# Exploring the Zero-Shot Capabilities of LLMs Handling Multiple Problems at once

**Anonymous NAACL submission**


## Abstract

Recent studies have proposed placing multiple problems in a single prompt to improve input token utilization for a more efficient LLM inference. We call this *multi-problem prompting* (MPP), in contrast to conventional *single-problem prompting* (SPP) that prompts an LLM with a single problem at a time. While MPP has been shown to work comparably well or even better than SPP under few-shot settings, its zero-shot performance is underexplored, which better reveals the innate multiple problem handling capabilities of LLMs. To address that, we study the zero-shot MPP performance of various LLMs on 6 classification and 12 reasoning benchmarks and confirm that LLMs are competent zero-shot multi-problem solvers. We also examine the conditions of effectiveness of zero-shot MPP and explore several model-level factors that may enable MPP. We observe that LLMs consistently perform worse with selecting indices of texts of a given class label and with multiple mixed-source reasoning problems. This indicates a lack of true understanding. We also find that instruction tuning is an important factor than enhances MPP.


## 1 Introduction

Large language models (LLMs) have recently shown remarkable natural language understanding and reasoning capabilities (OpenAI, 2023; Gemini-Team, 2023; Anthropic, 2024), both under zero-shot (Wei et al., 2022; Kojima et al., 2022) and few-shot (Brown et al., 2020; Wei et al., 2023) settings. Given the non-trivial cost of deploying LLMs at large scale, substantial efforts have been made to make LLM inferences more cost efficient, e.g., distilling a smaller but highly capable model from a large one (Abdin et al., 2024; OpenAI, 2024).

Besides model-level distillation, recent studies (Cheng et al., 2023; Lin et al., 2024; Son et al., 2024) have proposed various prompt-level approaches that place multiple problems in a single prompt, which we collectively call *multi-problem prompting* (MPP). As illustrated in Fig 1, MPP (e.g., BatchClf) inserts multiple problems $P = (P_1, P_2, ..., P_n)$ after a shared context $C$ (e.g., task instruction and/or exemplars) to avoid repeating $C$ $n$ times for each problem as in standard *single-problem prompting* (SPP, e.g., SingleClf), which improves input token utilization. In practice, MPP is proposed in few-shot settings, where the context, which contains exemplars, is typically significantly longer than individual problems and an increased inference efficiency is guaranteed for the less repetitions of the shared context (Cheng et al., 2023).

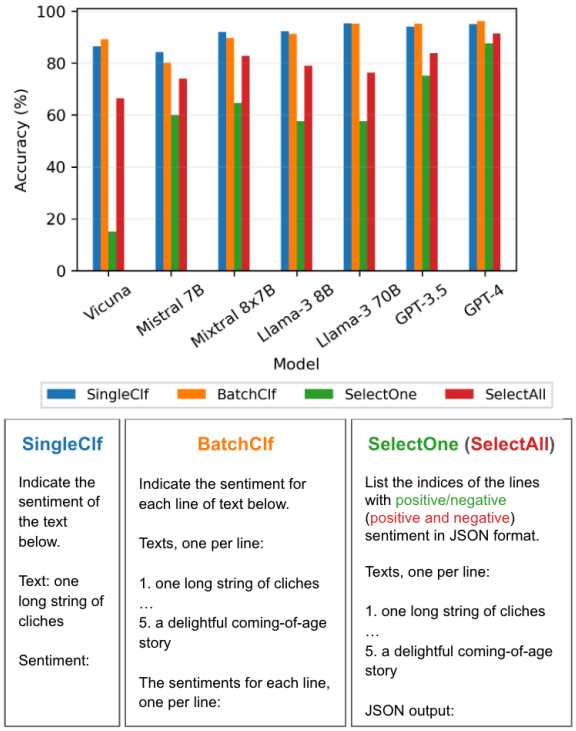


Figure 1: Accuracy of 7 LLMs on 4 task types constructed from SST-2 with up to 5 problems, including standard zero-shot prompting (SingleClf), zero-shot multi-problem prompting (BatchClf), and its reformulations (SelectOne and SelectAll). The text boxes provide simplified task illustrations. See Section 3 for details.

Empirically, MPP is also found to perform comparably well or even better than SPP across a wide range of domains under few-shot settings (Cheng et al., 2023; Lin et al., 2024; Son et al., 2024).

Despite progress in few-shot MPP, the performance of LLMs in zero-shot MPP remains underexplored. This is significant on at least two levels. First, on a practical level, many real-world tasks, such as classification, are typically approached in zero-shot settings (Ziems et al., 2024). Moreover, designing few-shot exemplars can be tedious and costly to obtain in practice (Kojima et al., 2022; Yasunaga et al., 2024). Second, from a scientific perspective, zero-shot MPP provides deeper insights into the innate capabilities of LLMs concurrently handling multiple tasks, a key indicator of intelligence (Meyer and Kieras, 1997).

To this end, we evaluate LLMs from 4 model families with zero-shot MPP on multi-problem prompts constructed from 6 classification and 12 reasoning benchmarks. To further understand the conditions of effectiveness of MPP, we also perform MPP with two task reformulations (i.e., SelectOne/SelectAll in Fig 1) and with multiple mixed-source problems leveraging these classification and reasoning benchmarks, respectively. While MPP is meant to be an inference-efficient prompting technique, *the main motivation of the paper is to provide scientific understandings* of LLM capabilities related to the concurrent handling of multiple problems through comprehensive and systematic contrastive evaluation. For this reason, we also explore additional evidence of strong multiple problem handling capabilities of LLMs and possible model-level factors that may enable MPP.

Our main contributions are as follows:

- We demonstrate the general effectiveness of zero-shot MPP, when the problems are homogeneous or from a single source.
- We identify two general conditions under which all tested LLMs perform significantly worse than expected and explore the roles of several model-level factors in MPP.
- We design a new benchmark ZeMPEB (**Ze**ro-shot **M**ulti-**P**roblem **E**valuation **B**enchmark), comprising 53,100 zero-shot multi-problem prompts containing classification and reasoning problems, for evaluating the zero-shot multiple problem handling capabilities of LLMs. ZeMPEB will be released upon acceptance.

The paper proceeds as follows. We reviews related works in Section 2. We perform zero-shot MPP and related experiments with classification and reasoning problems in Sections 3 and 4, respectively. We separate the classification and reasoning experiments in two sections due to the different natures of classification and reasoning, which we leverage for more nuanced evaluations. We explore several model-level factors that may enable MPP in Section 5. We conclude in Section 6.

## 2 Related Work

We note that current LLM evaluation has predominantly focused on LLM's performance on single-problem prompts. Each of such prompts presents a single problem and expects one specific answer to that problem, which may *implicitly* require multi-hop reasoning or multi-step task solving.

Recently, Cheng et al. (2023) propose few-shot MPP named batch prompting that prompts LLMs with problems batched together from single sources following a few batches of equally sized exemplars. They found that few-shot MPP greatly increases LLM inference efficiency while retaining downstream performance with a small batch size (e.g., <6). To ensure that batch prompting works with large batch sizes, Lin et al. (2024) introduce a sampling optimization method that takes a majority vote over repeated permutations of batch samples.

Instead of solving multiple *separate* problems, Son et al. (2024) prompt LLMs with exemplars to solve multiple *related* tasks based on a shared problem setup by placing an *explicit* instruction for each task. They find that instructing LLMs to solve all the tasks at once outperforms solving the individual tasks one by one or in a batch.

The related studies surveyed so far imply that LLMs have strong multiple problem handling capabilities under few-shot settings. To the best of our knowledge, Laskar et al. (2023) is the only study that shows instruction-tuned GPTs can handle 5 short questions sampled from two open-domain QA benchmarks at once zero-shot, but they do not conduct no further experiments. Overall, it is poorly understood how well LLMs can concurrently handle multiple problems zero-shot.

## 3 MPP: Classification

We first start our MPP experiments with classification problems, followed by experiments with reasoning problems in the next section.

| Problem Type | Input/Output Format | Benchmark | # Examples | Objective |
|---|---|---|---|---|
| Classification | Single-text input | SST-2 (Socher et al., 2013) | 1,821 | Sentiment analysis |
| | | CoLA (Warstadt et al., 2019) | 1,043 | Grammatical acceptability |
| | | AGNews (Gulli, 2004) | 1,000 | Topic classification |
| | Text-pair input | MRPC (Dolan and Brockett, 2005) | 1,725 | Paraphrase detection |
| | | SNLI (Bowman et al., 2015) | 1,000 | Natural language inference |
| | | WiC (Pilehvar and Camacho-Collados, 2019) | 1,400 | Word sense disambiguation |
| Reasoning | Yes/no output | StrategyQA (Geva et al., 2021) | 2,288 | Commonsense reasoning |
| | | Coin Flips (Wei et al., 2023) | 500 | Symbolic reasoning |
| | Multi-choice output | AQuA (Ling et al., 2017) | 254 | Arithmetic reasoning |
| | | CommonsenseQA (Talmor et al., 2019) | 1,221 | Commonsense reasoning |
| | | Object tracking (Srivastava et al., 2023) | 750 | Commonsense reasoning |
| | | Bigbench date (Srivastava et al., 2023) | 363 | Commonsense reasoning |
| | Free-response output | Last Letters (Wei et al., 2023) | 500 | Symbolic reasoning |
| | | SVAMP (Patel et al., 2021) | 1,000 | Arithmetic reasoning |
| | | GSM8K (Roy and Roth, 2015) | 1,319 | Arithmetic reasoning |
| | | MultiArith (Patel et al., 2021) | 600 | Arithmetic reasoning |
| | | AddSub (Hosseini et al., 2014) | 395 | Arithmetic reasoning |
| | | SingleEq (Koncel-Kedziorski et al., 2015) | 508 | Arithmetic reasoning |

Table 1: Benchmarks used in Section 3 (Classification) and Section 4 (Reasoning). We use the test splits wherever possible, except for CoLA, StrategyQA, and CommonsenseQA, for which we use the dev splits, since the test splits are not publicly available. For AGNews and SNLI, we randomly sample 1,000 examples from the test splits.

### 3.1 Data

We consider the following 6 classification benchmarks, as described and referenced in Table 1: SST-2, CoLA, AGNews, MRPC, SNLI, and WiC. These benchmarks are commonly used for NLP evaluation, with SST-2, CoLA, and MRPC appearing in GLUE (Wang et al., 2019) and WiC in SuperGLUE (Sarlin et al., 2020). They cover (1) two classification paradigms, i.e., single-text and text-pair classification, and (2) six distinct task objectives.

### 3.2 Evaluation Tasks

For each benchmark with $m$ class labels $L = (L_1, ..., L_m)$, we perform standard zero-shot single-problem prompting (SPP) on all the problems to establish baselines for LLMs. We randomly sample 100 distinct sets of 100 problems $\{P^{(i)} = (P_1^{(i)}, P_2^{(i)}, ..., P_{100}^{(i)})\}_{i=1}^{100}$ out of each benchmark and take the first $n$ problems from each problem set $P^{(i)}$ to generate a multi-problem prompt of task size $n$, with the task size for SPP being 1. We consider the following 5 task sizes and ensure that each task size has 100 distinct problem sets: 5/10/20/50/100 for single-text benchmarks and 3/5/10/20/50 for text-pair benchmarks. Note that by design a larger task size only differs from a smaller one by having additional problems given a benchmark, which helps us isolate the effect of task size. In total, this results in 13,500 prompts for all MPP tasks.

As illustrated in Fig 1, there are a total of 4 classification-related task types. These task types are designed to minimally differ from one another such that model performance on a task pair with minimal differences immediately reveals its (lack of) ability in certain aspects we intend to test. We provide further descriptions of the 4 task types below, where every two neighboring task types make a minimally different task pair.

**Single Classification (SingleClf)** This is SPP for classification. The prompt indicates the task objective as well as the possible class labels, followed by one target classification problem to be classified.

**Batch Classification (BatchClf)** This is MPP for classification. The prompt for BatchClf is nearly identical to that of SingleClf to maximize comparability, expect that it incorporates multiple target classification problems to be classified at once.

**Index Selection One Label (SelectOne)** SelectOne is a reformulation of BatchClf. Instead of doing batch classification, SelectOne instructs LLMs to select *all* indices out of $n$ problems $P$ belonging to each class label $L_i$ *one at a time* in $m$ *separate* prompts, i.e., $\{j : label(P_j) = L_i, j \in [1, n]\}$ for $i \in [1, m]$. The expected output for each $L_i$ is "None," "All," or a list of applicable indices.

**Index Selection All Labels (SelectAll)** SelectAll prompts a system to do SelectOne for all possible class labels $L$ *all at once*. The expected output is thus a concatenation of SelectOne output for $L$, i.e., $\{(L_i, \text{SelectOne}(P, L_i)) : i \in [1, m]\}$.

### 3.3 Experimental Setups

**LLMs and evaluation settings** We evaluate 7 LLMs from 4 model families with greedy decoding: Vicuna (13B, Chiang et al., 2023), Mistral

| | BatchClf vs SelectOne | BatchClf vs SelectAll | SelectOne vs SelectAll |
|---|---|---|---|
| Mean Acc Dif | 32.0 | 12.1 | -19.9 |
| Std Dev | 16.9 | 15.3 | 12.0 |
| Cohen's $d$ | 1.8 | 0.8 | -1.0 |

Table 2: Pairwise accuracy differences (x vs y = x - y). All the differences are statistically significant and with a large effect size (| Cohen's $d$ | $\geq 0.8$).

7B (Jiang et al., 2023), Mixtral 8x7B (Jiang et al., 2024), Llama-3 8B and 70B (Instruct, Meta, 2024), GPT-3.5, and GPT-4 (OpenAI, 2023).

**Performance metric** We measure the average per-problem accuracy (PPA) to unify the evaluation across the four task types. PPA, defined in Equation 1, is the average accuracy of classifying $n$ problems in each prompt or, in the case of SelectOne, in each set of directly related prompts targeting different class labels.

$$\text{PPA} = \frac{1}{n}\sum_{i=1}^{n}\delta(I(P_i), A_i) \quad (1)$$

$I(P_i)$ is the inferred LLM-generated answer to the $i$th problem in the input prompt, $A_i$ is the ground truth, and $\delta(i, j) = 1$ iff $i = j$ and 0 otherwise. For the two index selection tasks, $I(P_i)$ is determined by considering the LLM's assignments of indices to all class labels. Other than assigning an index to a wrong class label, there are two more error types. First, LLMs may assign an index with more than one class label, i.e., an *contradiction* error. Second, LLMs may assign no labels to an index at all, namely, an *non-excluded middle* error.

To compare performance difference, we use Mann-Whitney U tests for significant testing and Cohen's $d$ (Cohen, 1969) for measuring effect size.

### 3.4 Results

Overall, we observe that although LLMs are competent zero-shot multi-problem solvers for classification (Section 3.4.1), their performance decreases significantly when the same sets of multiple problems are presented differently (Section 3.4.2).

#### 3.4.1 Zero-shot MPP: SingleClf vs. BatchClf

In line with previous studies (Cheng et al., 2023; Lin et al., 2024) on few-shot MPP, we find that zero-shot MPP also performs comparably well or even better than zero-shot SPP in a more cost-efficient way. We further validate this finding by comparing the errors LLMs make under SPP and MPP.

**LLMs can handle multiple classifications at once under zero-shot with minimal performance loss.** Although the BatchClf accuracy generally declines as the task size increases (Fig 2), all 7 LLMs achieve accuracy of at least 90% that of SingleClf across the benchmarks most of the time (see Table 5 in Appendix B). Overall, the SingleClf accuracy for the 7 LLMs on the 6 benchmarks is 75.5% and the BatchClf accuracy is 72.3%, a minor 3.2% absolute drop from the former.

**Zero-shot MPP can be cost-efficient.** With a general task instruction being shared, zero-shot MPP avoids repeating this instruction for each problem to be solved. Given the strong zero-shot MPP performance, this makes MPP a promising cost-efficient zero-shot prompting technique. To illustrate, we choose for each model/task combination the largest BatchClf task size that achieves at least 95% SingleClf accuracy for that pair. We observe that MPP reduces substantial inference costs for all LLMs run on the 6 benchmarks, ranging from from 30.7% to 82.0% (see Fig 7 in Appendix B).

**LLMs make similar errors evenly across positions under multiple problems.** Given the results above, two natural questions arise: do LLMs make similar errors under MPP and how do their errors distribute across positions? For each one of the 180 LLM/benchmark/task size combinations, we use chi-squared tests to compare the proportional error distribution across class labels under SingeClf and BatchClf and compute the cumulative error density across positions under BatchClf. We observe that (1) only in 9 out of 180 (or 5%) cases, error labels are distributed significantly differently between SingleClf and BatchClf ($p < 0.05$); and (2) surprisingly, LLMs typically do not display a clear positional bias or a serial position effect as known in psychology (Murdock, 1962), when solving sufficiently many problems at once (Fig 3). This is in contrast to previous studies based on single-problem prompts where LLMs are found to be better at using information from the beginning (primacy bias) or the end (receny bias) of the prompt (Liu et al., 2024; Levy et al., 2024).

We hypothesize that the differences in observations may be due to different ways of measuring positional biases. Without a precise definition of positional biases and clear notions of what counts as "beginning," "middle," and "end," we find it necessary to look at cumulative error density (Fig 3) to identify the overall positional error distribution.

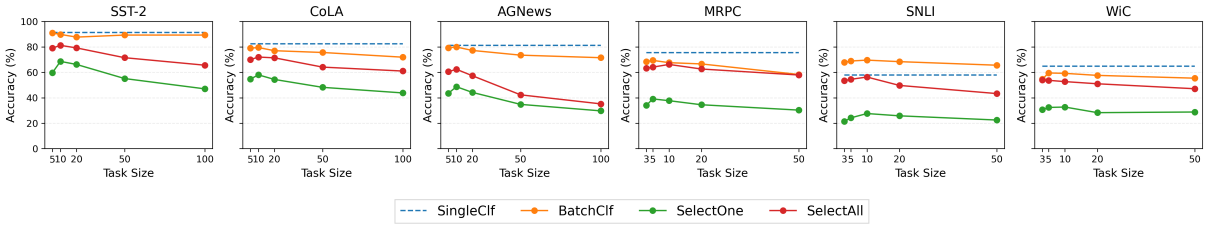


Figure 2: Aggregated average accuracy of the 7 LLMs on the 4 types of tasks across various task sizes for each benchmark. See Appendix B for the full results for each LLM.

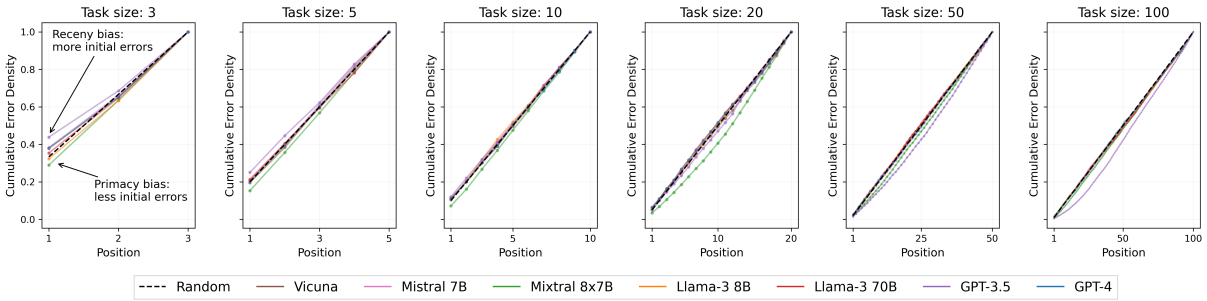


Figure 3: Cumulative error density across positions averaging results from the benchmarks. Task size 3/100: the 3 text-pair/single-text benchmarks; otherwise: all the 6 classification benchmarks. See Appendix B for full results.

We leave the arguments for future studies.

Taken together, the fact that LLMs make similar label prediction errors and the lack of an obvious positional bias imply that LLMs can use information equally well across different positions under multiple classification problems. This may explain their strong multiple problem handling capabilities.

### 3.4.2 MPP Reformulations: Selection Tasks

We further compare LLM's performance between BatchClf and the two selection tasks.

**LLMs perform significantly worse on the selection tasks.** In our experiments, LLMs nearly always perform much better in BatchClf than in SelectOne and SelectAll under the same conditions with a consistent and stable margin, even when the task size is just 3 or 5 (Figs 1, 2). The overall discrepancy in accuracy between BatchClf and the two tasks is large and statistically significant (32% for SelectOne and 10% for SelectAll, see Table 2) and generally increases with a larger task size (Fig 2).

The sharp drop in accuracy may not be human-like, because arguably, humans should at least be able to classify and select a small number (e.g., 3/5) of texts equally well simply by thinking over the problems (i.e., zero-shot). This suggests either LLMs' lack of true understanding of the problems presented in various forms or their struggle at index selection.

**LLMs lack internalized planning and consistently select inconsistently.** Surprisingly, such a consistent and rather stable performance gap also exists between SelectOne and SelectAll in favor of the latter, largely independently of the task size (Fig 2). On average, the SelectOne accuracy is 19.9% lower than the SelectAll accuracy, also with a large and significant effect size (Table 2).

We investigate the reasons in Fig 4, which shows that when asked to select text indices for one class label at a time independently, LLMs almost always assign an index to multiple labels (i.e., contradiction) and leave some indices unselected (i.e., non-excluded middles) more often. This showcases a lack of internalized planning with modern zero-shot LLMs, although different LLMs may make these two types of errors in different proportions. In contrast, when LLMs have to select indices for all labels at once, they are less likely to generate directly illogical answers in a single output as their answer to the $(i+1)_{th}$ label is conditioned by their answer to the $i_{th}$ label. This highlights the *purely probabilistic* aspect of LLM answer generation.

**LLMs can do index selection, but fail to combine it with text classification adequately.** To see if LLMs' significantly poorer performance with selection tasks is due to their struggle at index selection, we perform a follow-up study. More concretely, we simplify the two index selection tasks by directly

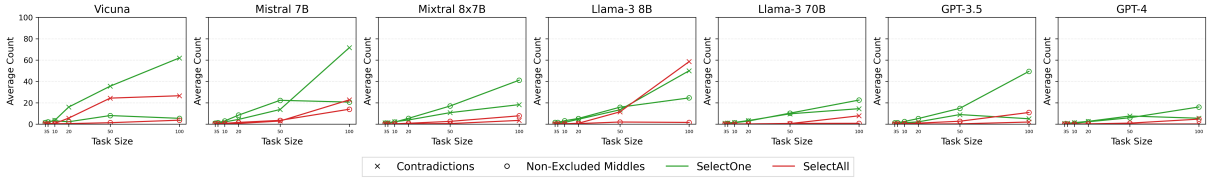


Figure 4: Pairwise comparisons of SelectOne and SelectAll for each LLM across different task sizes averaging over results from the 6 benchmarks.

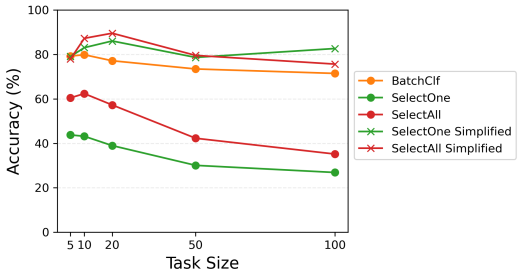


Figure 5: Average accuracy of the 7 LLMs on the two simplified index selection tasks based on AGNews. We provide the original results from BatchClf, SelectOne, and SelectAll for easy comparisons.

replacing each line of text with its gold class label in the prompts for AGNews, which has most labels. We then ask the LLMs to select indices of lines containing each label with minimal modifications to the original task instructions (see Appendix E).

Fig 5 shows that LLMs can perform in the simplified selection tasks much better than the original ones, with Llama-3 70B, GPT-3.5, and GPT-4 even achieving (nearly) 100% accuracy across task sizes. Their overall performance in the simplified selection tasks even surpasses their overall BatchClf accuracy. The previously observed performance gap between SelectOne and SelectAll also nearly disappears (see Appendix B for full results). In general, LLMs can indeed do index selection.

In conclusion, it appears that the tested LLMs perform less well on SelectOne and SelectAll as compared to BatchClf because they cannot adequately combine the index selection task and the classification tasks in response to a zero-shot prompt.[1] Put differently, the LLMs lack true understanding of the problems presented in different forms, even when the number of problems is quite small (e.g., 3 or 5), which may not be human-like.

[1]We also tried 1-shot-CoT to explicitly prompt LLMs to do BatchClf first and then select indices from there. Although that helps, this basic finding remains. See Appendix B.6.

## 4 MPP: Reasoning

We provide the MPP experimental details and results for reasoning problems in this section.

### 4.1 Data

We further experiment with another 12 reasoning benchmarks (listed and referenced in Table 1), commonly used in LLM evaluation (Kojima et al., 2022; Wei et al., 2023; Zhang et al., 2023, *inter alia*). These benchmarks test symbolic reasoning (Coin Flips & Last Letters), commonsense reasoning (StrategyQA, CommonsenseQA, Object tracking, & Bigbench date), and arithmetic reasoning (AQuA, SVAMP, GSM8K, MultiArith, AddSub, & SingleEq), and require three answer formats (Yes/No, multiple choice, and free-response).

### 4.2 Evaluation Tasks

We first test on all the reasoning problems in each benchmark to establish LLM SPP baselines. Two MPP evaluation task types are designed, i.e., single-source and mixed-source multi-problem reasoning, or MultiReason$^{SS}$ and MultiReason$^{MS}$. For both task types, we simply bundle multiple reasoning problems together with an indexed prefix "Qi: ," where i is the index of each problem starting from "Q1." We use two headers, namely, "Questions," followed by the bundled questions, and "Answers," followed by the answers to be generated by LLMs.

Besides vanilla zero-shot prompting, we also perform zero-shot-CoT following Kojima et al. (2022).[2] Inspired by Cheng et al. (2023) as well as to control for the number of prompts generated, we consider smaller task sizes from 2 to 10. To ensure a reliable evaluation (e.g., sufficient parsable outputs), we increase the number of prompts from 100 to 300 for each benchmark/task size combination. This results in 21,600 and 18,000 prompts for MultiReason$^{SS}$ and MultiReason$^{MS}$, respectively.

[2]In our early experiments, we found that zero-shot-CoT did not lead to different responses for the classification-related MPP tasks probably due to their novelty, so it was not used.

**MultiReason$^{SS}$** For each one of the 12 benchmarks, we first sample 300 distinct sets of 10 problems. We then use the first 2, 5, and 10 problems from each problem set to make prompts.

**MultiReason$^{MS}$** To robustly examine an LLM's performance on mixed-source prompts, we create 6 distinct benchmark combinations based on the 12 benchmarks, each consisting of 10 different benchmarks. For each combination, we randomly sample 300 distinct sets of 10 problems, each coming from one benchmark and with the order of benchmarks randomized across the 300 problem sets. We then take out 2/4/6/8/10 problems from the first 2/4/6/8/10 benchmarks for each benchmark combination across each problem set to make mixed-source prompts. See Appendix B for details.

### 4.3 Experimental Setups

**LLMs and evaluation settings** For budget reasons, we only use two LLMs, i.e., GPT-3.5 and Llama-3 70B. Since Llama-3 models tend to produce reasoning steps even when not instructed to do so, we only prompt GPT-3.5 with zero-shot-CoT. We always prompt with greedy decoding.

**Performance metric** We use the per-problem accuracy (PPA), defined in Equation 1, to measure the model performance. For mixed-source prompts containing $k$ problems evenly sampled from $k$ benchmarks, we compute the expected PPA by averaging over the observed SPP performance for each one of these benchmarks.

### 4.4 Results

We observe that although LLMs can be competent zero-shot multi-problem solvers for reasoning, their performance becomes consistently worse than expected under multiple mixed-source problems. Similar to our arguments in Section 3, the consistent performance declines even with a small number (e.g., 2 or 4) of mixed-source problems may indicate a lack of human-like understanding, as LLMs' reasoning capabilities are easily impacted by the mixing of problems from different sources.

**LLMs can do MultiReason$^{SS}$ on par with their SPP performance.** Similar to Cheng et al. (2023), we observe in Fig 6 (A) that both LLMs, to varying extents, can handle multiple single-source reasoning problems as well as or even better than when they handle these problems individually, although their MPP performance typically goes down with a larger task size.

**When the reasoning problems are from mixed sources, LLMs perform worse than expected.** Interestingly, as shown in Fig 6 (B), the observed MultiReason$^{MS}$ performance is almost always lower than the expected one computed by averaging over the SPP performance over each benchmark for both LLMs, with and without CoT. Out of 540 model (including GPT-3.5 with zero-shot-CoT) and benchmark pairs, there are only 18.3% cases in which the observed performance is better than the expected one for a given model/benchmark pair by a small margin (mean/std: 2.9%/2.5%). However, for the rest 81.7% cases when the expected performance is better, the margin is larger (mean/std: 13.5%/12.7%). In other words, LLMs typically perform worse in each benchmark when handling multiple reasoning problems from mixed sources.

**Benefits of zero-shot-CoT prompting are transferrable under MPP.** Analogous to the finding that zero-shot-CoT improves LLMs' reasoning performance under SPP (Kojima et al., 2022), GPT-3.5 generally performs better with CoT than without it under zero-shot MPP for both MultiReason$^{SS}$ and MultiReason$^{MS}$. The transferrability of zero-shot-CoT[3] indicates that LLMs can apply CoT over each problem in the prompt and benefit from the generated reasoning steps when solving each problem. This again implies the strong capabilities of LLMs to utilize information across positions under MPP.

## 5 Exploring Model-level Factors that may Enable MPP

Since so far we have only tested decoder-only and instruction-tuned LLMs, which all show strong performance under MPP, we explore if these two factors enable MPP. For these reasons, we test with greedy decoding (1) three FLAN-T5 models (Chung et al., 2022), i.e., Large, XL, XXL; and (2) three pretrained LLMs, i.e., Llama-3 8B (Base, Meta, 2024), GPT-3 1.3B, and GPT-3 175B (Brown et al., 2020). We run these 6 LLMs on CoLA at task sizes 1 and 5 under zero-shot settings with results shown in Table 3. Experiments with Coin Flips show similar results, but the LLM outputs are less meaningful, as discussed in Appendix D.

We make three observations from Table 3. First, **instruction tuning helps**, since pretraned decoder-

[3] Similarly, Cheng et al. (2023) show that the benefits of few-shot-CoT are transferrable under MPP.

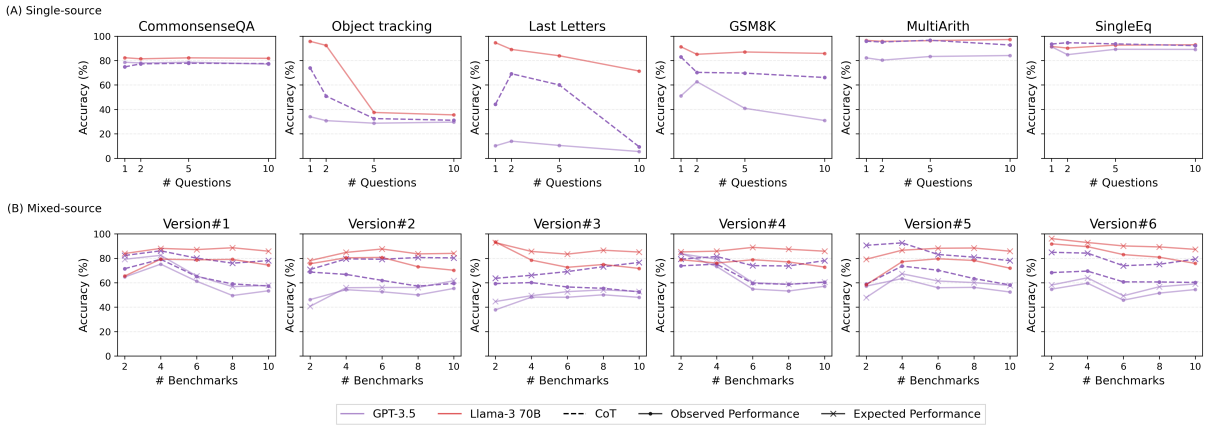


Figure 6: Average accuracy of GPT-3.5 and Llama-3 70B on multiple single-source (MultiReason$^{SS}$) and mixed-source (MultiReason$^{MS}$) reasoning problems. (A) MultiReason$^{SS}$ results on 6 selected benchmarks for space reasons. The other 6 benchmarks show similar results (see Appendix B). (B) MultiReason$^{MS}$ results across 6 distinct benchmark combinations, where each benchmark contributes a problem to each mixed-source prompt.

only LLMs either cannot handle multiple problems at once or their performance is much worse than their instruction-tuned counterparts. However, unlike what Laskar et al. (2023) suggests, instruction tuning may not be a necessary condition for MPP, as both Llama-3 8B and GPT-3 175B can perform reasonably well in BatchClf on CoLA.

Second, **FLAN-T5 can barely respond to MPP**, regardless of model sizes. We suspect that this is may not be due to their encoder-decoder structures, but other factors such as training data and reinforcement learning from human feedback (Christiano et al., 2017). We leave it for future investigation.

Third, **scaling model size seems helpful**. With other factors being identical, larger models appear to perform better than the smaller ones under MPP.

Overall, instruction tuning appears to be the most important factor that *enhances* MPP. We leave more careful explorations to future research.

| | SingleClf | BatchClf | Avg # Answers |
|---|---|---|---|
| Llama-3 8B (Instruct) | 80.5 | 79.4 | 5.0 |
| GPT-3.5 | 84.2 | 79.6 | 5.0 |
| Llama-3 8B (Base) | 78.5 | 60.6 | 5.04 |
| GPT-3 1.3B | 63.0 | 0.0 | 0.03 |
| GPT-3 175B | 66.6 | 64.4 | 5.08 |
| FLAN-T5-Large (0.78B) | 76.0 | NA | 1.0 |
| FLAN-T5-XL (3B) | 80.2 | NA | 1.0 |
| FLAN-T5-XXL (11B) | 78.2 | 4.0 | 1.2 |

Table 3: SingleClf and BatchClf (task size 5) accuracy (%) of three pretrained LLMs and three FLAN-T5 models on CoLA. We also include the results of Llama-3 8B (Instruct) and GPT-3.5 (likely base model: GPT-3 175B) from Section 3 to compare with their respective base models. The last column is the average number of LLM-generated answers for BatchClf (expects 5). When it is 1, accuracy is not calculated to avoid overinterpretation.

## 6 Conclusion

In this study, we present a comprehensive evaluation of LLMs with zero-shot MPP. We evaluate various LLMs from 4 model families on single-source multi-problem prompts constructed from 6 classification and 12 reasoning benchmarks. In line with previous few-shot results, we confirm that LLMs are competent multi-problem solvers for classification and reasoning under zero-shot settings. Moreover, we find multiple pieces of evidence that validate the strong innate multiple problem handling capabilities of LLMs, such as the similar classification errors LLMs make under SPP and MPP, the lack of obvious positional biases, and the transferrability of zero-shot-CoT under MPP.

Two conditions are identified under which LLMs show consistent performance declines with MPP: (1) reformulating Batch Classification as index selection tasks; and (2) mixing reasoning problems from different sources in a multi-problem prompt. Noticeably, these performance declines happen even when the number of problems included is rather small (e.g., $\leq 5$), which may not be human-like and indicates a lack of true understanding. In addition, we explore several model-level factors that may enable MPP and find instruction tuning to be an important factor that enhances MPP.

As a result of our study, we create a new benchmark comprising 53,100 zero-shot multi-problem prompts. We call it ZeMPEB, which stands for **Ze**ro-shot **M**ulti-**P**roblem **E**valuation **B**enchmark. We will release ZeMPEB to aid future research.

## Limitations

While we have done our best to make our experiments as comprehensive as possible, there is always more work that can be done in a large comparison study, including the addition of even more benchmarks and language models. We were limited by cost, time, and space and have attempted to select an informative and representative sample of experiments. Since we used pre-existing benchmarking datasets, we inherit any errors that they main contain. Finally, despite our efforts to compare different prompts, as is the case with all prompt-based LLM studies, we cannot guarantee that slight differences in the prompts would not meaningfully alter the results.

## Ethical Concerns

To the best of our knowledge, all results published in this paper are accurate. All data sources are free, publicly available, and cited in the article. No sensitive data was used which could violate individuals' privacy or confidentiality.

## References


Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Qin Cai, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Yen-Chun Chen, Yi-Ling Chen, Parul Chopra, Xiyang Dai, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Victor Fragoso, Dan Iter, Mei Gao, Min Gao, Jianfeng Gao, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Ce Liu, Mengchen Liu, Weishung Liu, Eric Lin, Zeqi Lin, Chong Luo, Piyush Madan, Matt Mazzola, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Xin Wang, Lijuan Wang, Chunyu Wang, Yu Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Haiping Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Sonali Yadav, Fan Yang, Jianwei Yang, Ziyi Yang, Yifan Yang, Donghan Yu, Lu Yuan, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone.

Anthropic. 2024. The Claude 3 model family: Opus, Sonnet, Haiku.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch prompting: Efficient inference with large language model APIs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810, Singapore. Association for Computational Linguistics.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* chatgpt quality.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

J. Cohen. 1969. *Statistical Power Analysis for the Behavioral Sciences*. Academic Press.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Gemini-Team. 2023. Gemini: A family of highly capable multimodal models.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Antonio Gulli. 2004. AG's corpus of news articles.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

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.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Huang. 2023. A systematic study and comprehensive evaluation of ChatGPT on benchmark datasets. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 431–469, Toronto, Canada. Association for Computational Linguistics.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models.

Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. 2024. Batchprompt: Accomplish more with less.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Meta. 2024. The llama 3 herd of models.

David E. Meyer and David E. Kieras. 1997. A computational theory of executive cognitive processes and multiple-task performance: Part i. basic mechanisms. *Psychological Review*, 104(1):3–65.

Bennet B. Murdock. 1962. The serial position effect of free recall. *Journal of Experimental Psychology*, 64(5):482–488.

OpenAI. 2023. GPT-4 technical report.

OpenAI. 2024. GPT-4o mini: advancing cost-efficient intelligence.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2020. SuperGlue: Learning feature matching with graph neural networks.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Guijin Son, SangWon Baek, Sangdae Nam, Ilgyun Jeong, and Seungone Kim. 2024. Multi-task inference: Can large language models follow multiple instructions at once? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5606–5627, Bangkok, Thailand. Association for Computational Linguistics.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne

Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H. Chi, and Denny Zhou. 2024. Large language models as analogical reasoners.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2024. Can large language models transform computational social science? *Computational Linguistics*, 50(1):237–291.

# A LLM Details

We use a total of 13 LLMs in our study, from Section 3 to Section 5. Table 4 describes the specific versions for these LLMs and highlights their differences in terms of architecture, open weights, Supervised Fine-Tuning (SFT, Wei et al., 2022), and Reinforment Learning from Human Feedback (RLHF, Christiano et al., 2017) etc.

# B MPP: Classification

This section contains additional details for Section 3. The prompt details for the experiments can be found in Appendix E.

## B.1 Full Results

The full results obtained from Section 3 are visualized in Fig 8. We exclude the results of Vicuna on AGNews when task size is 100 because the prompts exceed the model's context length.

## B.2 SingleClf vs. BatchClf

Table 5 indicates the proportion of BatchClf tasks for which each LLM surpasses a threshold percent of corresponding SingleClf performance.

## B.3 Zero-shot MPP can be cost-efficient

Fig 7 shows the cost saving rate for each model/task pair at the largest BatchClf task size that achieves at least 95% SingleClf accuracy for that pair.

## B.4 Positional Errors under BatchClf

Fig 9 shows the full results regarding the positional errors 7 LLMs make across benchmarks and task sizes. We note that (1) distribution of the positional errors becomes more random (or even) as the task

| Model | Version | Architecture | Open Weights | SFT | RLHF | MoE | # Parameter | Context Length |
|---|---|---|---|---|---|---|---|---|
| Vicuna (Chiang et al., 2023) | v1.5 | decoder-only | ✓ | ✓ | ✗ | ✗ | 13B | 4,096 |
| Mistral 7B (Jiang et al., 2023) | Instruct-v0.2 | decoder-only | ✓ | ✓ | ✗ | ✗ | 7B | 8,192 |
| Mixtral 8x7B (Jiang et al., 2024) | Instruct-v0.1 | decoder-only | ✓ | ✓ | ✗ | ✓ | 47B | 8,192 |
| Llama-3 8B (Meta, 2024) | Instruct | decoder-only | ✓ | ✓ | ✓ | ✗ | 8B | 8,192 |
| Llama-3 70B (Meta, 2024) | Instruct | decoder-only | ✓ | ✓ | ✓ | ✗ | 70B | 8,192 |
| GPT-3.5 | turbo-0125 | decoder-only | ✗ | ✓ | ✓ | ✗ | Unknown | 16,385 |
| GPT-4 (OpenAI, 2023) | turbo-2024-04-09 | decoder-only | ✗ | ✓ | ✓ | ✗ | Unknown | 128,000 |
| GPT-3 1.3B (Brown et al., 2020) | babbage-002 | decoder-only | ✗ | ✗ | ✗ | ✗ | 1.3B | 16,384 |
| GPT-3 175B (Brown et al., 2020) | davinci-002 | decoder-only | ✗ | ✗ | ✗ | ✗ | 175B | 16,384 |
| Llama-3 8B (Meta, 2024) | Base | decoder-only | ✓ | ✗ | ✗ | ✗ | 8B | 8,192 |
| FLAN-T5 (Chung et al., 2022) | Large | encoder-decoder | ✓ | ✓ | ✗ | ✗ | 0.78B | 512 |
| FLAN-T5 (Chung et al., 2022) | XL | encoder-decoder | ✓ | ✓ | ✗ | ✗ | 3B | 512 |
| FLAN-T5 (Chung et al., 2022) | XXL | encoder-decoder | ✓ | ✓ | ✗ | ✗ | 11B | 512 |

Table 4: Details about the 13 LLMs used in the study. For Mixtral 8x7B, a Mixture of Experts (MoE) LLM, although each token has access to 47B parameters, but only uses 13B active parameters during inference.

| | > 90% SCAcc | > 80% SCAcc | > 75% SCAcc |
|---|---|---|---|
| Vicuna 13B | 79.3 | 93.1 | 93.1 |
| Mistral 7B | 76.7 | 83.3 | 100.0 |
| Mixtral 8x7B | 63.3 | 83.3 | 86.7 |
| Llama-3 8B | 73.3 | 90.0 | 100.0 |
| Llama-3 70B | 80.0 | 100.0 | 100.0 |
| GPT-3.5 | 56.7 | 83.3 | 90.0 |
| GPT-4 | 100.0 | 100.0 | 100.0 |
| Overall | 75.6 | 90.4 | 95.7 |

Table 5: Percent of time that BatchClf performance surpasses a threshold percent of SingleClf accuracy (SCAcc) across benchmarks.

| Task<br>Model | SelectOne | + CoT | SelectAll | + CoT | BatchClf |
|---|---|---|---|---|---|
| Vicuna | 23.0 | 25.8 | 53.2 | 57.6 | 68.7 |
| Mistral 7B | 38.3 | 47.5 | 56.8 | 60.5 | 71.2 |
| Mixtral 8x7B | 47.4 | 38.7 | 65.1 | 58.4 | 73.4 |
| Llama-3 8B | 39.0 | 41.8 | 62.4 | 59.0 | 73.3 |
| Llama-3 70B | 59.4 | 67.1 | 72.9 | 79.2 | 79.4 |
| GPT-3.5 | 45.5 | 47.5 | 66.7 | 66.3 | 71.9 |
| GPT-4 | 66.3 | 71.8 | 78.8 | 81.8 | 81.9 |
| Overall | 45.5 | 48.6 | 65.1 | 66.1 | 74.3 |

Table 6: Aggregate average accuracy of SelectOne and SelectAll with and without 1-shot-CoT for each LLM. BatchClf performance is also provided for comparisons.

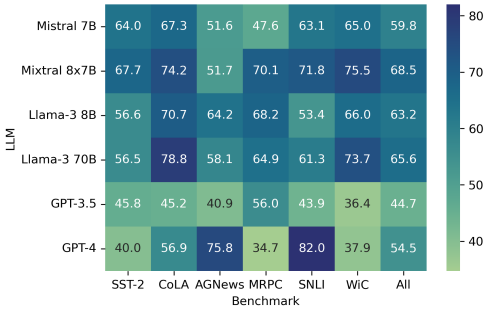


Figure 7: Cost saving rate (%) per classification based on our experiments. The cost is estimated by both the input and output token counts (using the respective tokenizers), weighted according to the pricing policy from OpenAI and TogetherAI (for non-GPT LLMs) websites.

size increases for all LLMs; (2) in most cases, the positional errors distribute nearly randomly, showing no evidence of obvious positional biases, if any; (3) some LLMs may display more severe positional biases on some benchmarks with a certain task size, such as GPT-3.5 on SST-2 with task size 50, but overall this is rare.

### B.5 Simplified Index Selection Tasks

Fig 10 shows the full results for the two simplified index selection tasks, along with results for the two original selection tasks and BatchClf, based on AGNews. Clearly, (1) all LLMs perform much better in the simplified tasks than the original ones; (2) Mixtral 8X7B, Llama-3 70B, GPT-3.5, and GPT-4 can do these simplified selection tasks even better than their BatchClf performance consistently, with the latter three achieving nearly 100% accuracy across task sizes; (3) the task complexity hierarchy between SelectOne and SelectAll nearly disappears for all LLMs except Vicua, implying that it may be challenging for Vicuna to perform index selection tasks in general.

### B.6 CoT Helps the Index Selection Tasks Slightly but the Gap Remains

We use 1-shot-CoT to prompt LLMs to do BatchClf first and then perform the two index selection tasks on the 6 benchmarks with a fixed task size 10. Table 6 shows that while overall LLMs benefit from CoT for both SelectOne and SelectAll, the benefits are generally larger for SelectOne than SelectAll (3.1% versus 1.0% improvement) and not consistent across models with Mixtral 8x7B performing even worse with CoT. The full results provided in Table 8 also show mixed effects across benchmarks. Interestingly, however, the task complexity hierarchy among BatchClf, SelectOne, and

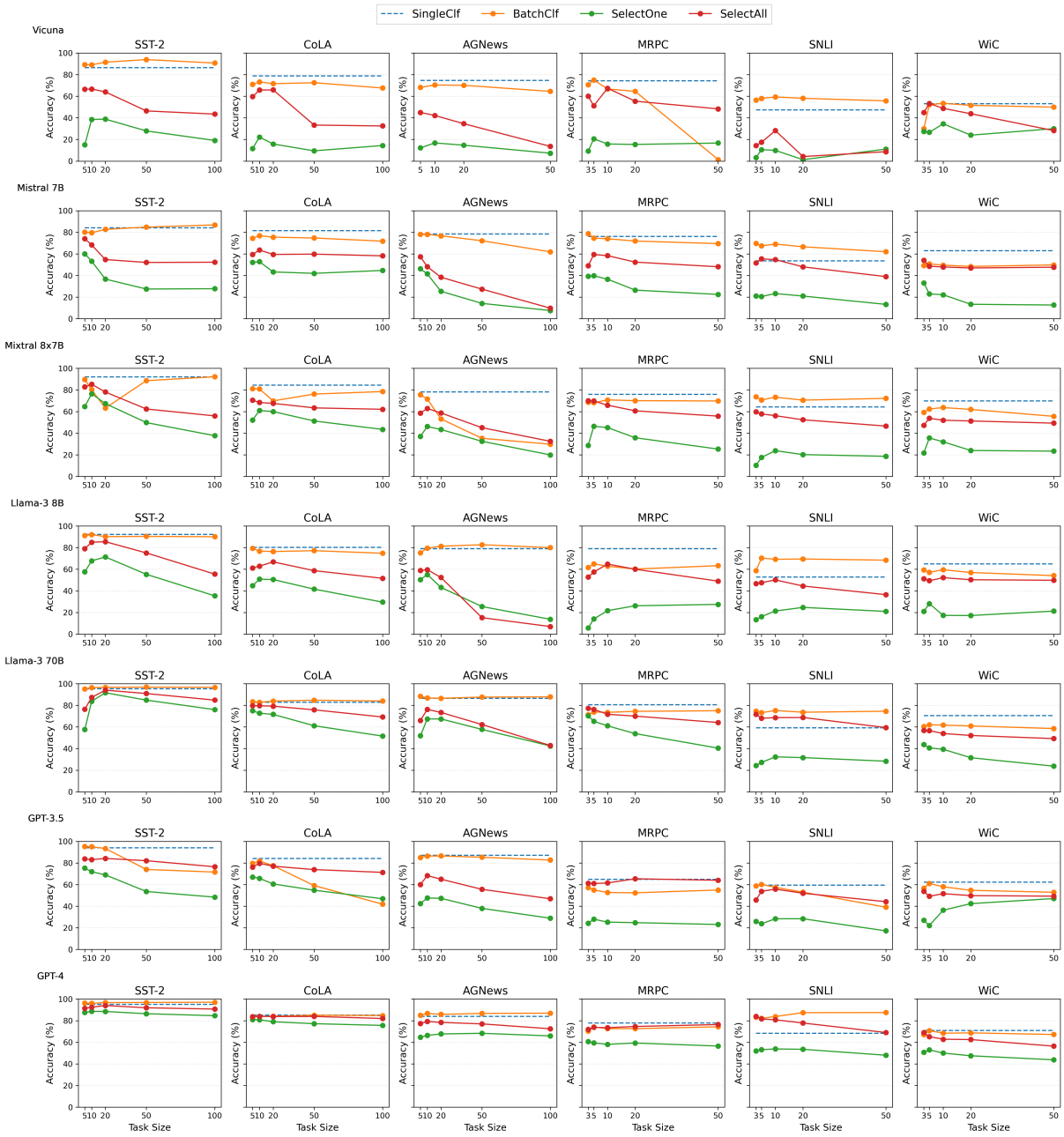


Figure 8: Full average accuracy of 7 LLMs on the 4 classification-related tasks across the 6 classification benchmarks.

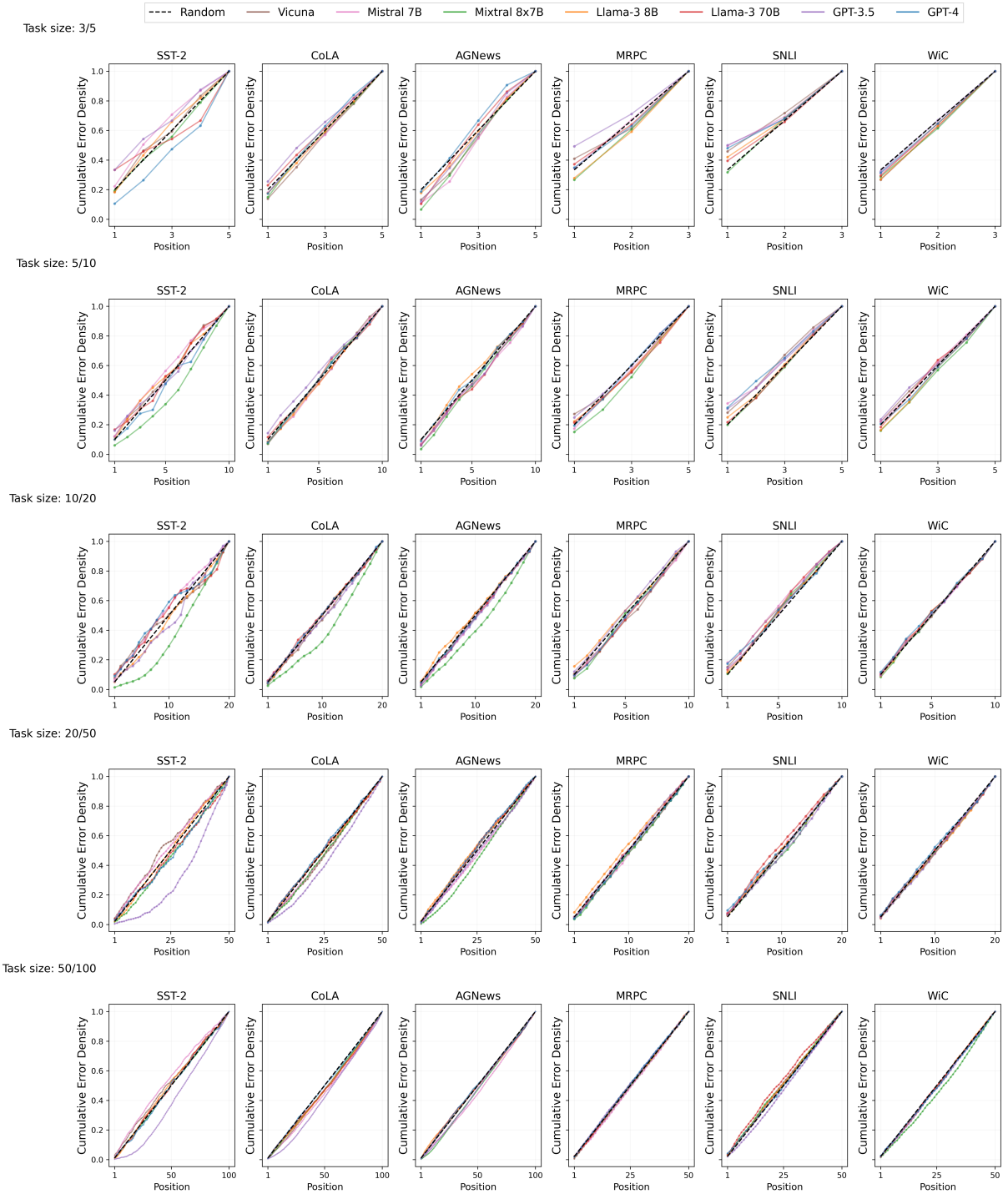


Figure 9: Cumulative error density across positions for all benchmarks and LLMs across different task sizes. The task size "M/N" on the left side of the plots denotes the task size for the 3 text-pair benchmarks (i.e., MRPC, SNLI, and WiC) and for the 3 single-text benchmarks (i.e., SST-2, CoLA, AGNews), respectively.

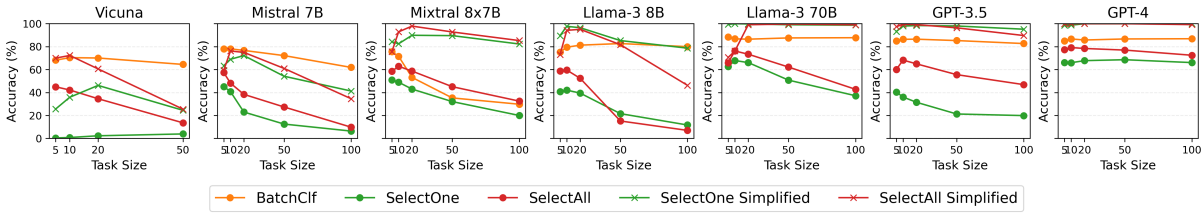


Figure 10: Full results for the two simplified index selection tasks versus the original tasks based on AGNews.

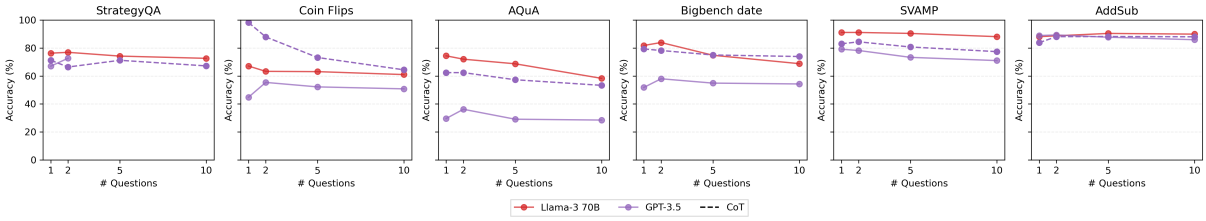


Figure 11: Average accuracy of GPT-3.5 and Llama-3 70B on the other 6 reasoning benchmarks with multiple single-source problems. We leave out results where the number of parsable outputs is less than 50, e.g., GPT-3.5 on StrategyQA at task size > 2.

| | # P = 1 | # P = 2 | Avg # Answers |
|---|---|---|---|
| Llama-3 8B (Instruct) | 46.8 | 50.0 | 2.1 |
| GPT-3.5 | 44.8 | 55.4 | 2.0 |
| Llama-3 8B (Base) | 45.9 (20.4) | 33.7 | 2.7 |
| GPT-3 1.3B | 49.0 | 45.0 | 6.3 |
| GPT-3 175B | 50.0 (43.4) | 28.4 | 6.6 |
| FLAN-T5-Large (0.78B) | 46.6 | NA | 1.0 |
| FLAN-T5-XL (3B) | 49.4 | NA | 1.0 |
| FLAN-T5-XXL (11B) | 57.2 | NA | 1.0 |

Table 7: Accuracy (%) of three pretrained LLMs and three FLAN-T5 models on Coin Flips with 1 and 2 problems. "# P": number of problems. When computing accuracy, we treat an LLM response with more than 2 answers as a wrong answer. The numbers in the two parentheses are accuracies when treating "uncertain" answers as wrong answers, instead of discarding them.

SelectAll remains in virtually all cases. We leave to future studies the investigation of how CoT may further help with the two selection tasks.

## C MPP: Reasoning

This section provides additional details for Section 4. The prompt details for the experiments can be found in Appendix E.

### C.1 Construction of Mixed-Source Prompts

We create 6 distinct benchmark combinations based on the 12 benchmarks, each consisting of 10 different benchmarks. When creating these 6 benchmark combinations, we implement the following 2 rules: (1) the first 2 benchmarks must be different across the 6 combinations to cover the 12 benchmarks; (2) the first 2 benchmarks cannot come from SVAMP, GSM8K, MultiArith, AddSub, and SingleEq to maximize the differences between them.

### C.2 More Single-Source Results

Fig 11 shows MultiReason$^{SS}$ results on the other 6 reasoning benchmarks not presented in Fig 6.

## D Additional Results for Section 5

This section describes the results of the three pretrained base LLMs and three FLAN-T5 models on Coin Flips at task sizes 1 and 2 from Section 5, shown in Table 7. Similar to Table 3, Table 7 shows that instruction-tuned models perform much better on multi-problem prompts and that FLAN-T5 models can barely handle multi-problem prompts.

However, after manual inspection, we find that the outputs of the three pretrained models, are often not very sensical with repetitions of the prompts (either partially or entirely). In particular, as shown in Table 7, the two GPT-3 models tend to produce more answers than needed (we set max output tokens to be 200). The answer can also be nonsensical even when well formatted, such as the example output from GPT-3 175B below. Llama-3 8B (Base) overall produce more meaningful outputs, but similar problems also apply. We thus discourage over-interpreting the results in Table 7.

> Q1: Yes. The coin is still heads up.
>
> Q2: No. The coin is now tails up.
>
> Explanation
>
> Q1: The coin is still heads up because the coin is not flipped. The coin is flipped only when someone flips the coin. If no one flips the coin, the coin is not flipped.
>
> Q2: The coin is now tails up because the coin is flipped. The coin is flipped only when someone flips the coin. If no one flips the coin, the coin is not flipped.
>
> Discussion
>
> This is a very interesting question. It is a question about the meaning of "flip". The meaning of "flip" is not clear. It is not clear whether "flip" means "reverse" or "flip". It is not clear whether "flip" means "flip" or "flip"...{The last sentence repeated for three more times.}...

| Model | Benchmark Task | SST-2 | CoLA | AGNews | MRPC | SNLI | WiC |
|---|---|---|---|---|---|---|---|
| Vicuna | SelectOne | 38.5 | 22.2 | 16.8 | 15.8 | 9.9 | 34.5 |
| | + CoT | 52.6 | 53.9 | 12.1 | 12.1 | 11.5 | 12.5 |
| | SelectAll | 66.7 | 65.7 | 42.2 | 67.4 | 28.2 | 48.8 |
| | + CoT | 81.7 | 69.2 | 40.9 | 63.9 | 40.9 | 48.9 |
| | BatchClf | 89.1 | 73.2 | 70.4 | 66.7 | 59.3 | 53.4 |
| Mistral 7B | SelectOne | 53.3 | 52.8 | 41.5 | 36.6 | 23.3 | 22.2 |
| | + CoT | 66.5 | 55.1 | 44.8 | 59.4 | 29.5 | 29.5 |
| | SelectAll | 68.3 | 63.6 | 48.2 | 58.4 | 54.7 | 47.9 |
| | + CoT | 74.4 | 68.2 | 57.0 | 68.5 | 47.6 | 47.3 |
| | BatchClf | 79.6 | 76.8 | 78.1 | 74.0 | 69.1 | 49.6 |
| Mixtral 8x7B | SelectOne | 76.5 | 61.0 | 46.2 | 45.2 | 23.8 | 32.0 |
| | + CoT | 41.5 | 67.3 | 12.9 | 48.4 | 25.1 | 37.2 |
| | SelectAll | 85.3 | 68.4 | 62.9 | 65.9 | 56.2 | 52.0 |
| | + CoT | 49.0 | 67.4 | 57.8 | 66.5 | 58.9 | 51.1 |
| | BatchClf | 80.2 | 80.9 | 71.6 | 70.8 | 73.3 | 63.7 |
| Llama-3 8B | SelectOne | 67.8 | 50.8 | 55.1 | 21.7 | 21.4 | 17.3 |
| | + CoT | 85.0 | 48.9 | 22.1 | 50.7 | 22.3 | 22.0 |
| | SelectAll | 85.0 | 62.9 | 59.6 | 64.8 | 50.1 | 52.2 |
| | + CoT | 85.8 | 66.5 | 45.3 | 64.5 | 39.8 | 52.3 |
| | BatchClf | 92.0 | 76.8 | 79.5 | 62.8 | 69.2 | 59.5 |
| Llama-3 70B | SelectOne | 83.6 | 72.7 | 67.4 | 61.0 | 32.3 | 39.3 |
| | + CoT | 94.7 | 79.7 | 75.2 | 67.9 | 69.4 | 15.8 |
| | SelectAll | 87.3 | 79.6 | 76.2 | 71.7 | 68.6 | 53.9 |
| | + CoT | 96.1 | 83.7 | 85.5 | 73.9 | 77.3 | 58.4 |
| | BatchClf | 96.4 | 82.8 | 86.8 | 73.5 | 75.2 | 61.7 |
| GPT-3.5 | SelectOne | 71.9 | 65.8 | 47.6 | 25.2 | 28.4 | 33.8 |
| | + CoT | 80.2 | 71.7 | 57.9 | 26.8 | 26.5 | 22.1 |
| | SelectAll | 83.1 | 79.7 | 68.3 | 61.5 | 55.8 | 51.6 |
| | + CoT | 89.5 | 77.5 | 58.8 | 70.2 | 50.5 | 51.2 |
| | BatchClf | 95.0 | 81.8 | 86.4 | 52.7 | 57.4 | 58.0 |
| GPT-4 | SelectOne | 88.6 | 80.8 | 66.4 | 58.1 | 53.8 | 50.0 |
| | + CoT | 84.5 | 81.9 | 67.8 | 60.1 | 77.0 | 59.6 |
| | SelectAll | 92.4 | 84.0 | 79.2 | 73.5 | 80.8 | 62.8 |
| | + CoT | 91.3 | 83.7 | 85.4 | 76.2 | 83.3 | 71.0 |
| | BatchClf | 96.0 | 84.0 | 86.7 | 72.6 | 83.8 | 68.4 |
| Overall | SelectOne | 68.6 | 58.0 | 48.7 | 37.7 | 27.6 | 32.7 |
| | + CoT | 72.1 | 65.5 | 41.8 | 46.5 | 37.3 | 28.4 |
| | SelectAll | 81.2 | 72.0 | 62.4 | 66.2 | 56.3 | 52.7 |
| | + CoT | 81.1 | 73.7 | 61.5 | 69.1 | 56.9 | 54.3 |
| | BatchClf | 89.8 | 79.5 | 79.9 | 67.6 | 69.6 | 59.2 |

Table 8: Full 1-shot-CoT results for all benchmarks and LLMS. The task size is fixed at 10.

# E Full Prompts

## E.1 Prompt Templates for SingleClf, BatchClf, SelectOne, and SelectAll

Tables 13 to 14 show the complete prompt templates for the four task types (i.e., SingleClf, BatchClf, SelectOne, and SelectAll) tailored for SST-2, CoLA, AGNews, MPRC, SNLI, and WiC, respectively. While there are differences in the exact wording of a prompt template for each task type across the 6 classification benchmarks, each prompt template type shares a similar underlying structure and can be easily applied to other classification benchmarks.

Throughout our research project, we have also tried prompts with different wordings and structures until we finally unified the prompt designs presented above. For example, we initially asked LLMs to directly generate indices line by line instead of a JSON output for SelectOne and we did not provide any formatted example for SelectAll. We also put the output format instruction in the end of each prompt for SelectAll, instead of in the beginning. Although we observed certain task-level performance variations, which are expected, the overall complexity among the 4 task types (SelectOne > SelectAll > BatchClf > SingeClf) remains unchanged, despite the variations in the prompts. This indicates the overall limited effects of rewording and restructuring prompts.

## E.2 Prompt Templates for the Two Index Selection Tasks

Table 15 presents the two prompt templates used for the two simplified index selection tasks based on AGNews.

## E.3 One-Shot-CoT Prompt Templates for the Two Index Selection Tasks

To construct a 1-shot-CoT exemplar, we first randomly sampled 10 examples from each benchmark that do not appear in our test examples. We then put these 10 examples into the prompt template for SelectOne or SelectAll, followed by a constructed answer. The answer first explicitly classifies the 10 examples into the corresponding class labels, and then perform index selection based on the generated class labels.

Table 16 shows the complete answer templates for SelectOne and SelectAll for each classification benchmark.

## E.4 Prompt Template for Multi-problem Prompts for Reasoning Problems

The prompt template for a multi-problem prompt made up of reasoning problems is straightforward, as described in Section 4. Below is a simple example prompt made up of 2 reasoning problems from CommonsenseQA.

> Questions
>
> Q1: The person wasn't bothered by the weather, she had remembered to bring her what?
> Answer Choices: (A) read book (B) own house (C) apartment (D) more rice (E) warm coat
>
> Q2: After working on the car, what did it end up doing?
> Answer Choices: (A) going too fast (B) last several years (C) honk the horn (D) go fast (E) start running
>
> Answers

To enable zero-shot-CoT, we simply append the string "Let's think step by step." (Kojima et al., 2022) to a zero-shot prompt like the one shown above in a newline (after "Answers").

| Task | Prompt template |
|---|---|
| SingleClf | Indicate the sentiment for the following line of text. The sentiment shall be either 'Positive' or 'Negative.'<br><br>Text: $text<br>Sentiment: |
| BatchClf | Indicate the sentiment for each of the $num following lines of text. The sentiment shall be either 'Positive' or 'Negative.'<br><br>Texts, one per line:<br><br>$texts<br><br>The sentiments for each of the $num lines of text, one per line: |
| SelectOne | Go over the $num lines of text below and list the index numbers of the lines with $polarity sentiment according to the following instructions: If none of the texts show $polarity sentiment, write 'None.' If all the texts show $polarity sentiment, write 'All.' Otherwise, provide the index numbers for each text with $polarity sentiment.<br><br>Output your responses in JSON format with the key '$polarity'. A formatted example output is provided below.<br>{'$polarity': [None/All or index numbers for the texts with $polarity sentiment]}<br><br>Texts, one per line:<br><br>$texts<br><br>JSON output: |
| SelectAll | Go over the $num lines of text below. First, list the index numbers of the lines with positive sentiment. Then, list the index numbers of the lines with negative sentiment. If none of the texts show a particular sentiment, write 'None.' If all the texts show a particular sentiment, write 'All.' Otherwise, provide the index numbers of the texts that fit a particular category.<br><br>Output your responses in JSON format with two keys: 'positive' and 'negative.' A formatted example output is provided below.<br>{'positive': [None/All or index numbers of positive sentences], 'negative': [None/All or index numbers of negative sentences]}<br><br>Texts, one per line:<br><br>$texts<br><br>JSON output: |

Table 9: Prompt templates for SST-2. Words immediately preceded by the dollar sign $ are placeholders. For the single-text classification task (SST-2, CoLA, AGNews), the sequence of texts in the place of '$texts' are indexed starting with '1' and each text is separated by a newline.

| Task | Prompt template |
| --- | --- |
| SingleClf | Indicate the grammatical acceptability for the following line of text. The acceptability shall be either 'Acceptable' or 'Unacceptable.'<br><br>Text: $text<br>Grammatical acceptability: |
| BatchClf | Indicate the grammatical acceptabilities for each of the $num following lines of text. The acceptability shall be either 'Acceptable' or 'Unacceptable.'<br><br>Texts, one per line:<br><br>$texts<br><br>Grammatical acceptabilities for each of the $num lines of text, one per line: |
| SelectOne | Go over the $num lines of text below and list the index numbers of the lines that are grammatically $acceptability according to the following instructions: If none of the texts are grammatically $acceptability, write 'None.' If all the texts are grammatically $acceptability, write 'All.' Otherwise, provide the index numbers for each grammatically $acceptability text.<br><br>Output your responses in JSON format with the key '$acceptability'. A formatted example output is provided below.<br>{'$acceptability': [None/All or index numbers of $acceptability sentences]}<br><br>Texts, one per line:<br><br>$texts<br><br>JSON output: |
| SelectAll | Go over the $num lines of text below. First, list the index numbers of the lines that are grammatically acceptable. Then, list the index numbers of the lines that are grammatically unacceptable. If none of the sentences show a particular acceptability, write 'None.' If all the sentences show a particular acceptability, write 'All.' Otherwise, provide the index numbers of the texts that fit a particular category.<br><br>Output your responses in JSON format with two keys 'acceptable' and 'unacceptable.' A formatted example output is provided below.<br>{'acceptable': [None/All or index numbers of acceptable texts], 'unacceptable': [None/All or index numbers of unacceptable texts]}<br><br>Texts, one per line:<br><br>$texts<br><br>JSON output: |

Table 10: Prompt templates for CoLA.

| Task | Prompt template |
|---|---|
| SingleClf | Classify which news category the following line of text belongs to among the following four categories: 'Business,' 'Sports,' 'World,' and 'Sci/Tech.'<br><br>Text: $text<br>News category: |
| BatchClf | Classify which news category each of the $num following lines of text belongs to among the following four categories: 'Business,' 'Sports,' 'World,' and 'Sci/Tech.'<br><br>Texts, one per line:<br><br>$texts<br><br>News categories for each of the $num lines of text, one per line: |
| SelectOne | This is a news classification task in which each line of text belongs to one of four categories 'Business,' 'Sports,' 'World,' and 'Sci/Tech.'<br><br>Go over the $num lines of text below and list the index numbers of the lines that can be classified as $category according to the following instructions: If none of the texts can be classified as $category, write 'None.' If all the texts can be classified as $category, write 'All.' Otherwise, provide the index numbers of the texts that can be classified as $category.<br><br>Output your responses in JSON format with the key '$category'. A formatted example output is provided below.<br>{'$category': [None/All or index numbers of the texts that can be classified as $category]}<br><br>Texts, one per line:<br><br>$texts<br><br>JSON output: |
| SelectAll | This is a news classification task in which each line of text belongs to one of four categories 'Business,' 'Sports,' 'World,' and 'Sci/Tech.'<br><br>Go over the $num lines of text below and list the index numbers of the lines that belong to each category according to the following instructions: If none of the texts can be classified as a particular category, write 'None.' If all the texts can be classified as a particular category, write 'All.' Otherwise, provide the index numbers of the texts that can be classified as the category.<br><br>Output your responses in JSON format with the following keys: 'business,' 'sports,' 'world,' and 'sci/tech.' A formatted example output is provided below.<br>{'business': [None/All or index numbers of texts in 'business' category], 'sports': [None/All or index numbers of texts in 'sports' category], 'world': [None/All or index numbers of texts in 'world' category], 'sci/tech': [None/All or index numbers of texts in sci/tech category]}<br><br>Texts, one per line:<br><br>$texts<br><br>JSON output: |

Table 11: Prompt templates for AGNews.

| Task | Prompt template |
|---|---|
| SingleClf | Compare text A with text B and determine if text A is a paraphrase of text B. Respond with 'Yes' if text A is a paraphrase, and 'No' if it is not.<br><br>$text<br>Answer: |
| BatchClf | Compare text A with text B for the following $num text pairs and determine if text A is a paraphrase of text B line by line. Respond with 'Yes' if text A is a paraphrase, and 'No' if it is not. Provide your answers line by line.<br><br>$texts<br>Answers: |
| SelectOne | Go over the $num text pairs below and list the index numbers of the text pairs where text A $be a paraphrase of text B according to the following instructions: If none of the text pairs satisfy this condition, write 'None.' If all the text pairs satisfy this condition, write 'All.' Otherwise, provide the index numbers of the text pairs where text A $be a paraphrase of text B.<br><br>Output your responses in JSON format with the key 'answer'. A formatted example output is provided below.<br>{'answer': [None/All or index numbers of the text pairs where text A $be a paraphrase of text B]}<br><br>Here are the text pairs:<br><br>$texts<br>JSON output: |
| SelectAll | Go over the $num text pairs below. First, list the index numbers of the text pairs that contain paraphrases. Then, list the index numbers of the text pairs that contain non-paraphrases. If none of the text pairs satisfy a condition, write 'None.' If all the text pairs satisfy a condition, write 'All.' Otherwise, provide the index numbers of the text pairs that satisfy each condition.<br><br>Output your responses in JSON format with two keys: 'yes' for paraphrases and 'no' for non-paraphrases. A formatted example output is provided below.<br>{'yes': [None/All or index numbers of text pairs that contain paraphrases], 'no': [None/All or index numbers of text pairs that contain non-paraphrases]}<br><br>Here are the text pairs:<br><br>$texts<br>JSON output: |

Table 12: Prompt templates for MRPC. For the text-pair classification task (MRPC, SNLI, WiC), the sequence of text pairs in the place of '$texts' are indexed starting with '1' and each text pair is separated by two newlines (each text pair ends with a newline be design, followed by another newline before the next text pair).

| Task | Prompt template |
|---|---|
| SingleClf | Given the following premise and hypothesis, determine the inference relation between them. Respond with 'Entailment' if the hypothesis logically follows from the premise, 'Contradiction' if they are in direct opposition, and 'Neutral' if neither applies.<br><br>$text<br>Inference relation: |
| BatchClf | Given the following $num pairs of premises and hypotheses, determine the inference relation for each pair line by line. Respond with 'Entailment' if the hypothesis entails the premise, and 'Contradiction' if they contradict. If neither is the case, respond with 'Neutral.' Provide your answers line by line.<br><br>$texts<br>Inference relations for the $num text pairs provided above: |
| SelectOne | Go over the $num text pairs below and list the index numbers of the text pairs where the inference relation between the premise and the hypothesis is $relationship according to the following instructions: If none of the text pairs contain $relationship inference relation, write 'None.' If all text pairs contain $relationship inference relation, write 'All.' Otherwise, provide the index numbers of the text pairs where the inference relation between the premise and the hypothesis is $relationship.<br><br>Output your responses in JSON format with the key '$relationship'. A formatted example output is provided below.<br>'$relationship': [None/All or index numbers of text pairs that contain $relationship inference relation]<br><br>Here are the text pairs:<br><br>$texts<br>JSON output: |
| SelectAll | Go over the $num text pairs below. First, list the index numbers of the text pairs that contain entailment inference relation. Then, select all text pairs that contain contradiction inference relation. Finally, select all text pairs that contain neutral inference relation. If none of the text pairs satisfy a condition, write 'None.' If all the text pairs belong satisfy a condition, write 'All.' Otherwise, provide the index numbers of the text pairs that satisfy each condition.<br><br>Output your responses in JSON format with three keys: 'entailment', 'contradiction', and 'neutral'. A formatted example output is provided below.<br>{'entailment': [None/All or index numbers of text pairs that contain entailment inference relation], 'contradiction': [None/All or index numbers of text pairs that contain contradiction inference relation], 'neutral': [None/All or index numbers of text pairs that contain neutral inference relation]}<br><br>Here are the text pairs:<br><br>$texts<br>JSON output: |

Table 13: Prompt templates for SNLI.

| Task | Prompt template |
|---|---|
| SingleClf | Analyze the usage of the given target word in the two subsequent contexts. The target word may appear in various grammatical forms in each context. Respond with 'Yes' if it maintains the same meaning across both contexts, and 'No' if it does not.<br><br>$text<br>Answer: |
| BatchClf | Analyze the usage of the following $num target words in the two contexts that immediately follow them. These target words may appear in different grammatical forms across the two subsequent contexts. Determine if each target word maintains the same meaning in the two subsequent contexts. Provide your answers line by line, indicating 'Yes' if it does and 'No' if it does not.<br><br>$texts<br>Answers: |
| SelectOne | Analyze the following $num target words and determine the index numbers of the target words where the same meaning $be maintained across the two contexts that immediately follow them. These target words may appear in different grammatical forms in each context. If none of the target words satisfy this condition, write 'None.'. If all the target words satisfy this condition, write 'All.' Otherwise, provide the index numbers.<br><br>Output your responses in JSON format with the key 'answer'. A formatted example output is provided below.<br>{'answer': [None/All or index numbers of the target words where the same meaning $be maintained in the two subsequent contexts]}<br><br>Here are the target words along with their contexts:<br><br>$texts<br>JSON output: |
| SelectAll | Analyze the following $num target words, which may appear in different grammatical forms in the two subsequent contexts. First, list the index numbers of target words that maintain the same meaning in the two subsequent contexts. Then, list the index numbers of target words that do not maintain the same meaning in the two subsequent contexts. If none of the target words satisfy a condition, write 'None.' If all the target words satisfy a condition, write 'All.' Otherwise, provide the index numbers of the target words that satisfy each condition.<br><br>Output your responses in JSON format with two keys: 'yes' for target words used with consistent meanings and 'no' for those used with inconsistent meanings. A formatted example output is provided below.<br>{'yes': [None/All or index numbers of target words used with consistent meanings], 'no': [None/All or index numbers of target words used with inconsistent meanings]}<br><br>Here are the target words along with their contexts:<br><br>$texts<br>JSON output: |

Table 14: Prompt templates for WiC.

| Task | Prompt template |
|---|---|
| SelectOne Simplified | Go over the $num lines of text below and list the index numbers of the lines that contain the word '$category' according to the following instructions: If none of the texts contain the word '$category,' write 'None.' If all the texts contain the word '$category,' write 'All.' Otherwise, provide the index numbers of the texts that contain the word '$category' each on a separate line.<br><br>Texts, one per line:<br><br>$texts<br><br>'None,' 'All,' or the index numbers of the texts that contain the word '$category,' one per line: |
| SelectAll Simplified | In this task, each line of text contains one of four words 'Business,' 'Sports,' 'World,' and 'Sci/Tech.'<br><br>Go over the $num lines of text below and list the index numbers of the lines that contain each word according to the following instructions: If none of the texts contain a particular word, write 'None.' If all the texts contain a particular word, write 'All.' Otherwise, provide the index numbers of the texts that contain each word.<br><br>Output your responses in JSON format with the following keys: 'business,' 'sports,' 'world,' and 'sci/tech.' A formatted example output is provided below.<br>{'business': [None/All or index numbers of texts containing 'Business'], 'sports': [None/All or index numbers of texts containing 'Sports'], 'world': [None/All or index numbers of texts containing 'World'], 'sci/tech': [None/All or index numbers of texts containing 'Sci/Tech']}<br><br>Texts, one per line:<br><br>$texts<br><br>JSON output: |

Table 15: Prompt templates for simplified SelectOne and SelectAll based on AGNews. For SelectOne Simplified, we used an earlier prompt template that asks LLMs to produce outputs line by line, instead of a JSON object. As illustrated earlier, we find the effect of the two output formats for SelectOne to be minimal. Therefore, having a JSON output should not make a meaningful difference.

| Benchmark | Task | Answer template |
|---|---|---|
| SST-2 | SelectOne | To solve this task, let's first classify the 10 lines of text above, one per line: {labels}<br>From here, we see that lines {indices} show positive sentiment. Therefore, the answer in JSON format is as follows: {final answer} |
| SST-2 | SelectAll | To solve this task, let's first classify the 10 lines of text above, one per line: {labels}<br>From here, we see that lines {indices} show positive sentiment, and lines {indices} show negative sentiment. Therefore, the answer in JSON format is as follows: {final answer} |
| CoLA | SelectOne | To solve this task, let's first classify the 10 lines of text above, one per line: {labels}<br>From here, we see that texts in lines {indices} are unacceptable. Therefore, the answer in JSON format is as follows: {final answer} |
| CoLA | SelectAll | To solve this task, let's first classify the 10 lines of text above, one per line: {labels}<br>From here, we see that texts in lines {indices} are acceptable and texts in lines {indices} are unacceptable. Therefore, the answer in JSON format is as follows: {final answer} |
| AGNews | SelectOne | To solve this task, let's first classify the 10 lines of text above, one per line: {labels}<br>From here, we see that texts in lines {indices} can be classified as 'Sports.'. Therefore, the answer in JSON format is as follows: {final answer} |
| AGNews | SelectAll | To solve this task, let's first classify the 10 lines of text above, one per line: {labels}<br>From here, we see that texts in lines {indices} can be classified as 'Business,' texts in lines {indices} can be classified as 'Sports,' texts in lines {indices} can be classified as 'World,' and texts in lines {indices} are 'Sci/Tech.' Therefore, the answer in JSON format is as follows: {final answer} |
| MRPC | SelectOne | To solve this task, let's first determine if text A is a paraphrase of text B for the 10 lines of text above, one per line: {labels}<br>From there, we see that text pairs in lines {indices} are paraphrases. Therefore, the answer in JSON format is as follows: {final answer} |
| MRPC | SelectAll | To solve this task, let's first determine if text A is a paraphrase of text B for the 10 lines of text above, one per line: {labels}<br>From there, we see that text pairs in lines {indices} are paraphrases and text pairs in lines {indices} are not. Therefore, the answer in JSON format is as follows: {final answer} |
| SNLI | SelectOne | To solve this task, let's first determine the inference relation between the premise and the hypothesis for the 10 lines of text above, one per line: {labels}<br>From there, we see that text pairs in lines {indices} contain entailment inference relation. Therefore, the answer in JSON format is as follows: {final answer} |
| SNLI | SelectAll | To solve this task, let's first determine the inference relation between the premise and the hypothesis for the 10 lines of text above, one per line: {labels}<br>From there, we see that text pairs in lines {indices} contain entailment inference relation, text pairs in lines {indices} contain contradiction inference relation, and text pairs in lines {indices} contain neutral inference relation. Therefore, the answer in JSON format is as follows: {final answer} |
| WiC | SelectOne | To solve this task, let's first determine if the target word is used with consistent meanings in the two subsequent contexts for the num lines of text above, one per line: {labels}<br>From there, we see that text pairs in lines {indices} contain use the target words with inconsistent meanings. Therefore, the answer in JSON format is as follows: {final answer} |
| WiC | SelectAll | To solve this task, let's first determine if the target word is used with consistent meanings in the two subsequent contexts for the num lines of text above, one per line: {labels}<br>From there, we see that text pairs in lines {indices} contain use the target words with inconsistent meanings and text pairs in lines {indices} do not. Therefore, the answer in JSON format is as follows: {final answer} |

Table 16: Answer templates for the two selection tasks for each classification benchmark. {indices}: a list of indices seperated by comma. {final answer}: answer in a JSON format specified by the SelectOne or SelectAll prompt.