BLT: Can Large Language Models Handle Basic Legal Text?

Andrew Blair-Stanek[1,2] **Nils Holzenberger**[3] **Benjamin Van Durme**[1]
[1] Johns Hopkins University, [2] University of Maryland School of Law,
[3] Télécom Paris - Institut Polytechnique de Paris

Abstract

We find that the best publicly available LLMs like GPT-4 and Claude currently perform poorly on basic legal text handling. This motivates the creation of a benchmark consisting of examples that lawyers and paralegals would expect LLMs to handle zero-shot, such as looking up the text at a line of a witness deposition or at a subsection of a contract. LLMs' poor performance on this benchmark casts into doubt their reliability as-is for legal practice. However, fine-tuning on our training set brings even a small model to near-perfect performance. This benchmark will be useful for fine-tuning LLMs for downstream legal tasks, as well as for tracking LLMs' reliability as-is for basic legal tasks.

1 Introduction

There has been much excitement about the potential of large language models (LLMs) to handle complex reasoning tasks. In the legal domain, recent work has focused on LLMs' ability to perform high-level tasks such as passing the bar exam, identifying hearsay evidence, or navigating the tax code. These results have led to predictions that LLMs will imminently revolutionize law practice.

However, no prior work has investigated LLMs' ability to handle the most very basic text tasks done in daily legal practice. In this paper, we introduce a benchmark for **B**asic **L**egal **T**ext (BLT) handling. BLT mimics simple tasks done by lawyers and paralegals, including looking up citations to the record, finding statutory citations containing particular text or definitions, or finding the text in a contract that another lawyer has explicitly cited.

BLT is designed so that a paralegal, new lawyer, or even non-subject matter expert could perform zero-shot at or near 100%. But we find that leading publicly available LLMs perform poorly out-of-the-box. An example of this poor performance is in Figure 1, where GPT-4o fails to retrieve the text on the specified line of a deposition transcript excerpt that is a mere 15 lines long.

1: MR. MACEJKA: Object to form.
2: A. The medical invoice from the MAO would be
3: the medical claims data. So we're already in
4: receipt of that.
5: BY MR. PENNEKAMP:
6: Q. That's what you referred to a medical
7: invoice, is the claims data?
8: MR. MACEJKA: Object to form.
9: A. That essentially is what the medical
10: claims data is.
11: BY MR. PENNEKAMP:
12: Q. Do you ask the providers for their
13: invoices for the medical services?
14: MR. MACEJKA: Object to form.
15: A. We will request for EOBs or billing

What is the exact text of just line 7 above?

The exact text of line 7 is: "Q. That's what you referred to a medical"

Figure 1: GPT-4o incorrectly answering a simple question about a page from a line-numbered witness deposition transcript. Line numbers, exactly as shown above, are passed to GPT-4o, which incorrectly answers 13% of such one-page, 15-line deposition retrieval prompts.

Yet we find that fine-tuning a less-advanced model, GPT-3.5-turbo, on BLT's training set leads to near-human (i.e., 100%) performance. We also find that this fine-tuned model also performs better on a more complex legal task, demonstrating BLT's value for fine-tuning LLMs for legal applications.

The BLT dataset has additional strengths. Our code[1] can generate unlimited new examples never before seen in corpora. Moreover, BLT is scalable to different window sizes and is one of the few datasets taking full advantage of LLMs with window sizes of 64k or 128k tokens.

[1] https://github.com/blairstanek/blt.

Proceedings of the Natural Legal Language Processing Workshop 2024, pages 216–232

2 Background

Law is a largely text-based profession and thus is often used to demonstrate LLMs' capabilities. OpenAI's GPT-4 technical report (OpenAI, 2023b) mentioned only a single benchmark in the abstract itself: GPT-4 had passed the bar exam, with the score in the 90th percentile.

OpenAI's developer livestream by co-founder Greg Brockman introducing GPT-4 (OpenAI, 2023a) used four examples to show GPT-4's capabilities. One involved U.S. tax law, where Brockman prompted GPT-4 with several tax-code sections and had it calculate the taxes of hypothetical taxpayers Alice and Bob. Brockman proclaimed that GPT-4 can "do taxes."

2.1 Legal Use of LLMs

Legal NLP is concerned with a diverse range of tasks, reflecting the diversity of tasks lawyers perform. Examples include legal judgment prediction (Medvedeva and McBride, 2023; Chalkidis et al., 2019; Xiao et al., 2018), contract review (Hendrycks et al., 2021), document review (Lewis et al., 2023), and retrieving relevant case law (Kim et al., 2022). There has been extensive discussion of how NLP can benefit the legal system (Zhong et al., 2020; Aidid and Alarie, 2023). LLMs have been deployed for a wide range of legal tasks, including case analysis (Savelka et al., 2023), discovery (Pai et al., 2023), and analyzing contracts (Roegiest et al., 2023).

Several LLMs have been fine-tuned on legal materials (Colombo et al., 2024) and for legal tasks. Dominguez-Olmedo et al. (2024) discovered that for some legal tasks, a 8-billion-parameter LLM that has been lightly pretrained on legal tasks substantially outperforms GPT-4, which has several orders of magnitude more parameters.

Many legal benchmarks for LLMs have been created (Chalkidis et al., 2022; Fei et al., 2023), with many incorporated into the broad LegalBench project (Guha et al., 2022). These are all much higher-level tasks than BLT, including identifying testimony to which the hearsay doctrine applies or whether contractual terms impose particular restrictions (Hendrycks et al., 2021).

The SARA (StAtutory Reasoning Assessment) dataset (Holzenberger et al., 2020) is one of the higher-level tasks in LegalBench. It consists of nine tax-related sections of the U.S. Code, plus 376 hand-crafted "cases" consisting of facts and a question that can be unambiguously answered applying the nine sections to the facts. Because SARA is a higher-level task clearly predicated on lower-level text handling (specifically, finding text at a citation), we use it to measure the effectiveness of fine-tuning with BLT, discussed in Section 5.

The ability of GPT-3 to handle SARA was evaluated in Blair-Stanek et al. (2023), with lackluster performance found. Qualitatively, GPT-3 often retrieved text from the wrong part of the given statute. For example, GPT-3 was prompted with the text of the U.S. tax code's section 152, some facts about Alice and Bob, and the question of whether Alice's relationship to Bob fell under section 152(d)(2)(C). GPT-3's response analyzed the question using the text of section 152(d)(2)(**D**), with the result that GPT-3 answered the question about Alice and Bob incorrectly. This inability to retrieve clearly specified text – resulting in incorrect answers to legal questions – was a motivation behind the BLT dataset and this paper.

LLMs have seen much of the internet during their training. To evaluate LLMs' ability to handle novel legal questions, Nay et al. (2023) generate synthetic multiple-choice legal questions. Similarly, to test whether LLMs can handle truly novel legal texts, Blair-Stanek et al. (2023) generate synthetic sections constructed with nonces (phonetically plausible nonsense words) and probe GPT-3's ability to reason over these synthetic sections. BLT incorporates such synthetic sections, albeit for simpler tasks than statutory reasoning.

2.2 Related LLM Evaluations

Numerous evaluation metrics have been developed for LLMs (Shahriar et al., 2024; Chang et al., 2023). For example, BIG-Bench (Srivastava et al., 2023) includes basic word handling tasks like word sorting and text editing. Parsing software logs is evaluated by (Le and Zhang, 2023). Simplifying complex sentences is evaluated in (Wu and Arase, 2024). LLMs can solve quite complicated tasks by being prompted to provide a chain of thought (Kojima et al., 2022), including in the legal domain (Yu et al., 2022). More generally, choosing the appropriate way to prompt LLMs, called prompt engineering, often has a substantial impact on LLM performance (White et al., 2023; Liu et al., 2023b). A complementary approach has been to decompose the task at hand into tasks the LLM can handle (Dua et al., 2022; Khot et al., 2023). LLMs have been evaluated in various professional domains, including medicine (Beaulieu-Jones et al., 2024) and accounting (Zhao and Wang, 2024).

LLMs have been trained or otherwise induced to use "Tools" (Schick et al., 2023; Paranjape et al., 2023). For example, an LLM might detect that it needs to call a calculator tool to handle a math problem posed to it in text form. In theory, tools could be written to handle the BLT tasks and then be integrated into LLMs. But the BLT prompts are oversimplified versions of tasks lawyers need LLMs to do seamlessly. A lawyer will not ask an LLM for the citation to the record where the plaintiff says "I have therapy tomorrow." But a lawyer might expect an LLM to insert a citation to the record that proves a plaintiff receives psychological care, and a basic text-matching tool would not handle that.

Some of BLT's prompts are quite long, which is realistic because lawyers regularly handle long texts. Liu et al. (2023a) investigated how LLMs handle retrieval from long prompts. They found that LLMs' accuracy followed a U-curve with respect to the information's position, with information in the middle of the prompt used much less than if it were at the start or end. They connected this to the "serial-position" effect exhibited by humans, who best remember material presented near the beginning or end.

3 The BLT Benchmark

The BLT benchmark involves three different types of legal text, each of which has between two and five different tasks run on it.

3.1 Deposition Transcripts

In litigation in the U.S., depositions of witnesses under oath are a key factfinding tool. The depositions typically occur in lawyers' offices and allow lawyers to ask witnesses questions on virtually any topic. Professional court reporters transcribe the depositions into transcripts, typically with 25 numbered lines per page, often running over 100 pages for a single witness deposition. Attorneys must cite relevant portions of the resulting transcripts in subsequent motions, such as those asking the court to grant their side summary judgment. Portions of transcripts are cited by page and line number.

One basic legal text-handling task a lawyer must do is finding the page and line of a transcript where particular text appears. This motivates the **text→cite** task, where the prompt consists of one or more pages of actual deposition transcript followed by the question, "What are the page number and line number of the line above with the text "__"?" (after single pages, the prompt does not ask for the page number). To ensure there is only one clearly correct answer, prompts are not constructed asking about lines with less than four words, that are subsets of another line, or that are too similar to other lines (defined as Levenshtein distance under four (Levenshtein et al., 1966)).

The converse is another basic text-handling task: given a citation to a transcript, find the text at the cited location. Lawyers must do this basic task in order to evaluate the opposing side's motions. Paralegals do it on their side's own motions before submitting them (ProParalegal, 2017). This motivates the **cite→text** task, where the prompt consists of one or more pages of a deposition transcript followed by the question "What is the exact text of just line __ of page __ above?" (after single pages, the prompt omits the page number). An example of GPT-4o failing this task appears in Figure 1.

BLT's deposition transcript tests are built from a novel corpus we constructed of 33,176 lines of actual deposition transcripts filed with federal district courts and downloaded via Bloomberg Law from the Public Access to Court Electronic Records (PACER) system. They are from a variety of cases; our main criterion was that the transcripts be cleanly OCR'ed and could be fully cleaned with regular expressions. This corpus can be extended *ad infinitum* by downloading further transcripts and cleaning them, as deposition transcripts are not copyrightable.[2] The existing page and line numbers are stripped out, and random spans of appropriate length are selected, with new page numbers and line numbers, in case the LLM saw the original numbering during training.

The size of the prompt is scalable, which was a key goal of BLT. For BLT-4k (the version fitting within a 4,000 token window), we use a mix of 1 and 2 page subsets of depositions. Larger versions have larger subsets. For BLT-128k (fitting inside a 128,000 token window), we use a mix of 120 and 140-page subsets of depositions. Appendix B details the sizes used.

[2]Lipman v. Massachusetts, 311 F. Supp. 593 (D. Mass. 1970); 1 Nimmer on Copyright §5.12[C]. Even in the unlikely event transcripts are copyrightable, this use of them is likely fair use. 17 U.S.C. §107.

Section 5217. Definition of cleight.
(a) General rule. The term "cleight" means any baitisolist or any roussiont.
(b) The term "baitisolist" means any ballinated or any caset.
(c) The term "roussiont" means any dicemercu or any accodpoileare.

What is the exact citation above where the term "roussiont" is defined? (Use standard legal formatting like section 1001(b)(2)).

Section 5217(b)

Figure 2: Example of GPT-4 incorrectly answering **defined→cite** question with a 2-deep, 2-wide synthetic section. The correct answer is "section 5217(c)".

3.2 Synthetic Sections

Lawyers regularly work with hierarchical text, including statutes, contracts, regulations, treaties, court rules, and corporate charters. Hierarchical text is often organized into sections, subsections, paragraphs, subparagraphs, etc. Being able to navigate such hierarchical text is a basic legal text-handling task required of all lawyers, whether they are litigators arguing that a statute applies to their case or are transactional lawyers negotiating contract terms.

We generate synthetic hierarchical sections, following the approach of Blair-Stanek et al. (2023). They use repeated application of the logical form $A \Rightarrow B$, with two parameters: depth, the maximum number of times it is applied; and width, the number of times it is applied to each B. The terms defined are nonces that are not real words but are pronounceable. These synthetic sections can be arbitrarily large, by adjusting the width and depth, thus making tasks based on them scalable to different-sized token windows. For example, BLT-4k has synthetic sections ranging from 2-wide, 2-deep, as in Figure 2, which are very short, up to 3-wide, 4-deep, which takes up much of the 4k token window. At the highest end, BLT-128k has a variety ranging from 60-wide, 2-deep to 5-wide, 5-deep, which (because size is exponential with respect to depth) takes up much of the 128,000-token window. For the full list of sizes in each BLT-*, see Appendix A. Being synthetic ensures they are novel and not seen by LLMs during training. This simulates the challenges faced by lawyers in handling newly drafted contracts, legislation, or other hierarchical text. Nearly unlimited quantities of synthetic sections of any sizes can be generated by permuting the nonces.

A basic legal text-processing skill is finding the citation, in a hierarchical text, of the text to which you are pointing a court or another lawyer. This motivates applying the **text→cite** task on synthetic sections, where the prompt consists of one synthetic section followed by the question "What is the exact citation above of the text "__"? (Use standard legal formatting like section 1001(b)(2))." The code we use to generate synthetic sections guarantees there is only a single correct answer.

The converse legal skill is, given a hierarchical citation, finding the text at it. Hence we apply the **cite→text** task to synthetic sections, with the prompt consisting of one synthetic section followed by the question "What is the exact text of just section __ above?"

We ask this question only of "leaves" in the statute, meaning they have no subsections underneath them. This ensures there is only a single correct answer. For example, suppose that section 573(a) was not a leaf, perhaps with paragraphs 573(a)(1) and 573(a)(2) underneath it. If you asked for the text of section 573(a), it is ambiguous whether you should also return the text of 573(a)(1) and 573(a)(2). Such ambiguity is avoided by considering only leaves.

We also include two other basic legal text-handling tasks on the synthetic sections. Terms are defined in hierarchical texts and often referenced elsewhere in the same hierarchical text. Lawyers must be able to cite a term's precise definition. With **defined→cite**, the prompt is one synthetic section followed by the question "What is the exact citation above where the term "__" is defined? (Use standard legal formatting like section 1001(b)(2))." Conversely, when given such a citation by another lawyer, a lawyer must be able to find the term, which motivates **cite→define**. There, the prompt is one synthetic section followed by the question "What is the term defined at section __?" An example of GPT-4 incorrectly answering a defined→cite problem appears in Figure 2.

3.3 U.S. Code

The U.S. Code is the official compilation of general and permanent U.S. federal statutes. The U.S. Code is a large corpus of hierarchical text. We apply to the U.S. Code all four tasks that we applied to synthetic sections: **text→cite**, **cite→text**,

defined→cite, and **cite→defined**. For these four tasks on the U.S. Code, the prompt is the same as for synthetic sections.

During training, LLMs have doubtless seen all of the U.S. Code, which is not copyrighted and is publicly available on multiple websites. To test whether LLMs' familiarity with U.S. Code sections causes errors, we add a fifth test for U.S. Code sections: **cite→amended**. In all but one respect, this test is identical to cite→text, in that it has the text of one or more sections and asks "What is the exact text of just section __ above?" about a leaf.

The sole difference is that we make a small but semantically-important change to the text in that leaf to see if the LLM returns the original text or the changed text (which is the correct answer). This tests a basic legal skill: applying a given newly-amended statute, rather than its old version. If the leaf contains any numbers, we add or subtract one from the last appearing number (although we never move from 1 to 2 or from 2 to 1 since that would also require changing singular nouns to plural or vice versa). Otherwise, we tweak the last appearing citation from, say, "(D)" to "(A)". Otherwise, we toggle the last "and" to "or" or vice versa. Otherwise, we toggle the last "shall" to "may" or vice versa. If none of these changes are available, we insert "unless otherwise provided by section 101," at the start of the leaf.

For all tasks on the U.S. Code, we do not use sections containing tables, which are not purely text. Examples include the income tax tables at 26 U.S.C. §1, and 5,946 sections are excluded for this reason. We do not use sections, like 5 U.S.C. §9507 and 25 U.S.C. §5329, with quoted hierarchical text such as model contracts, which are hard for even a human lawyer to read. We never use any of the cites that Congress has sloppily added twice, such as the two subsection (e)'s in 42 U.S.C. §1397hh.

For text→cite, we do the same test as with transcripts, not using lines that are under four words long, are subsets of any line appearing elsewhere in the prompt, or that have a Levenshtein distance under four from another line in the prompt. For defined→cite, we do not use terms defined in more than one place in the prompt.

Unlike synthetic sections, which can be generated in unlimited quantities in arbitrarily large sizes, there are a limited number of U.S. Code sections. But it is a huge corpus, with 43,916 sections that meet the criteria discussed above, 447,037 leaves, and 23,562 unique definitions. Although 94% of sections are under 2,000 GPT-4 tokens, that still leaves 2,602 sections over 2,000 tokens, including 813 sections over 4,000 tokens and 196 sections over 8,000 tokens. When there are insufficient numbers of large enough sections, we can generate prompts of any desired size *ad infinitum* by adding randomly selected other sections of approximately the same size. We randomly shuffle the order of the sections in the prompt so that the target section's position is not a cue to the model.

Having multiple sections in a prompt resembles how OpenAI's Greg Brockman pasted nine tax-related sections into GPT-4 during the livestream introducing GPT-4. This is realistic: lawyers handling real-world issues often must apply several statutes in conjunction, not just one.

3.4 General Considerations

For each of the 11 tests, and for each possible size (ranging from BLT-4k to BLT-128k), we generate a training/test split of 1000/100 prompts.[3] Why only 100 test prompts for each test split? Three reasons. First, there are 11 tests, thus 1,100 test prompts for each size of BLT-*. Second, the monetary cost of calling many LLMs with just 1,100 BLT-8k prompts with around 5,000 tokens per prompt is already nontrivial. Third, any LLM deployed for real-world legal practice should be at or near 100%, and as accuracy approaches 100% the t-statistic goes to zero.

Prompt engineering is not our focus, because the BLT tasks are expressly designed to be subsidiary tasks: legal users would not ask LLMs to solve the BLT tasks themselves. Rather, LLMs being able to handle the BLT tasks will generally be a prerequisite to accomplishing higher-level tasks, like drafting court documents citing statutes and deposition transcripts. Moreover, lawyers are unlikely to engage in more than rudimentary prompt engineering. We do try different prompts, as discussed in section 4.1 and appendices E and F.

4 Results and Discussion

We tested four models from OpenAI: GPT-3.5-turbo, GPT-4, and GPT-4-turbo, and GPT-4o. From Google, we tested the chat-bison-32k variant of

[3] All BLT data can be downloaded from:
http://nlp.jhu.edu/law/blt/BLT-4k.zip
http://nlp.jhu.edu/law/blt/BLT-8k.zip
http://nlp.jhu.edu/law/blt/BLT-16k.zip
http://nlp.jhu.edu/law/blt/BLT-32k.zip
http://nlp.jhu.edu/law/blt/BLT-64k.zip
http://nlp.jhu.edu/law/blt/BLT-128k.zip.

		transc.		synthetic section				U.S. Code					
	model	text→cite	cite→text	text→cite	cite→text	defined→cite	cite→defined	text→cite	cite→text	cite→amended	defined→cite	cite→defined	**mean**
BLT-4k	GPT-3.5-turbo	53	32	72	38	83	79	89	52	56 (0)	77	98	66.3
	GPT-4	82	78	88	97	90	100	98	93	93 (0)	98	100	92.5
	GPT-4-turbo	87	88	85	63	76	95	98	84	77 (7)	96	99	86.2
	chat-bison-32k	84	29	7	77	37	92	83	90	89 (1)	81	97	69.6
	Claude-2.1	54	38	74	71	78	85	97	87	87 (1)	96	95	78.4
	GPT-4o	88	84	99	90	98	98	99	91	94 (0)	97	100	94.4
	Claude-3.5	96	80	100	99	100	100	100	97	97 (0)	98	100	97.0
BLT-8k	GPT-4	44	26	64	49	82	83	94	74	76 (0)	88	97	70.6
	GPT-4-turbo	57	53	66	45	75	74	94	80	71 (3)	98	99	73.8
	chat-bison-32k	59	6	9	29	48	59	70	83	86 (1)	80	89	56.2
	Claude-2.1	35	11	58	51	70	54	91	81	79 (4)	94	92	65.1
	GPT-4o	79	37	86	73	88	87	97	85	86 (3)	99	96	83.0
	Claude-3.5	75	66	94	72	99	99	100	100	94 (1)	100	99	90.1
BLT-16k	GPT-4-turbo	30	20	78	36	83	83	90	64	58 (5)	93	95	66.4
	chat-bison-32k	42	6	36	30	69	57	25	54	52 (2)	43	58	42.9
	Claude-2.1	21	5	64	48	76	61	82	66	65 (4)	85	85	59.8
	GPT-4o	59	17	93	86	98	96	96	65	63 (2)	93	96	78.4
	Claude-3.5	69	36	94	68	97	96	99	72	73 (2)	94	95	78.4
BLT-32k	GPT-4-turbo	23	11	42	7	64	63	77	38	36 (1)	82	85	48.0
	chat-bison-32k	13	0	8	10	39	32	12	32	36 (0)	30	36	22.5
	Claude-2.1	20	3	45	24	54	38	76	45	43 (3)	71	70	44.5
	GPT-4o	50	17	82	64	83	76	96	48	44 (6)	95	95	68.2
	Claude-3.5	72	35	62	33	82	72	90	57	56 (6)	91	88	67.1
BLT-64k	GPT-4-turbo	17	4	27	10	60	54	51	19	16 (4)	55	66	34.5
	Claude-2.1	5	1	52	26	56	35	58	31	29 (4)	55	66	37.6
	GPT-4o	65	16	87	61	92	88	95	53	53 (1)	91	93	72.2
BLT-128k	GPT-4-turbo	9	3	3	0	20	17	34	12	12 (0)	51	56	19.7
	Claude-2.1	10	0	11	3	16	21	47	21	18 (1)	45	60	22.9
	GPT-4o	53	10	36	18	72	49	79	30	28 (1)	88	88	50.1

Table 1: Accuracy in percent of several models against all the different sizes of BLT. GPT-4o and Claude-3.5 are broken out separately since their training cutoff was several months after the BLT dataset was made public online, so they may have seen the BLT test data during training. Under cite→amended, the number in parentheses is how often the model erred by returning the unamended U.S. Code text rather than the amended text provided to the model in the prompt, results discussed in Subsection 4.4.

PaLM 2. From Anthropic, we tested Claude-2.1 and Claude-3.5 Sonnet. (Due to token limits, we were not able to test Claude-3.5 on our two largest test sets, BLT-64k and -128k.)

Crucially, GPT-4o and Claude-3.5's training cutoffs were several months after we released the BLT test and training data sets on the internet in November 2023. They may have seen the BLT test sets during training, artificially boosting their performance. Accordingly, they are broken out separately in Table 1.

Table 1 contains our results. All tests were by API call, with temperature set to 0.0 to maximize reproducibility and minimize hallucination. Each number in Table 1 (other than the means and the numbers in parentheses) corresponds to 100 calls to the relevant LLM's API – one call each for each of the 100 prompts in each test set. For example, there are 100 prompts in test for text→cite for synthetic sections in BLT-4k. Our code measures accuracy with forgiving rules, ignoring case and whitespace. Our code uses handwritten rules to classify errors, a feature we draw on in the discussion below.

We note that accuracy monotonically decreases as prompt size increases, with each model achieving higher accuracy on BLT-4k than BLT-8k, higher accuracy on BLT-8k than BLT-16k, and so on.

4.1 GPT-4 on transcript text→cite

All GPT-4 models (GPT-4, GPT-4-turbo, and GPT-4o) performed under 90 percent on transcript text→cite for BLT-4k. To further investigate this poor performance, we generate 1,000 new prompts in the same format (25-lines per page, with half being one-page and half being two-page) and pass them to GPT-4. GPT-4 achieves 87.5% on these 1,000. Qualitatively, most errors are either identifying the line after the correct one or before the correct one. The full error breakdowns are in Appendix D.

The biggest determinant of performance is whether the transcript was a single page or two pages. GPT-4 correctly answered 91% of single-page transcript prompts, but just 84% of 2-page transcript prompts. This makes sense, since 2-page transcripts have 50 lines of text, whereas 1-page transcripts have just 25 lines of text. (An example of GPT-4 getting a wrong answer on a 2-page transcript appears in Appendix C.)

To see whether the problem is the greater number of lines or having the text split into two pages, we generate 500 new prompts with single pages but with 50 lines per page. (In other words, the transcript quotation is all one page, but with line numbers starting at "1:" and ending at "50:", followed by the question). We find GPT-4 achieves 84.8% accuracy, nearly identical to the two-page transcripts, indicating the problem is length, not being split into two pages.

To investigate how the location, within the transcripts, of the text impacts accuracy, we generate 5,000 new two-page prompts, and run against GPT-4. The results are the red dashed line in Figure 3. We see a generally downwards trend. We do not see the distinct U-pattern observed by Liu et al. (2023a) in accuracy versus position of requested information within a long prompt.

We also perform a sensitivity analysis, trying four question formats other than “What are the page number and line number of the line above with the text "__"?” appearing after the quotations. (Details in Appendix E.) We find the biggest improvement simply by swapping the question from the end to the beginning and changing “above” to “below”. We tried the same switch – moving the question from the bottom to the top – for all of BLT-4k and re-ran against GPT-4. The results are in Appendix F. It turns out that moving the question from the end to the start actually hurt performance in 7 of 11 tasks, indicating no general trend.

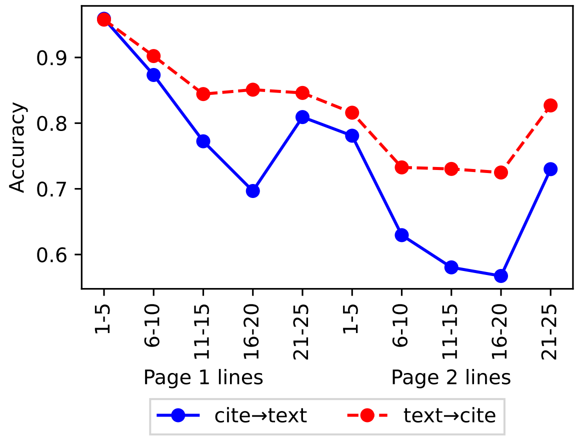

Figure 3: Graph of location of relevant line versus accuracy on both transcript **cite→text** and **text→cite** on 5,000 prompts to GPT-4.

4.2 GPT-4 on transcript cite→text

All models performed poorly on transcript cite→text for BLT-4k. For example, Claude-3.5 got 80%, GPT-4 got 78%, and GPT-4o got 84%. To further investigate, we generated 1,000 new prompts in the same format and passed them to GPT-4, which got got 75.7% accuracy on these 1,000. We found little difference between one-page and two-page transcripts, on which GPT-4 got 76.6% and 74.8% respectively.

To investigate how accuracy varies with the location of the requested cite, within the transcripts, we generate 5,000 new two-page prompts, and pass them to GPT-4. The results are the solid blue line in Figure 3. We see a trend towards lower accuracy further into the transcript, with higher accuracy near the beginning and end of each page.

4.3 Poor Performance on synthetic cite→text

Several models have their worst performance, among synthetic section tasks, on cite→text. Each BLT-* has a variety of different size sections, shown in Appendix A. Some of the worst BLT-4k performance is on 3-wide, 4-deep synthetic sec-

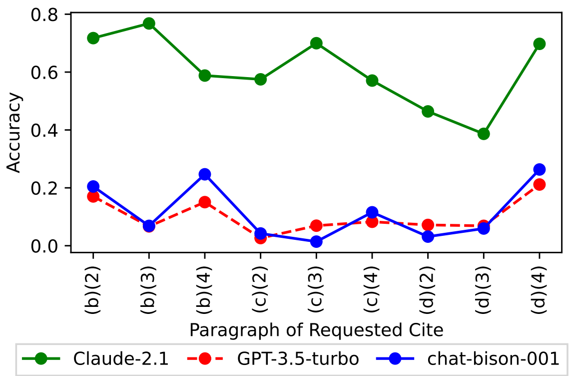

Figure 4: Graph of location of requested cite versus accuracy for 5,000 synthetic **cite→text** prompts, all using 3-wide, 4-deep synthetic sections, which are 127 lines long. Note that each first subdivision (e.g., (a), (1)) is used for a "General Rule" that has few lines, so such subdivisions are not included in this graph.

text and task	not tuned	fine-tuned
transcript text→cite	53	100
transcript cite→text	32	99
synthetic text→cite	72	98
synthetic cite→text	38	100
synthetic defined→cite	83	100
synthetic cite→defined	79	100
uscode text→cite	89	100
uscode cite→text	52	100
uscode cite→amendedtext	56	100
uscode defined→cite	77	100
uscode cite→defined	98	100

Table 2: Results of fine-tuning GPT-3.5-turbo on 9,900 training samples from BLT-4k. Both numerical columns contain percent accuracy on BLT-4k's test prompts. Fine-tuning GPT-3.5-turbo improves it to near perfect.

tions. To see if location within the section plays a role in accuracy, we generated 5,000 prompts using 3-wide, 4-deep synthetic sections, and we ran all against three models with poor performance on synthetic cite→text. The results are in Figure 4. Once again, in contrast to Liu et al. (2023a), there is no U-pattern. We found the same lack of discernible pattern in accuracy versus location with all models and all tasks involving either synthetic sections or the U.S. Code.

4.4 Problem Revealed by cite→amended

Recall that for cite→amended we make a minimal, but semantically-important, change to the subsection of the U.S. Code section being requested. The correct answer is returning the subsection's text with this amendment. Many of the errors involve returning the text of the wrong subsection. But one type of error is particularly concerning: an LLM returning the subsection *without* the amendment, presumably relying on the original U.S. Code text seen during training.

In practice, this error means that, even if a lawyer or paralegal pastes in the new version of legislation, the LLM ignores it. The LLM will act as if the legislation had never been amended.

The occurrence of this error is in parentheses under the cite→amended column in Table 1, showing that several LLMs make this error a nontrivial number of times. To investigate further, we generated 1,000 new BLT-4k style cite→amended prompts. On these, Claude-2.1 incorrectly returned the original text 17 times (1.7%), chat-bison-32k did so 4 times (0.4%), GPT-4o did so 10 times (1%), and GPT-4-turbo did so fully 44 times (4.4%).

5 Fine-Tuning

We fine-tune the 4,000-token version of GPT-3.5-turbo with BLT-4k's training set. For each of the 11 task types, BLT-4k has a training set with 1,000 prompts and answers, for a total of 11,000 prompts and answers. Recall that the training set and test set are generated in the same way, with the same code. Of the 11,000 prompts and answers in the training set, we use 90% for training, leaving 10% as a possible evaluation set for future work. We train for two epochs with all hyperparameters, like learning rate, set to the defaults. The results of the fine-tuning are in Table 2. We find that fine-tuning brings GPT-3.5-turbo, which is far from OpenAI's most advanced model, to near the 100% performance expected of lawyers and paralegals.

We tested how this fine-tuned GPT-3.5-turbo performs on the SARA dataset (Holzenberger et al., 2020), using the 276 cases where the answer is entail/contradict. Each prompt consists of each U.S. Code section(s) mentioned in the case, plus the facts (i.e., the premise) and the hypothesis (i.e., the question). Without fine-tuning, GPT-3.5-turbo's accuracy was 54.3% (150 / 276), but with our fine-tuning it rises to 60.9% (168 / 276). Qualitatively, the fine-tuned model's answers focus on applying only the relevant statutory provisions. (An example is in Appendix H.)

This result shows the utility of fine-tuning LLMs on BLT's training data to improve performance on higher-level legal tasks. Also, this result is in line with the findings of Dominguez-Olmedo et al. (2024) that small LLMs that have been lightly fine-tuned on legal tasks often substantially outperform larger LLMs on legal tasks.

6 LLMs Trained After BLT Release

We generally see the best performance from Claude-3.5 and GPT-4o, the two most recently released LLMs that we test. There are three possible, non-exclusive explanations. First, these models may be more advanced and thus better at most tasks. Second, we released the BLT test sets on the internet in November 2023, several months before these models' training cutoff dates, so there may have been test-set leakage, with the models already having seen the answers during training. Third, the BLT training sets (which are ten times larger than the test sets) were also released in November 2023, before these models' training cutoffs. As the fine-tuning experiments in Section 5 demonstrate, training on the BLT training data can substantially improve performance. Seeing the BLT training data may have improved these models' basic legal text handling.

7 Conclusion

The chief innovation officer at a large international law firm observed to the *New York Times* of LLMs, "At its best, the technology seems like a very smart paralegal." (Lohr, 2023). We find LLMs are more like sloppy paralegals.

Currently available LLMs perform poorly out-of-the-box on basic legal text handling. The BLT tasks are designed to be truly basic, with humans able to perform them at or near 100%. The GPT-4 family, PaLM 2, and Claude-2.1 all fall far short. Only one model, Claude-3.5, comes close, with 97.0% aggregate performance, and then only on our smallest test set, BLT-4k. Yet even Claude-3.5 achieves only 80% on retrieving the text on a line of one or two pages of deposition transcript.

We find that fine-tuning on our training set brings performance up to near 100%. We expect BLT to be a useful resource for those fine-tuning LLMs for much more complicated legal tasks, as well as a benchmark for LLMs' ability to do basic legal text handling without fine-tuning.

Ethics Statement

LLMs can be misused by legal professionals and laypersons alike to address legal problems properly requiring the full attention of a legal professional. One of our goals is to alert potential users of the failings of existing LLMs at basic legal tasks. Users may misconstrue our findings on the value of fine-tuning to assume incorrectly that an LLM with such fine-tuning can handle legal matters. Even if an LLM gets 100% on BLT, that does not mean the LLM can handle legal matters.

Acknowledgments

This work has been supported by the U.S. National Science Foundation under grant 2204926. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the National Science Foundation. We would like to thank Susan McCarty and Orion Weller for valuable assistance.

References

Abdi Aidid and Benjamin Alarie. 2023. *The Legal Singularity: How Artificial Intelligence Can Make Law Radically Better*. University of Toronto Press.

Brendin R Beaulieu-Jones, Margaret T Berrigan, Sahaj Shah, Jayson S Marwaha, Shuo-Lun Lai, and Gabriel A Brat. 2024. Evaluating capabilities of large language models: Performance of gpt-4 on surgical knowledge assessments. *Surgery*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Andrew Blair-Stanek, Nils Holzenberger, and Benjamin Van Durme. 2023. Can GPT-3 Perform Statutory Reasoning? In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, ICAIL '23, page 22–31.

Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019. Neural legal judgment prediction in English. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4317–4323, Florence, Italy. Association for Computational Linguistics.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. LexGLUE: A benchmark dataset for legal language understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers), pages 4310–4330, Dublin, Ireland. Association for Computational Linguistics.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, Andre F. T. Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, and Michael Desa. 2024. Saullm-7b: A pioneering large language model for law. *arXiv*.

Ricardo Dominguez-Olmedo, Vedant Nanda, Rediet Abebe, Stefan Bechtold, Christoph Engel, Jens Frankenreiter, Krishna Gummadi, Moritz Hardt, and Michael Livermore. 2024. Lawma: The power of specialization for legal tasks. *arXiv*.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Songyang Zhang, Kai Chen, Zongwen Shen, and Jidong Ge. 2023. Lawbench: Benchmarking legal knowledge of large language models. *Preprint*, arXiv:2309.16289.

Neel Guha, Daniel E. Ho, and Julian Nyarko et al. 2022. Legalbench: Prototyping a collaborative benchmark for legal reasoning. *Preprint*, arXiv:2209.06120.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated NLP dataset for legal contract review. *Advances in Neural Information Processing Systems*.

Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2020. A dataset for statutory reasoning in tax law entailment and question answering. In *Proceedings of the Natural Legal Language Processing Workshop 2020 co-located with the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020), Virtual Workshop, August 24, 2020*, volume 2645 of *CEUR Workshop Proceedings*, pages 31–38. CEUR-WS.org.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Mi-Young Kim, Juliano Rabelo, Randy Goebel, Masaharu Yoshioka, Yoshinobu Kano, and Ken Satoh. 2022. COLIEE 2022 summary: Methods for legal document retrieval and entailment. In *New Frontiers in Artificial Intelligence - JSAI-isAI 2022 Workshop, JURISIN 2022, and JSAI 2022 International Session, Kyoto, Japan, June 12-17, 2022, Revised Selected Papers*, volume 13859 of *Lecture Notes in Computer Science*, pages 51–67. Springer.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Van-Hoang Le and Hongyu Zhang. 2023. An evaluation of log parsing with chatgpt. *arXiv preprint arXiv:2306.01590*.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

David D. Lewis, Lenora Gray, and Mark Noel. 2023. Confidence sequences for evaluating one-phase technology-assisted review. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, ICAIL '23, page 131–140.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Steve Lohr. 2023. A.I. threatens lawyers? We've heard this before. *New York Times*, page B1.

Masha Medvedeva and Pauline McBride. 2023. Legal judgment prediction: If you are going to do it, do it right. In *Proceedings of the Natural Legal Language Processing Workshop 2023*, pages 73–84, Singapore. Association for Computational Linguistics.

John J. Nay, David Karamardian, Sarah B. Lawsky, Wenting Tao, Meghana Bhat, Raghav Jain, Aaron Travis Lee, Jonathan H. Choi, and Jungo Kasai. 2023. Large language models as tax attorneys: A case study in legal capabilities emergence. *Preprint*, arXiv:2306.07075.

OpenAI. 2023a. Gpt-4 developer livestream.

OpenAI. 2023b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Sumit Pai, Sounak Lahiri, Ujjwal Kumar, Krishanu Baksi, Elijah Soba, Michael Suesserman, Nirmala Pudota, Jon Foster, Edward Bowen, and Sanmitra Bhattacharya. 2023. Exploration of open large language models for eDiscovery. In *Proceedings of the Natural Legal Language Processing Workshop 2023*, pages 166–177, Singapore. Association for Computational Linguistics.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.

Drescher ProParalegal. 2017. Practice tip sheet.

Adam Roegiest, Radha Chitta, Jonathan Donnelly, Maya Lash, Alexandra Vtyurina, and Francois Longtin. 2023. Questions about contracts: Prompt templates for structured answer generation. In *Proceedings of the Natural Legal Language Processing Workshop 2023*, pages 62–72, Singapore. Association for Computational Linguistics.

Jaromír Savelka, Kevin D. Ashley, Morgan A. Gray, Hannes Westermann, and Huihui Xu. 2023. Can GPT-4 support analysis of textual data in tasks requiring highly specialized domain expertise? In *Proceedings of the 6th Workshop on Automated Semantic Analysis of Information in Legal Text co-located with the 19th International Conference on Artificial Intelligence and Law (ICAIL 2023), Braga, Portugal, 23rd September, 2023*, volume 3441 of *CEUR Workshop Proceedings*, pages 1–12. CEUR-WS.org.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Sakib Shahriar, Brady D. Lund, Nishith Reddy Mannuru, Muhammad Arbab Arshad, Kadhim Hayawi, Ravi Varma Kumar Bevara, Aashrith Mannuru, and Laiba Batool. 2024. Putting gpt-4o to the sword: A comprehensive evaluation of language, vision, speech, and multimodal proficiency. *Applied Sciences*, 14(17):7782.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, and Adam Fisch et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *Preprint*, arXiv:2302.11382.

Xuanxin Wu and Yuki Arase. 2024. An in-depth evaluation of gpt-4 in sentence simplification with error-based human assessment. *Preprint*, arXiv:2403.04963.

Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, and Jianfeng Xu. 2018. CAIL2018: A large-scale legal dataset for judgment prediction. *CoRR*, abs/1807.02478.

Fangyi Yu, Lee Quartey, and Frank Schilder. 2022. Legal prompting: Teaching a language model to think like a lawyer. *CoRR*, abs/2212.01326.

Joanna Zhao and Xinruo Wang. 2024. Unleashing efficiency and insights: Exploring the potential applications and challenges of chatgpt in accounting. *Journal of Corporate Accounting & Finance*, 35(1):269–276.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. How does nlp benefit legal system: A summary of legal artificial intelligence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5218–5230.

A Synthetic Section Sizes

Larger versions of BLT have longer and more complicated prompts. Below are the size of synthetic sections in each size of BLT. The generated prompts are distributed uniformly among these section sizes. For example, one-quarter of BLT-16k's synthetic statutes are 5-wide, 4-deep; one-quarter are 8-wide, 3-deep; and so on.

Version	Sizes
BLT-4k	2 wide, 2 deep
	2 wide, 3 deep
	2 wide, 4 deep
	2 wide, 5 deep
	3 wide, 2 deep
	3 wide, 3 deep
	3 wide, 4 deep
	4 wide, 2 deep
	4 wide, 3 deep
BLT-8k	2 wide, 6 deep
	3 wide, 5 deep
	4 wide, 4 deep
	7 wide, 3 deep
	20 wide, 2 deep
BLT-16k	5 wide, 4 deep
	8 wide, 3 deep
	9 wide, 3 deep
	30 wide, 2 deep
BLT-32k	3 wide, 6 deep
	4 wide, 5 deep
	6 wide, 4 deep
	12 wide, 3 deep
	11 wide, 3 deep
	44 wide, 2 deep
	40 wide, 2 deep
BLT-64k	7 wide, 4 deep
	16 wide, 3 deep
	15 wide, 3 deep
	14 wide, 3 deep
	13 wide, 3 deep
	60 wide, 2 deep
	65 wide, 2 deep
BLT-128k	4 wide, 6 deep
	5 wide, 5 deep
	8 wide, 4 deep
	9 wide, 4 deep
	20 wide, 3 deep
	80 wide, 2 deep

B Transcript Quotation Sizes

Larger versions of BLT have longer prompts. Below are the number of pages of deposition transcript quotation used in each size BLT model.

Version	Transcript Pages
BLT-4k	1, 2
BLT-8k	5, 10, 15
BLT-16k	25, 40
BLT-32k	30, 60, 80
BLT-64k	100, 130
BLT-128k	120, 140

The generated prompts are distributed uniformly among these page sizes. For example, half of BLT-4k's prompts have one-page deposition quotations and the other half have two-page deposition quotations.

C Example GPT-4 failing on a 2-page transcript text→cite Prompt

Page 68:
1: after the accident?
2: A. He called me – that same day,
3: he called me at nighttime and asked me how
4: I was.
5: Q. He called you and asked you how
6: you were?
7: A. Yes.
8: Q. What did you tell him?
9: A. I just told him that I wasn't
10: feeling good, that I was just laying in
11: bed.
12: Q. And what did he say?
13: A. He said take the pills I gave
14: you, you'll be ready and better by next
15: week. He was just giving me something like
16: a fast way for me to feel up, to cheer up.
17: It wasn't helping.
18: Q. Going back to your primary care
19: physician. You said she wanted to send you
20: to a specialist?
21: A. Yes.
22: Q. But then you called workers'
23: comp?
24: A. Yeah, that's when the issue
25: happened and I called workers' comp.

Page 69:
1: Q. Then who did you treat with
2: next?

3: A. I treated with CityMD.
4: Q. You said CityMD?
5: A. Yes.
6: Q. Was the place that you went to
7: the first time the same place that you went
8: like after you went to your primary care
9: physician?
10: A. Excuse me again?
11: MR. GROSSMAN: You said you
12: went to CityMD. She thinks you went
13: twice, did you go twice?
14: THE WITNESS: No.
15: MR. GROSSMAN: Then you're not
16: listening to the question. You went
17: to CityMD first, then you went to
18: your primary physician. Now, she
19: wants to know where you went next.
20: THE WITNESS: No, first I went
21: to the primary.
22: MR. GROSSMAN: First primary,
23: and then CityMD and then where did
24: you go?
25: THE WITNESS: I was in CityMD

What are the page number and line number of the line above with the text "your primary physician. Now, she"?

The page number is 69 and the line number is 19.

Note that GPT-4's answer above is wrong, since the requested text is on line 18, not 19.

D GPT-4 Errors on Transcripts

GPT-4 performs surprisingly poorly on transcripts, even the 1- and 2-page transcripts in BLT-4k. Below are GPT-4's errors on the large runs used to investigate the problem. Our API-calling code automatically categorizes the errors.

D.1 GPT-4's errors on transcript text→cite from BLT-4k

Recall that we generated 1,000 new BLT-4k style transcripts with text→cite prompts to investigate GPT-4's poor performance. Below is a tabulation of the lines returned by GPT-4 relative to the correct ones. We see that the overwhelming majority of errors are returning the line before or line after the correct one.

#	**Relative Error**
3	prior page
1	19 before
1	3 before
1	2 before
12	1 before (i.e. prior line)
76	1 after (i.e. next line)
1	2 after
1	3 after
1	next page

D.2 GPT-4's errors on transcript cite→text from BLT-4k

Recall that we generated 1,000 new BLT-4k style transcripts with cite→text prompts to investigate GPT-4's poor performance on that task. Below is a tabulation of the errors.

#	**Type of Error**
166	wrong line
48	superset
10	not found
9	subset wrong line
7	multiple wrong lines
3	subset

There was a difference in the type of errors on one-page versus two-page. Almost all (84%) the errors on two-page transcripts was returning the text from the wrong line (almost always the line before or after the correct one). But one-page transcripts were only 51% wrong-line errors, with 35% being superset errors, where GPT-4 returned not only the text of the requested line, but also part of adjacent lines. Here is an example of a superset error that occurred on this dataset:

...
22: MS. WOLVERTON: Objection. Vague. Whose
23: e-mail records?
24: Q Ms. Abedin's e-mail records located. I
25: apologize.

What is the exact text of just line 25 above?

GPT-4 returned "Q Ms. Abedin's e-mail records located. I apologize." which is actually lines 24 and 25, whereas the prompt asks for "the exact text of **just** line 25 above". These superset errors are much more understandable from a human perspective than an entirely wrong line, but they still fail according to the clear instructions in the prompt, with the word 'just'.

Of the 1,000 prompts, 166 were "wrong line" errors. Below shows the position, relative to the correct one, from which the text was incorrectly drawn:

#	**Relative Error**
2	prior page
4	same page, >3 lines before
1	3 lines before
6	2 lines before
81	prior line
39	next line
10	2 lines after
7	3 lines after
6	4 lines after
15	same page, >4 lines after
3	next page

As with text→cite, here we see the overwhelming majority of wrong line errors are returning either the prior line or next line.

E Sensitivity analysis on GPT-4's transcript text→cite errors

Recall that GPT-4 performs poorly on text→cite, so we attempted a sensitivity analysis, seeing how performance on the 100 test prompts for the task in BLT-4k changed with changes in the phrasing within the prompt. The results are below.

BLT-4k default **What are the page number and line number of the line above with the text "__"?**	82/100
Move question from end to beginning, so question is **What are the page number and line number of the line below with the text "__"?**	99/100
Keep question at end, adding "exact" so question is **What are the exact page number and the exact line number of the line above with the text "__"?**	84/100
Keep question at end, adding "precise" so question is **What are the precise page number and the precise line number of the line above with the text "__"?**	84/100
Keep default question at end, but add the following introduction at the start: **Below is a portion of a transcript, with each line starting with a number that is important for referring to that line.**	93/100

The improvement from moving the question to the start motivated further experimentation, below.

F Question at Start versus End

Because transcript text→cite on GPT-4 saw large improvements from moving the question from the bottom (which is the standard for all of BLT-* for all tasks) to the top, we attempted the same change for all BLT-4k, running against GPT-4:

text and task	question at...	
	end	start
transcript text→cite	82	99
transcript cite→text	78	85
synthetic text→cite	88	91
synthetic cite→text	97	82
synthetic defined→cite	90	82
synthetic cite→defined	100	98
uscode text→cite	98	96
uscode cite→text	93	82
uscode cite→amendedtext	93	82
uscode defined→cite	98	71
uscode cite→defined	100	100

We see that GPT-4 is quite sensitive to whether the question is at the top or bottom. But moving the question to the start actually produced worse results for 7 of the 11 tasks. This indicates that the improvement in transcript text→cite by moving the question to the top was an outlier.

G Sensitivity analysis on GPT-4's transcript cite→text errors

Recall that GPT-4 did poorly on transcript text→cite and that we performed a sensitivity analysis, discussed in Appendix E. Since GPT-4 also did pooorly on cite→text, we also do a sensitivity analysis on that.

BLT-4k default **What is the exact text of just line _ of page _ above?**	78/100
Move question from end to beginning, so question is **What is the exact text of just line _ of page _ below?**	85/100
Rephase as **Return the exact text of just line 4 of page 59 above?**	74/100
Change "exact" to "precise", so question is **What is the precise text of just line _ of page _ above?**	79/100
Add clarification, so question is **What is the exact text of just line _ of page _ above? Return just the text on that line and return none of the text on the line before or after, even if necessary for a full sentence.**	62/100
Keep default question at end, but add the following introduction at the start: **Below is a portion of a transcript, with each line starting with a number that is important for referring to that line.**	90/100

H Example of fine-tuned GPT-3.5-turbo performing better

Recall that fine-tuning GPT-3.5-turbo on part of BLT-4k's training set improved its performance on the SARA dataset (Holzenberger et al., 2020). Below we have an example SARA prompt used, plus both non-tuned GPT-3.5-turbo's answer and the fine-tuned version's answer. We see the fine-tuned version sticks much more closely to the relevant statutory text and thus comes to the correct answer.

The prompt (edited with ellipses for brevity):

We are going to be doing Entailment/Contradiction reasoning applying the statute(s) below:
§2. Definitions and special rules
(a) Definition of surviving spouse
(1) In general
For purposes of section 1, the term "surviving spouse" means a taxpayer-
(A) whose spouse died during either of the two years immediately preceding the taxable year, and
(B) who maintains as his home a household which constitutes for the taxable year the principal place of abode (as a member of such household) of a dependent (i) who (within the meaning of section 152) is a son, stepson, daughter, or stepdaughter of the taxpayer, and (ii) with respect to whom the taxpayer is entitled to a deduction for the taxable year under section 151.
***
(b) Definition of head of household
(1) In general ***
(2) Determination of status
Notwithstanding paragraph (1),
(A) an individual who is legally separated from his spouse under a decree of divorce or of separate maintenance shall not be considered as married;
(B) a taxpayer shall be considered as not married at the close of his taxable year if at any time during the taxable year his spouse is a nonresident alien; and
(C) a taxpayer shall be considered as married at the close of his taxable year if his spouse (other than a spouse described in subparagraph (B)) died during the taxable year.
***
Premise: Alice and Bob got married on Feb 3rd, 1992. Alice died on July 9th, 2014.
Hypothesis: Section 2(b)(2)(C) applies to Bob in 2014.
Answer:

Non-tuned GPT-3.5-turbo's incorrect answer:

To determine whether Section 2(b)(2)(C) applies to Bob in 2014, we need to analyze the relevant provisions of the tax code.

Section 2(b)(2)(C) states that a taxpayer shall be considered as married at the close of his taxable year if his spouse (other than a spouse described in subparagraph (B)) died during the taxable year.

In this case, Alice, Bob's spouse, died on July 9th, 2014. To determine if Section 2(b)(2)(C) applies to Bob in 2014, we need to consider whether Alice's death during the taxable year qualifies Bob as a surviving spouse.

According to Section 2(a)(1)(A), a surviving spouse is defined as a taxpayer whose spouse died during either of the two years immediately preceding the taxable year. In this case, Alice died in 2014, which is within the two years immediately preceding the taxable year.

Next, we need to consider Section 2(a)(1)(B), which states that a surviving spouse must maintain as his home a household that constitutes the principal place of abode of a dependent. The dependent must be a son, stepson, daughter, or stepdaughter of the taxpayer, and the taxpayer must be entitled to a deduction for the taxable year under Section 151.

In this case, we don't have information about whether Bob has a dependent who meets the criteria mentioned above. Therefore, we cannot determine if Bob meets the requirements of a surviving spouse under Section 2(a)(1)(B).

Since we cannot determine if Bob meets the requirements of a surviving spouse, we cannot conclude whether Section 2(b)(2)(C) applies to Bob in 2014.

Fine-tuned GPT-3.5-turbo's correct answer:

To determine if Section 2(b)(2)(C) applies to Bob in 2014, we need to consider the marital status of Bob and whether his spouse died during the taxable year.

According to the premise, Alice died on July 9th, 2014, and she was married to Bob. Therefore, we need to determine Bob's marital status at the close of the taxable year 2014.

Since Bob and Alice got married on Feb 3rd, 1992, and Alice died in 2014, Bob was married to Alice at the close of 2014.

Therefore, Section 2(b)(2)(C) applies to Bob in 2014 because his spouse (Alice) died during the taxable year.

I Dataset Statistics

We computed statistics on the BLT dataset using NLTK's tokenizers and Stanford's tagger (Bird

	transc.		synthetic section				U.S. Code				
	text→cite	cite→text	text→cite	cite→text	defined→cite	cite→defined	text→cite	cite→text	cite→amended	defined→cite	cite→defined
mean	11.3	11.3	51.3	51.5	30.5	30.6	48.7	48.6	48.6	48.6	48.5
stddev	11.4	11.4	58.5	58.6	29.3	29.3	93.9	95.0	95.0	93.4	93.8
median	7	7	37	37	22	22	29	29	29	29	29
min	1	1	3	3	3	3	2	1	1	1	1
max	286	286	490	490	250	250	6383	6383	6383	6383	6383

Table 3: Number of words per sentence in the training set. Statistics computed across all sizes of the BLT dataset. Sentence and word boundaries were determined using NLTK's standard tokenizer (Bird et al., 2009).

Adjective	3	3	1	1	2	2	7	7	7	7	7
Adverb	4	4	0	0	1	1	1	1	2	1	2
Conjunction	11	11	2	2	3	4	19	19	19	19	19
Determiner	7	8	15	15	25	23	11	11	11	11	11
Noun	19	21	28	29	29	29	29	29	29	29	30
Number	11	11	3	2	0	0	3	3	3	2	3
Pronoun	8	6	0	0	0	0	0	0	0	0	0
Punctuation	24	23	51	50	36	37	19	19	19	19	19
Verb	13	13	0	0	3	4	10	10	10	10	10

Table 4: Part-of-speech tags in % of occurrence, rounded to the closest percentage point. Statistics computed across all sizes of the BLT dataset, on a 1% subset drawn at random from the training set for each task. Sentence and word boundaries were determined using NLTK's standard tokenizer (Bird et al., 2009). Part-of-speech tags were inferred using Stanford's POS tagger (Toutanova et al., 2003).

et al., 2009; Toutanova et al., 2003). Number of sentences per document (Table 5) and number of words per sentence (Table 3) were computed on the entire training set. The distribution of part-of-speech tags (Table 4) was computed using 1% of the training set, chosen at random. Given that these automatic tools were trained on standard written English, they may give somewhat inaccurate results on legal English.

BLT		transc.		synthetic section				U.S. Code				
		text→cite	cite→text	text→cite	cite→text	defined→cite	cite→defined	text→cite	cite→text	cite→amended	defined→cite	cite→defined
4k	mean	34.5	33.7	25.0	24.0	25.0	24.0	58.9	58.7	58.7	63.8	63.1
	stddev	15.3	15.1	16.9	16.9	16.9	16.9	19.4	20.6	20.6	20.9	21.3
	median	32	31	21	20	21	20	58	58	58	63	62
	min	8	8	8	7	8	7	5	4	4	9	8
	max	81	79	57	56	57	56	119	219	219	133	132
8k	mean	215.4	214.6	93.6	92.6	93.6	92.6	121.4	119.7	119.7	127.1	127.4
	stddev	105.9	105.8	46.0	46.0	46.0	46.0	34.9	32.9	32.9	36.6	35.8
	median	205.5	205	98	97	98	97	124	122	122	127	129
	min	58	57	26	25	26	25	34	29	29	23	32
	max	472	471	165	164	165	164	213	210	210	240	238
16k	mean	722.4	721.6	104.5	103.5	104.5	103.5	231.7	231.7	231.7	233.5	231.5
	stddev	262.7	262.5	56.0	56.0	56.0	56.0	62.3	62.0	62.0	65.0	65.0
	median	644.5	643	95.5	94.5	95.5	94.5	231	231	231	240.5	235
	min	315	315	36	35	36	35	68	61	61	90	89
	max	1187	1186	191	190	191	190	435	401	401	389	390
32k	mean	1263	1263	234.7	233.7	234.7	233.7	461.8	460.4	460.4	462.6	458.0
	stddev	566.3	566.2	164.2	164.2	164.2	164.2	104.8	102.8	102.8	104.9	107.5
	median	1183	1181	174	173	174	173	469	473	473	478	470
	min	391	390	46	45	46	45	173	183	183	171	185
	max	2307	2305	489	488	489	488	768	767	767	740	752
64k	mean	2531	2530	226.2	225.2	226.2	225.2	873.1	874.4	874.4	884.9	884.1
	stddev	799.3	799.0	126.2	126.2	126.2	126.2	199.4	199.2	199.2	177.3	176.2
	median	2721	2720	230	229	230	229	875	874	874	883	882
	min	1365	1365	66	65	66	65	425	402	402	425	424
	max	3648	3646	461	460	461	460	1377	1375	1375	1318	1378
128k	mean	2874	2873	793.3	792.3	793.3	792.3	1817	1820	1820	1805	1818
	stddev	872.6	872.4	503.2	503.2	503.2	503.2	320.7	304.3	304.3	301.3	307.2
	median	3306	3305	788.5	787.5	788.5	787.5	1804	1825	1825	1794	1816
	min	1671	1671	86	85	86	85	904	903	903	983	923
	max	3929	3926	1710	1709	1710	1709	2752	2657	2657	2776	2751

Table 5: Number of sentences per document in the training set. At most 4 significant digits were kept. Sentence boundaries were determined using NLTK's standard tokenizer (Bird et al., 2009).