in X$, quantify the textual alignment towards A and B.*

Naturally, the user could accomplish this task by reading every text in X and marking (labelling) whether the text is textually aligned with the conditioning or framing texts from A or B. However, the aim of our work is to develop methods that might reliably accomplish this task at scale in an automated manner.

We conjecture that LLMs are well suited to performing the issue-framing task since, with the advent of attentional transformer technology, they have been shown to be remarkably successful at modelling human language. That is, forming a deep abstract representation of the meaning of human communication. Precisely the capability we require to assess textual alignment.

2.1 Paired Completion

We propose the "paired completion" method as a solution for the textual-alignment definition given above. Figure 1 gives an overview of its components. Given some set of target texts on a given topic we wish to analyse, and a small set of texts which provide frames for perspective A and B on a given topic (e.g. 'get a dog' vs. 'don't get a dog'), we construct a pair of prompt sequences, $s_1 + x$ and $s_2 + x$ to pass to a generative LLM. Each prompt sequence is composed of a random selection from one of the priming sets (e.g. s_1 'get a dog'), followed by the target text (x).

For example, a prompt sequence could be '[priming text from A, s_1] Owning a dog will improve your life. [target text, x] Dog owners have lower blood pressure and less stress in general.' A similar sequence would be created for the same text x with priming text(s) from set B. Each prompt sequence is then passed, one at a time, to a generative LLM, and instead of seeking a completion (i.e. generating new tokens) from the LLM, we instead exploit many LLM's ability to provide log-probabilities (the log of the likelihood that the model would have chosen that token/word next) for each token passed to the language model *as if it had generated this exact sequence of text*. By so doing, we generate two conditional log-probabilities, $lp(x|s_1)$ and $lp(x|s_2)$ (see details in sec 2.1.1), the conditional log-probs of x being the completion to the priming sequence s_1 and s_2 respectively.

In this way, we are leveraging the twin features of LLMs: first, that LLM attentional mechanisms are highly adept at representing the latent semantic state of a given text; and second, that LLMs have been trained to provide *coherent* sequences of text (i.e. to avoid *non sequiturs*). Together, the priming sequence will set the LLM on a particular statistical trajectory to keep the framing state consistent, which implies that if x is within this trajectory (i.e. x is textually aligned with the priming state), the summed log-probabilities the LLM assigns to the words in x will be high. Whereas, if x appears to contradict or speak for a different framing than the priming sequence, the log-probabilities for the words in x will be very low. It is this difference that we exploit by testing both priming sequences from A and B to then calculate the Diff metric.

To summarise, *paired-completion* leverages the 'deep' language modelling properties of LLM – their deep contextual representation of human meaning – to quantify the likelihood that a target text will follow from a given conditioning prior. As such, we conjecture that powerful LLMs that are neither fine-tuned (to a particular task or corpus) nor moderated by post-training methods to suppress some behaviours and up-promote others (e.g. reinforcement learning with human feedback, RHLF (Ziegler et al., 2020)) will be most well suited to the paired completion method. Either of these adaptations of LLMs could reasonably trade off fundamental (general) language modelling capabilities of LLMs in service of performance or safety on a particular task or in a particular context.

See the Appendix for details of the implementation of this method in evaluation.

2.1.1 The Diff Metric

Suppose we have a set of n priming sequences, $S = \{s_1, s_2, ..., s_n\}$, and a set of m target sequences $X = \{x_1, ..., x_m\}$. We wish to find the relative alignment, in some sense, of the elements within X towards the different priming sequences in S.

We define the diff metric as follows:

$$\Delta(s_1, s_2, x) = \text{lp}(x|s_1) - \text{lp}(x|s_2)$$

Note that $\Delta(s_1, s_2, x) = -\Delta(s_2, s_1, x)$.

The diff metric Δ describes the difference between the conditional probability of sentence x after priming sequence s_1 and the conditional probability of sentence x after priming sequence s_2. In practice, we calculate the prior probabilities of all priming sequences $s \in S$ as p_s, and all texts in $x \in X$ as p_x, and the probability of a concatenated string $s + x$ as $p_s x$. Note that concatenation is not necessarily simple string concatenation, but rather ensures grammatical correctness - there is no perfect way to do this, but we found that just ensuring grammatical correctness seems to work sufficiently well in practice.

We then compute $\text{lp}(x|s) = p_{sx} - p_s$ to find the conditional probability of x. We can compare this to the prior probability p_x to determine whether the presence of s has made x more or less likely, and we can compute $\text{lp}(x|s_1) - \text{lp}(x|s_2)$ (i.e. the Δ metric). Since a larger logprob indicates a higher probability, Δ will be positive if x is more likely after s_1 than after s_2, and negative if x is less likely after s_1 than after s_2. Because LLMs (and language models in general) might assign different prior probabilities to both the conditioning sentences s and the alignment text x, any such method must be robust to priors. This is why we use the *difference* in conditional probabilities of the same text with different prompts, which is robust to the prior probabilities of both s and x.

One interpretation of this approach, with reference to Def. 1, is that the LLM performs the role of the expressive entity $\mathfrak{E}$, and so provides a quantification of the likelihood that the text x follows

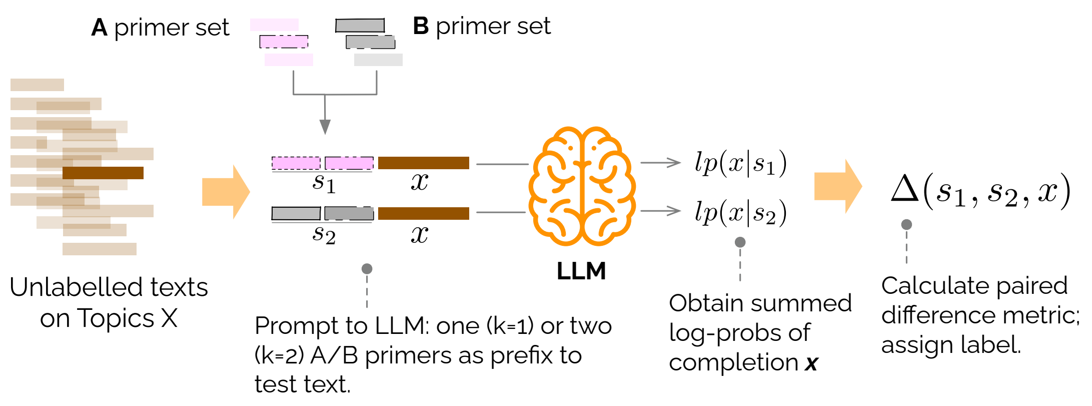
A primer set
B primer set
Unlabelled texts on Topics X
s_1
x
s_2
x
LLM
Prompt to LLM: one (k=1) or two (k=2) A/B primers as prefix to test text.
$lp(x|s_1)$
$lp(x|s_2)$
Obtain summed log-probs of completion *x*
$\Delta(s_1, s_2, x)$
Calculate paired difference metric; assign label.

Figure 1: **Narrative classification with the LLM Paired completion method.** Texts are taken, one at a time, as *completions* to one ($k = 1$) or two ($k = 2$) priming conditioner sentences from two opposing issue framing sets, A, B, in turn. Each conditioner is chosen randomly from the respective set when used, i.e. two conditions from set A, followed by x, or two conditions from set B, followed by x. Instead of requiring a *generative* action from the LLM the summed log-probabilities of the completion text (x) are obtained from the LLM *as if* the LLM had used the text to follow the conditioners. The two resultant summed log-probs inform the $Delta$ metric.

text s_1, versus following text s_2, i.e. we obtain a measure of *textual alignment*.

Since the core idea of paired completion is to use the priming/conditioning sequence to statistically deflect the LLM towards the given framing (and so, measure the model's degree of 'surprise' with the completion text) we conjecture that a longer conditioning sequence may lead to improved accuracy in classifying and retrieving texts aligned with a given framing. To explore this possibility we test two treatments, with either one ($k = 1$) or two ($k = 2$) priming/conditioning text(s) being used. Implementation details are provided in the Appendix.

3 Synthetic Dataset Formation

Synthtic data was chosen for evaluation for two reasons. First, non-synthetic examples of texts drawn from a given framing perspective lack scientific control with a variety of unobserved characteristics potentially driving outcomes. Second, we need that the LLMs have not seen the evaluation samples in their training, else, completion logprobs could reflect familiarity rather than textual alignment.

The synthetic dataset generation pipeline takes a topic (e.g. "dog ownership", "climate change", etc.) and produces a corpus of sentences that reflect different perspectives on the topic. The generation process is a two-step hierarchical process where we generate seed perspective and then generate sentences that align with each perspective. We also generate distillations (into a smaller number of sentences, e.g. 5), summaries, and simple names for each side, with each of these generated from the seed dataset (and having no knowledge of the sentences generated thereafter).

The "dog ownership" corpus was designed (and later proved) to be a straightforward classification problem, for which most LLMs would find success. The other corpora ("climate change", "domestic violence", and "misogyny") reflect more subtle, controversial, and/or otherwise difficult issues to discuss, and are much more likely to be affected by the alignment process used to train the LLMs. These indeed proved more challenging for the models to classify. However, we did not experience outright prompt refusal in our generation experiments.

4 Evaluation Approach

We compare the novel paired completion method with two other LLM-based approaches: a prompt-based approach with instruct/chat fine-tuned models (described next), and contextual embeddings (described earlier).

4.1 LLM Prompting

Starting with a corpus of texts to test, we construct a prompt with three components: 1) a static instructional component which provides the LLM with the task information; 2) a set of context texts that represent framing A and B to be tested (A, B); and 3) a single target text (x). Unlike in LLM paired completion, we do not require the LLM to provide log-probs for the input sequence, but instead, we obtain the log-probs of the first two tokens produced by the LLM in response to this prompt, i.e. the first two generated tokens. Note that, by virtue of the constraints in the prompt, these probabilities include the log-probs for both response A and B. We extract the probability of the first token of the label assigned to A (e.g. '[**equality**]' [1 token]), and B (e.g. '[**mis**][og][yny]' [3 tokens]), respectively. With this information we can both identify which set the LLM has assigned the text to (based on the higher probability of its tokens) and calculate the equivalent Diff metric, $\Delta(A, B, x)$.

We use a fixed prompt, which was initially fine-tuned for GPT-4 and GPT-3.5, and then further tuned for Mixtral-8x7b-Instruct-v0.1 and LLaMA-2-70B-Chat. In hindsight, it was a mistake to tune our prompts for GPT-4 first, as while GPT-4 was very likely going to give the best performance on the tasks at hand (compared to the other models in consideration), it was also a lot more forgiving of errors, confusing wording, and conflicting instructions within the prompt. It is also possible this somewhat biased the prompts towards the OpenAI models, but this was unavoidable given our preliminary results (not included in this paper) were gathered using only the OpenAI APIs; in any case, our results demonstrate the superiority of the open-source models on these tasks. We used a single prompt across all models in our final experiments.

4.2 Performance Analysis

In terms of true-positives (TP), false positives (PF) and false negatives (FN), the F1 score is calculated as,

$$f1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \ .$$

The f1 score takes a value from 0 to 1, and will be equal to 1 when the method perfectly identifies all the 'As' in the data, and does not mis-identify any 'Bs' as 'As'.

Confidence intervals (95%) for f1 scores were either calculated directly from replicates, in the case of the logistic regression methods (TF-IDF, word- and LLM- embeddings), and using bootstrapping in the LLM API cases (100 replicates, 1000 samples).

4.3 Summary of experiments

Together, across the five methods, four topics, and related variants, 192 experiments were conducted, as summarised in Table 1.

5 Results & Discussion

Our experiments demonstrate strong performance across the board for both prompt-based and paired completion methods, as shown in Figure 2. Paired completion methods tend to statistically perform the same or better than prompt-based methods. This section includes a broad summary of results. More detailed results, tables, and discussion can be found in the appendices.

5.1 Comparative Analysis of Classification Methods

With sufficient data (200+ samples), the embedding approach was competitive with GPT-4 prompting. However, embeddings performed significantly worse in few-shot learning contexts. Among LLM instruct models, GPT-4-Turbo outperformed all other models. GPT-3.5-Turbo, Mixtral-8x7b-Instruct-v0.1, and LLaMA-2-70B-Chat had similar performance, with LLaMA-2-70B-Chat having the highest propensity for failure modes. For the paired completion approach, performance trended with model parameter count, with LLaMA-2-70B performing best, followed by Mixtral-8x7b, davinci-002, and babbage-002. This consistency may occur because paired completion is less sensitive to model-specific factors like architecture, alignment, and fine-tuning.

The three methods for interaction with LLMs that were analysed are all effective, and uniquely suited to different scenarios. The paired completion approach proved highly effective, efficient, and robust to model-specific influences. Embedding-based methods are extremely cheap due to a combination of cheap models and fewer calls to the APIs, and proved very effective with sufficient data. However, this data threshold was far beyond the five exemplars used for the other two approaches, and performance suffered greatly when using 50 exemplars (which is still an order of magnitude more than the five exemplars provided to the other two methods). Prompt-based completion proved effective, particularly with GPT-4 (which does not support the logit outputs required for paired completion), but when possible we generally found paired completion to be more cost-effective than the prompting approach. On the other hand, the prompt-based method is very straightforward, and as models increase in capability (and, ideally, increase in cost-performance as well) the cost-performance distinction between these two techniques might diminish.

Method	Models	Topics	Variants	Total
Log-Reg :: TF-IDF	1	4	6	24
Log-Reg :: FastText	1	4	6	24
Log-Reg :: LLM Embeddings	2	4	6	48
LLM Paired Completion	4	4	2	32
LLM Prompting	4	4	4	64
TOTAL				**192**

Table 1: **Summary of Experiments** The same four topics were tested across all configurations ('dog-ownership', 'climate-change', 'domestic violence', 'misogyny'). For each Log-Reg (Logistic Regression) style experiment where a logistic-regression model was trained on a training sub-set of the data, 6 different sub-set sizes were used ($n \in \{10, 20, 50, 100, 200, 500\}$). For LLM Paired Completion two variants for the number of conditioners were used ($k \in \{1, 2\}$). For LLM Prompting, 4 prompt variants were used (seeds, distilled, summary, zero-shot).

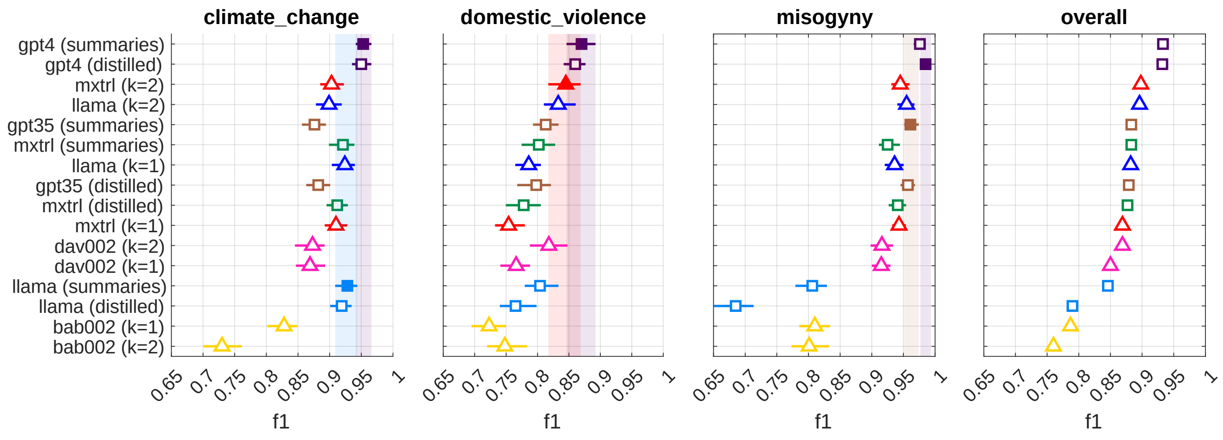

Figure 2: **F1 outcomes across LLM prompting (□) and paired-completion (△).** Filled markers indicate approaches that are statistically similar to most performant method. Semi-transparent shading shows 95% confidence interval for these methods to indicate other methods which provide similar performance to performant models. Ranking is by overall performance. See appendix for performance comparison with log-reg classification methods.

5.2 Cost vs. Performance

An analysis of the cost-performance trade-off for the LLM methods (Figure 3) reveals that the paired completion approach with LLaMA-2-70b and Mixtral-8x7b is very cost-effective for their level of performance. While GPT-4 had the best overall performance, it was also by far the most expensive. Other configurations can be chosen based on requirements and funding availability. All LLM-based approaches were significantly more expensive than the embedding approaches, which require more data but proved competitive given sufficient training examples.

5.3 Model Bias

We observed differences in the bias displayed by models and techniques that were dataset-dependent (Figure 4). Embedding-based approaches appear most robust to bias, with no statistically significant bias found for any configuration. LLM-based approaches demonstrated bias in some scenarios, with the $k = 2$ paired completion configuration potentially reducing bias compared to $k = 1$. The top performing LLM paired completion methods (mxtrl-k=2; llama-k=2) show significantly less bias than the top LLM prompting approaches, including GPT-4. Further studies are needed to examine the sources of these biases, such as bias in training data, language modeling, or alignment. However, the results suggest the stronger LLM paired completion methods (e.g. llama-k=2) achieve a balance of high accuracy and low bias.

6 Limitations & Further Work

One advantage of the paired completion approach is that it can easily be **extended to more than two narrative sets with only a linear increase in the number of model calls**, rather than the quadratic one might expect. The time complexity of the comparison algorithm after calling the model is possi-

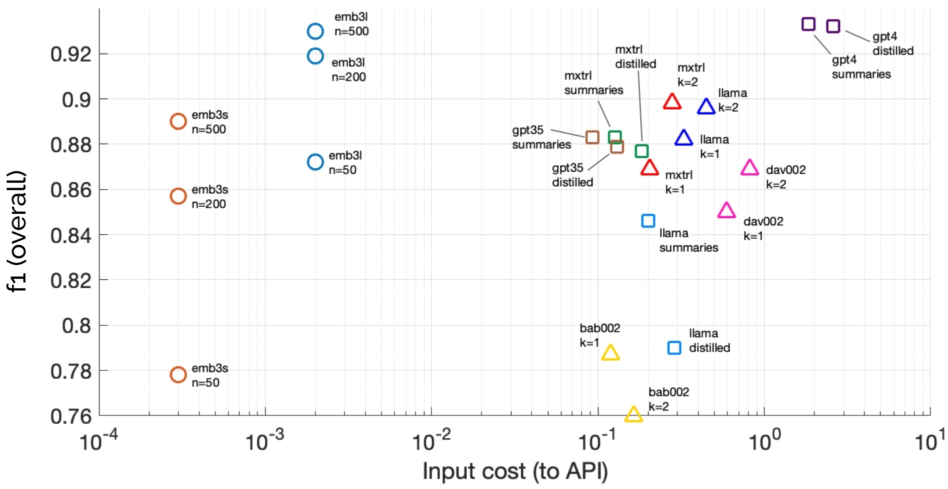

Figure 3: **Cost — performance trade-off for LLM methods.** Colouring and styling follows performance figures in the rest of the paper. Model short name and variant are provided for clarity. Note: logarithmic scaling is applied to the x-axis to handle the five orders of magnitude difference betweeen LLM embedding approaches and GPT-4 prompting approaches.

bly superlinear in the number of classes (though the exact behaviour would be implementation-dependent), but the number of model calls is linear in the number of classes. We thus expect the latter to be the overwhelming factor at least until reaching thousands of classes, given an efficient implementation of the comparison algorithm.

The data seems to offer some **support for the conjecture that aligned models are more prone to bias when performing framing alignment**, but we cannot make any definitive claims without significantly more evidence and data. We only used three "serious" topics (climate change, domestic violence, and misogyny); for further study, we would significantly expand this (perhaps to 10, 20, or even 100 topics, ranging across and beyond, say, the Overton window (Russell, 2006)). While out of the scope of this study, such an experiment would hopefully demonstrate correlations (or lack thereof) between the placement of ideas and framings within the spectrum of discourse and the models' bias towards or against them.

Throughout this paper, we have demonstrated that more powerful models are more capable of these tasks and that the performance differences can be quite distinct. One might invert this line of thinking and **use performance on this task as an evaluation benchmark for large language models**. Note that this is different to perplexity, which is an absolute measure of how "surprised" the model is by *the (or a) correct completion*, with the intuition that a good model should be less surprised by the correct answer (i.e. a maximum likelihood approach). We instead measure how "surprised" the model is by each text within a set of completions relative to one another (even if, as a whole, the model finds the set of completions very likely or unlikely).

The number of possible experiments also increases quadratically with the dataset size, rather than linearly as with perplexity.

In our experiments, we ran LLMs on labelled classification corpora designed to mimic lines of human discourse across a particular topic, but we only used the accuracy (and similar measures), rather than more **deeply analysing the failure modes of the model(s)**. By performing such an analysis, either with humans or with LLMs directly, one might better understand "why" a given model tends to make certain kinds of mistakes. A meta-analysis with an LLM (perhaps even the same LLM) could facilitate a level of introspection from the models, where they look at their experimental performance and note their biases (which can perhaps be incorporated into further prompts or fine-tuning efforts).

Acknowledgments

This work was supported by funding from the Paul Ramsay Foundation, under the 'narratives of disadvantage' project.

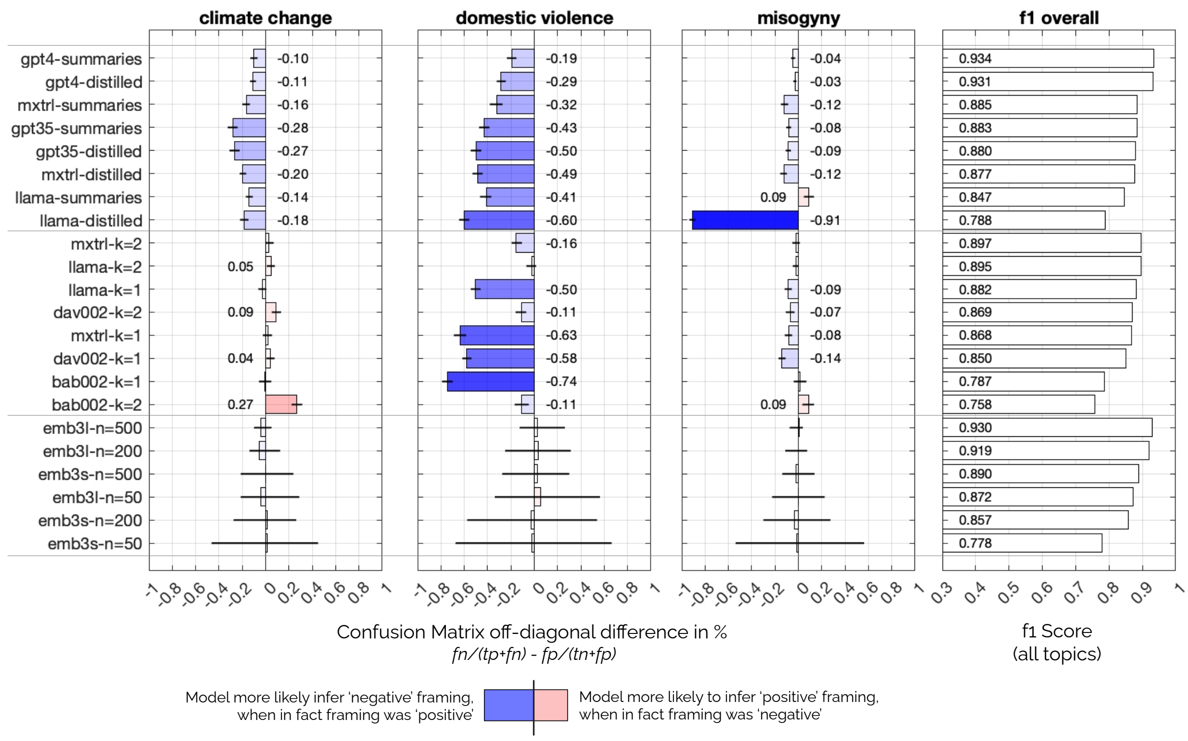

Figure 4: **Issue-framing classification error asymmetry ("bias") across LLM methods and topics.** The first three panels (left to right) give mean and (95% confidence intervals) for issue-framing asymmetry, or model bias, calculated as the difference between the off-diagonals in a normalised confusion matrix. A method which makes the same proportion of mistakes when classifying texts from known framing set A or B should score 0 on this metric. Significant departures (i.e. where we cannot reject H_0:that no asymmetry exists, no score is given). Methods are presented in blocks from top to bottom through LLM-Prompt, LLM-Paired completion, and LLM Embeddings. The final (right-most) panel provides F1 overall scores for comparison.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Robert M Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of communication*, 43(4):51–58.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied logistic regression*. John Wiley & Sons.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. 2022. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

OpenAI. 2024a. GPT base.

OpenAI. 2024b. New embedding models and API updates.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Preprint*, arXiv:1802.05365.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nathan J Russell. 2006. An introduction to the overton window of political possibilities. *Mackinac Center for Public Policy*, 4.

Gerard Salton. 1983. Introduction to modern information retrieval. *McGraw-Hill*.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-tuning language models from human preferences. *Preprint*, arXiv:1909.08593.

A Appendix

A.1 Comparative Analysis of Classification Methods

A.1.1 Traditional Classification

Both the tf-idf and fasttext-based classification methods showed inferior performance to the LLM-methods. The tf-idf approach generally performed better than the fasttext approach, indicating that on these datasets a keyword approach is superior to a summed embedding vector approach. This does not include the contextual embeddings, which are more complex than the simple summation performed by fasttext when embedding a sentence. The superiority of LLM-based approaches was, of course, expected, and these traditional methods were included to provide a baseline for performance.

A.1.2 Contextual Embeddings

The significant improvement in the performance of the contextual embedding models, compared to the non-contextual fasttext embeddings, demonstrates the importance of contextuality when creating embedding vectors for text. We observed a significant uplift in the performance of contextual embeddings correlating with dataset size, far more than the uplift between any of the prompting methods (e.g. seeds vs summaries). With a large enough amount of data, generally 200 or more samples, the embedding approach approached or even exceeded the performance of GPT-4 with prompting, demonstrating the potential power of this approach. However, the embedding approaches did significantly worse in the few-shot learning contexts (e.g. with 10 samples per class, which is still double the number of examples provided to the LLMs with the distilled prompting approach).

We therefore conclude that contextual embeddings can be a good, and potentially cost-effective, method for performing classification in contexts with large amounts of training data, but they are not as suitable when there is little training data (i.e. in a few-shot learning context). A hybrid approach, where one generates a training corpus with LLM classification and then uses this to train an embedding system, might be cost-optimal, but analysis of this approach is beyond the scope of this study.

A.1.3 LLM Instruct Models

We observed general superiority from GPT-4-Turbo, the most powerful LLM model available (and the only model of its performance class to

Table 2: **F1 Score Summary Table: LLM Completion/Prompt methods.** Boldface indicates best performance ($+/-$ 0.01) in a column (including GPT-4), whilst underline indicates best performance ($+/-$ 0.01) outside of GPT-4. Note: seeds (*) and zero-shot (†) prompt variants shown in the table below, but are not presented in the main paper.

Model		Variant	F1 Score			
Version	Abbr		Climate Change	Domestic Violence	Misogyny	Overall
A. LLM Completion/Prompt Methods						
gpt-4-turbo-preview	gpt4	*seeds	0.977	0.917	0.989	0.961
gpt-4-turbo-preview	gpt4	summaries	**0.953**	**0.870**	**0.976**	**0.933**
gpt-4-turbo-preview	gpt4	distilled	**0.950**	**0.860**	**0.985**	**0.932**
gpt-3.5-turbo	gpt35	*seeds	0.935	0.868	0.986	0.930
Mixtral-8x7B-Instruct-v0.1	mxtrl	*seeds	0.952	0.838	0.950	0.913
Mixtral-8x7B-instruct-v0.1	mxtrl	k=2	0.903	<u>0.845</u>	0.945	<u>0.898</u>
Llama-2-70b-chat-hf	llama	k=2	0.899	0.833	<u>0.955</u>	<u>0.896</u>
gpt-3.5-turbo	gpt35	summaries	0.876	0.813	<u>0.961</u>	0.883
Mixtral-8x7B-Instruct-v0.1	mxtrl	summaries	<u>0.921</u>	0.802	0.925	0.883
Llama-2-70b-chat-hf	llama	k=1	<u>0.924</u>	0.786	0.936	0.882
gpt-3.5-turbo	gpt35	distilled	0.882	0.798	<u>0.957</u>	0.879
Llama-2-70b-chat-hf	llama	*seeds	0.926	0.849	0.861	0.879
Mixtral-8x7B-Instruct-v0.1	mxtrl	distilled	0.912	0.778	0.941	0.877
gpt-4-turbo-preview	gpt4	†zero-shot	0.889	0.785	0.949	0.874
davinci-002	dav002	k=2	0.873	0.818	0.916	0.869
Mixtral-8x7B-instruct-v0.1	mxtrl	k=1	0.910	0.754	0.943	0.869
davinci-002	dav002	k=1	0.869	0.766	0.915	0.850
gpt-3.5-turbo	gpt35	†zero-shot	0.880	0.766	0.891	0.846
Llama-2-70b-chat-hf	llama	summaries	*0.928*	0.804	0.806	0.846
Mixtral-8x7B-Instruct-v0.1	mxtrl	†zero-shot	0.828	0.730	0.897	0.818
Llama-2-70b-chat-hf	llama	distilled	*0.919*	0.765	0.685	0.790
babbage-002	bab002	k=1	0.828	0.723	0.810	0.787
Llama-2-70b-chat-hf	llama	†zero-shot	0.865	0.767	0.724	0.785
babbage-002	bab002	k=2	0.730	0.748	0.801	0.760

Table 3: **LLM Completion/Prompt Methods Cost Analysis.** Cost analysis of different LLM models based on their performance and token usage.

Model	Abbr	Variant	Score	F1	Tokens	$/Mil	Cost
gpt-4-turbo-preview	gpt4	distilled	0.961	0.932	260,470	10	2.6047
gpt-4-turbo-preview	gpt4	summaries	0.955	0.933	184,470	10	1.8447
llama-2-70b-chat-hf	llama	k=2	0.920	0.896	498,091	0.9	0.4483
llama-2-70b-chat-hf	llama	k=1	0.920	0.882	364,063	0.9	0.3277
mixtral-8x7b-instruct-v0.1	mxtrl	k=2	0.914	0.898	467,721	0.6	0.2806
mixtral-8x7b-instruct-v0.1	mxtrl	k=1	0.912	0.869	341,292	0.6	0.2048
davinci-002	dav002	k=2	0.892	0.869	408,862	2	0.8177
davinci-002	dav002	k=1	0.871	0.850	297,721	2	0.5954
gpt-3.5-turbo	gpt35	summaries	0.846	0.883	184,470	0.5	0.0922
gpt-3.5-turbo	gpt35	distilled	0.828	0.879	260,470	0.5	0.1302
mixtral-8x7b-instruct-v0.1	mxtrl	summaries	0.799	0.883	211,435	0.6	0.1269
mixtral-8x7b-instruct-v0.1	mxtrl	distilled	0.788	0.877	306,635	0.6	0.1840
babbage-002	bab002	k=1	0.749	0.787	297,721	0.4	0.1191
llama-2-70b-chat-hf	llama	summaries	0.742	0.846	223,457	0.9	0.2011
babbage-002	bab002	k=2	0.740	0.760	409,300	0.4	0.1637
llama-2-70b-chat-hf	llama	distilled	0.448	0.790	319,991	0.9	0.2880

Table 4: **Embedding Method Cost Analysis.** Cost analysis of embedding methods based on their performance and token usage. The embedding methods are much cheaper, but require a large source of labelled training data.

Model	Abbr	Variant	Score	F1	Tokens	$/Mil	Cost
text-embedding-3-large	emb3l	n=500	0.981	0.930	15,674	0.13	0.0020
text-embedding-3-large	emb3l	n=200	0.973	0.919	15,674	0.13	0.0020
text-embedding-3-small	emb3s	n=500	0.939	0.890	15,674	0.02	0.0003
text-embedding-3-large	emb3l	n=50	0.928	0.872	15,674	0.13	0.0020
text-embedding-3-small	emb3s	n=200	0.915	0.857	15,674	0.02	0.0003
text-embedding-3-small	emb3s	n=50	0.802	0.778	15,674	0.02	0.0003

support outputting logprobs, making it suitable for classification in our logprob-based pipeline). This was expected, as none of the other models claim parity with GPT-4, and we therefore use GPT-4 as an upper bound on performance (similar to using traditional classification as a lower bound on performance).

The other LLM instruct-capable models, including OpenAI's GPT-3.5-Turbo, Mistral's Mixtral-8x7b-Instruct-v0.1, and Meta's LLaMA-2-70B-Chat, went blow for blow throughout the experiments, though LLaMA-2-70B-Chat demonstrated the highest propensity for failure modes. We recommend trying several models to determine which is most suitable. GPT-3.5-Turbo seems to be a relatively dependable choice, and the relative reliability of the OpenAI API coupled with relatively high rate limits make it a straightforward option for running large experiments. Mixtral-8x7B generally seems to be as good as, if not better than, LLaMA-2-70B-Chat, which is in line with previous experimental results (Jiang et al., 2024).

A.1.4 LLM Paired Completion

The Paired Completion approach requires an API which supports, in OpenAI API parlance, "echo[ed]" logprobs (i.e. outputting the logprobs for input tokens). For unknown reasons, OpenAI launched and then subsequently disabled this feature on their "gpt-3.5-turbo-instruct" model, and to our knowledge have never offered it on their Chat API (only their "legacy" Completions API). However, they do support the feature on their legacy completion models, including "davinci-002" and "babbage-002". vLLM also supports the "echo" parameter through its OpenAI-compatible API, which we leveraged to get results from LLaMA-2-70B and Mixtral-8x7b-v0.1. Note that none of these models are fine-tuned for chat.

We found babbage-002 generally performed poorly compared to davinci-002, which in turn was outperformed moderately by the two open-source models. The performance trend was relatively stable, with LLaMA-2-70B performing best, Mixtral-8x7b close behind LLaMA, davinci-002 close behind Mixtral, and babbage-002 quite behind the pack.

We conjecture that this consistency in performance occurs because the paired completion approach is less sensitive to outside influences such as architectural changes that make model training easier (which are continually developed as the literature expands), alignment (via mechanisms such as RLHF), fine-tuning for instruct/chat, and the size of the datasets used for post-training tuning steps. It may also be that these problems (which were designed for use with davinci-002 and babbage-002) are too easy for the newer, more powerful models, and that more complex experiments would tease out more distinctions between them. It should be noted that the performance trend places the models in order of their number of parameters (although the parameter counts of davinci-002 and babbage-002 are only estimated, we suspect that the models are indeed placed in correlation with their parameter count).

A.2 Model Bias and Confusion Matrices

Whilst accuracy and recovery metrics speak to the overall performance of a given approach on the issue-framing task, an important additional characteristic of an approach is to ask whether, when making classification mistakes, the approach fails more often in one issue direction versus another. In other words, does the approach demonstrate *bias*?

To explore bias in the modelling approaches, we compute the difference in proportions of 'off-diagonal' confusion matrix behaviour. In a standard 2-by-2 confusion matrix approach we compare ground-truth to model inference. Model decisions which align with the ground truth (e.g. a 'pro-science' synthetic text, is assigned 'pro-science' by the method) contribute to the main diagonal of the confusion matrix. Off diagonals then capture the proportion of times that the model makes a mistake. If the model is unbiased, the frequency of mistakes made between both sides of the issue-framing debate should be similar.

To account for this, and knowing that our synthetic data has equal numbers of examples on each 'side' of an issue framing, we develop a simple difference in off-diagonal proportions: $fn/(tp + fn) - fp/(tn + fp)$, i.e the ratio of false negatives (fn) to all true positive cases, minus the ratio of false positives (fp) to all true negative cases. If no bias exists, this metric should be 0.

Figure 4 presents the results of this bias calculation across each different configuration on the three main datasets.

While we observed significant differences in performance between models and classification techniques, we also observed differences in the bias displayed by models, and even the same model with different techniques. This seems to be dataset-dependent.

In general, the embedding-based approaches appear to be the most robust to bias, with no statistically significant bias found across the three experimental datasets for any embedding-based configuration. The other two LLM-based approaches demonstrated bias in some scenarios. It may be the case that the $k = 2$ configuration of the pairwise completion method reduces bias compared with the $k = 1$ approach, but we do not have enough evidence to say this with any certainty.

However, it is clear that the more performant LLM paired completion methods (mxtrl-k=2; llama-k=2) show significantly less bias than the performant LLM prompting approaches, even GPT-4.

It remains for other studies to examine what might be driving the sources of these biases. For instance, bias can arise due to bias in training data, biased language modelling, or bias in model alignment (a refinement process applied to models like ChatGPT and GPT-4 to improve their task compliance and reduce toxicity). Nevertheless, this exercise demonstrates that, if one wishes to leverage high accuracy and low bias, the stronger LLM paired completion methods present (e.g. llama-k=2) present as likely balanced candidates.

A.3 Computational Complexity and Resource Requirements

Both the paired completion and prompting approaches increase time complexity linearly with the number of classes. The paired completion is non-comparative, in that a new set of framings can be added independently of past/future framing sets, and thus the compute scales linearly with the number of framings (though it might scale faster than linearly with the number of framings *within* the framing set if using $k > 1$, as the number of comparisons for a framing set of size n is $O(n^k)$).

The prompting approach only requires a single call to the model, regardless of how many classes are used, but the number of tokens used within the call will scale linearly with the number of classes. There will be a large constant term in the size of the input prompt corresponding to an explanation of the problem, the expected output, and the required output format, meaning that for smaller numbers of classes, the relative increase in prompt size can be small. However, the difficulty of the task also increases with the number of classes, and we conjecture that the paired completion approach will scale better to a very large number of classes, as the model only needs to "consider" one class (i.e. one priming sequence) at a time.

From a practical perspective, the OpenAI API has a seemingly little-known feature that allows calls to the Completions API to be batched (thus including multiple texts in a single API call, and receiving all the results for those texts in a single response to the API call). We found this a useful speed boost (by a factor of 20x) when using babbage-002 and davinci-002, because we were primarily rate-limited by API calls rather than tokens used. However, our experience is that other OpenAI-compatible vendors tend not to implement this feature, and it's unclear if there would be time savings from this anyway (as they might not execute the calls in parallel in their backend).

A.4 Cost

GPT-4 proved by far the most expensive model, which was the expected result and the price (in a quite literal sense) paid for its excellent performance on all benchmarks. Other models varied in cost, but as demonstrated in Figure 3, the paired completion approach with LLaMA-2-70b and Mixtral-8x7b proved very cost-effective for their performance. Trade-offs can be made based on requirements and funding availablility, but all LLM-based approaches were significantly more expensive than the embedding approaches. These require sufficient data (up to two orders of magnitude more than the LLM approaches), but proved competitive and cost-effective given enough data.