Generative LLMs and Textual Analysis in Accounting:
(Chat)GPT as Research Assistant?

Ties de Kok
tdekok@uw.edu
University of Washington

April 2023

*Early work-in-progress.*
*Comments and feedback are welcome!*

**Abstract**

Generative Large Language Models (GLLMs), such as the ChatGPT and GPT-4 models by OpenAI, are emerging as powerful tools for textual analysis tasks in accounting research. GLLMs can solve any textual analysis task solvable using non-generative methods, as well as tasks previously only solvable using human coding. This paper highlights the applications of GLLMs for accounting research and compares them to existing textual analysis methods. I also provide a framework for researchers to effectively utilize GLLMs in their work, addressing key considerations such as model selection, prompt engineering, and construct validity. Furthermore, I highlight the importance of addressing bias, replicability, data privacy, and attributability concerns when employing GLLMs. Finally, I explore current GLLM developments and provide practical guidance and code examples in the appendix. Taken together, this paper equips researchers with the necessary knowledge and tools to harness the potential of GLLMs and editors and reviewers with the knowledge to better evaluate papers that use the GLLM approach.

**LLM assistance statement:** GPT-4 and ChatGPT were used to help with copy-editing and coding assistance in this project.

**Acknowledgements:** I thank Beth Blakespoor, Ed deHaan, Andy Leone, and Xue Li for helpful comments.
1. Introduction

Rapid advancements in natural language processing and machine learning have given rise to Generative Large Language Models (GLLMs), such as the ChatGPT and GPT-4 models by OpenAI. This new family of models holds significant potential for accounting research studying textual data. GLLMs can solve textual analysis tasks with the power, versatility, and ease-of-use of human research assistants (RAs) while enjoying the scalability, speed, and cost benefits of machine learning methods. For example, a well-instructed GLLM can extract emission targets in ESG reports, find CEOs trying to be funny in conference calls, compare risk factor consistency across firms, cluster topics in employee reviews, extract corporate culture references in job postings, and much more. Solving these problems using a GLLM can yield better results at a fraction of the researchers’ time relative to using non-generative machine learning (ML) approaches or human coding. This paper shows the application of GLLMs for accounting research and provides a framework and discussion that researchers, reviewers, and editors can use when working with or evaluating GLLMs.

GLLMs significantly improve on existing textual analysis approaches in several ways. First, they operate through natural language, which makes it easier to express a wide range of textual analysis tasks. Second, they can complete many tasks with no or little training data, which saves significant time and resources. Tasks that would previously take months might take only a couple of weeks to solve using a GLLM. For example, Hassan et al. (2019) identify discussions about political risk in earnings call by training an algorithm on a large training dataset of political texts. Achieving this is a challenging and time-consuming task without GLLMs. In contrast, a state-of-the-art GLLM like GPT-4 can solve this problem out-of-the-box.1 The only required action is to

1 See Appendix C for examples of GPT-4 completing these tasks.
show GPT-4 an earnings call excerpt and ask whether it discusses political risks or not. Using a less powerful GLLM, such as GPT-3, requires more work but can still be significantly easier and faster relative to existing textual analysis techniques. Furthermore, GLLMs exhibit basic reasoning abilities and world understanding and can solve problems that would otherwise be infeasible to automate. For example, Call et al. (2022) use a sophisticated ML algorithm to detect laughter in conference call audio to identify CEO humor. This approach is complex and limited to successful humor, as unsuccessful humor doesn’t elicit laughter. In contrast, a GPT-4 model without training can detect attempts at humor by just showing it the transcript text and asking whether the CEO tried to be funny. GLLMs show the potential to lower the barrier of entry to analyzing textual data as well as enable researchers to study new research questions previously not feasible.

While GLLMs are powerful, they come with their own challenges and limitations. GLLMs require a new way of working to ensure valid outputs and safeguard construct validity. In many ways GLLMs require a way of thinking that falls between interacting with a human and a non-generative machine learning model. Their rapid rise to popularity is accompanied by a lack of guidance on how to use GLLMs effectively and properly for academic research. For example, the recent overview of textual analysis methods in accounting by Bochkay et al. (2022) does not cover GLLMs. This paper provides a comparison, framework, and discussion to help researchers adopt and evaluate GLLMs in their research projects.

I start by comparing the GLLM approach to existing textual analysis techniques and to using human research assistants. The summary is that state-of-the-art GLLMs can solve any textual analysis task solvable using non-generative methods. However, non-generative techniques can be substantially cheaper and faster for less complex problems that are feasible to solve using simpler techniques. Where GLLMs start to become viable is for problems that would otherwise require...
complex ML machinery or human coding. GLLMs are substantially cheaper and faster relative to human RAs and can complete equivalent work for many textual analysis tasks. However, well incentivized RAs will be easier to work with for small sample tasks that don’t require the scalability benefits of a GLLM.

Next I provide a five-step framework that researchers can use to solve a textual analysis problem using a GLLM. This framework focusses on the implementation details and challenges specific to using GLLMs and complements the general textual analysis guidance by Anand et al. (2020) and Bochkay et al. (2022). The framework covers understanding your task requirements, selecting the right GLLM model and instruction approach, developing your instruction prompt, evaluating the construct validity, and finally executing your task. I also include several appendices with technical guidance and code to make it easier to get started with GLLMs. Following the guidance and steps in this framework helps researchers to be more successful in applying GLLMs to their textual analysis problems. And it also provides co-authors, reviewers, and editors with the necessary guidance to evaluate and analyze papers that use the GLLM approach.

Finally, I provide a discussion on several key considerations when using GLLMs for academic research. These considerations include potential biases, attributability concerns, and limitations regarding recent events in the GLLM training data. Additionally, I provide guidance to address replicability challenges and data privacy and copyright concerns. Lastly, I discuss current GLLM developments, such as multi-modal support and GLLMs for other languages, and provide a discussion on the use of GLLMs for writing and programming tasks.

This paper makes three contributions. First, it extends Bochkay et al. (2022) by highlighting the use of Generative Large Language Models as a powerful research method for accounting projects involving textual data. This paper helps researchers to consider and evaluate
the use of GLLMs for their project relative to the existing textual analysis methods. Second, it raises attention to the challenges and questions surrounding GLLMs, such as construct validity, biases and knowledge limitations, replicability, data privacy, and attributability concerns and provides guidance on how to address these concerns when using GLLMs for accounting projects. Third, it offers practical resources and guidance to help researchers get started with GLLMs for their own projects, including a conceptual framework, a working example including Python code, prompt engineering tips, and multiple other technical guides through a companion GitHub repository. Taken together, this paper equips researchers with the necessary knowledge and tools to harness the potential of GLLMs and editors and reviewers with the knowledge to better evaluate papers that use the GLLM approach.

2. Generative Large Language Models

2.1. What are GLLMs?

Generative Large Language Models (GLLMs) represent a class of machine learning models which consume and generate human-like text. Generative Pre-trained Transformer (GPT) models are the most common type of GLLM, but other generative architectures exist. GLLMs have three key characteristics. Firstly, they are built on large-scale neural networks with billions of parameters, allowing them to capture intricate patterns and relationships in the data. This large scale enables GLLMs to develop a basic, yet limited, reasoning ability and world understanding. Secondly, GLLMs are generative in that they take natural language text as input (the prompt) and generate natural language text as output (the completion). Finally, GLLMs are initially trained and fine-tuned for general-purpose tasks, enabling them to adapt to a wide range of applications.

GPT and BERT (Bidirectional Encoder Representations from Transformers) models are both part of the transformer architecture family, which originates from the development of the attention
mechanism (Bahdanau, Cho, and Bengio 2016; Vaswani et al. 2017). BERT models, introduced by Devlin et al. (2019), provide a way to represent text using its surrounding context instead of treating every word as independent. This makes BERT a powerful building block as part of a broader machine learning pipeline, such as for classification or other sequence-to-sequence tasks. In contrast, GPT models, developed by OpenAI, emphasize generative capabilities, and provide an end-to-end approach where both the input and output are natural language (Radford et al. 2018; 2019; T. Brown et al. 2020; OpenAI 2023). BERT models and GPT models share similarities, such as their transformer architecture and their transfer learning abilities (e.g., Siano and Wysocki 2021). The fundamental difference is how one adapts them to solve their textual analysis problem. A BERT model is adapted through the training procedure and ML machinery that one builds around it. In contrast, a GPT model is adapted through the development of a natural language prompt and potentially fine-tuning it for those types of prompts. The GPT approach makes it possible to get usable results more quickly for many textual analysis tasks. In the next section I discuss the pros and cons of GLLMs and highlight the types of tasks for which they are a good fit.

2.2. Comparison to current methods

2.2.1. Non-generative textual analysis and ML techniques

GLLMs have several features that make them stand out from current textual analysis approaches. Firstly, GLLMs take text as input and yield natural text as output. Natural text is a powerful way to embed and express complex tasks; it is how we humans transfer information. Any task and information expressible through natural language can be part of the instruction to a GLLM. Similarly, any outcome or output expressible through natural language can be an output by a GLLM. This is significant because it makes it easier to adapt a GLLM to solve a researchers’ task. Adapting a GLLM to a specific task only requires formulating the instructions using human
language without the need to develop a custom machine learning setup. To illustrate, Hassan et al. (2019) identify conversation about political risk in conference calls by training a pattern-based sequence-classification method on a dataset of (non-)political texts in order to identify bigrams that reflect political conversations. They construct a training dataset based on political and non-political texts based on newspaper articles and textbooks. This works, however, the GLLM approach would only require showing the model a part of the earnings call and asking the question “does this section discuss political risks, yes or no?”. Furthermore, the rigid nature of non-generative ML approaches can make it challenging, or impossible, to express more complex or specific tasks. Expressing a task through natural language helps in that regard as it is more flexible and easier to adjust. For example, Brown et al. (2018) try to identify how firms adjust their risk factor disclosures when a peer firm receives an SEC comment letter. Identifying specific types of changes to text is a challenging task so they limit their primary analysis to only looking at changes in the number of words. The GLLM approach would make it possible to identify and classify specific types of year-over-year changes. Moreover, one could even include details about the SEC comment letter to identify only changes that relate to the letter. One could achieve this by showing the risk factors and a summary of the comment letter (which could also be generated by the GLLM), followed by the question “Did the risk factor change in response to the comment letter, yes or no?.”

Secondly, the scale of the original training procedure and the large size of GLLMs enable them to exhibit basic forms of reasoning and have a basic understanding of the world.\(^2\) This makes it easier to instruct a GLLM model to perform a task relative to a non-generative approach. Specifically, non-generative approaches require a large training sample or researcher defined rules.

\(^2\) To illustrate, the GPT-4 model by OpenAI is likely trained on trillions of words and cost over $100 million to train. See, https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/
(such as word lists) to perform a specific task. For GLLMs it is often enough to describe the task without the need for examples (called “zero-shot”) or with only a small number of examples as part of the prompt (called “few-shot”). For example, Li et al. (2021) use a complex word embedding approach and a manual check to extrapolate the culture seed-words from Guiso et al. (2015) to a broader word-list of the five culture values. This is a valid approach, but a GLLM approach would make this easier. One could create an instruction showing an excerpt from the conference call, the seed-words, and ask the model to identify all sentences that relate to each of the five culture values. Doing so would not require first constructing a broader word list. Similarly, Call et al. (2022) use a sophisticated machine learning algorithm to analyze conference call audio to identify cases of laughter in response to humor. They limit their focus to only successful cases of humor, as unsuccessful attempts to be funny do not elicit laughter. With current state-of-the-art GLLMs, such as GPT-4, one can instead show earnings call excerpts by the CEO and ask the question “did the CEO try to be funny, yes or no?”.

Finally, GLLMs can handle more text at once (i.e., the “context window”) relative to existing models such as BERT. The current state-of-the-art model, GPT-4, has a context window of 8,192 tokens (about 13 pages of text). Another variant of GPT-4 is under development with a context window of 32,768 tokens (about 50 pages of text). For reference, BERT models typically have a context window of 512 tokens. These larger context windows make it easier to work with longer documents, which are common for accounting research. For example, Gow et al. (2021) develop a regular expression approach to detect non-answers in conference calls. Extending their analysis to classify other types of answers using ML techniques using BERT would be challenging because the combination of a question and answer often gets too long. The ability of GLLMs to handle
larger inputs would make it possible to show it a question, the answer(s), as well as a task like “did the manager refuse to answer the question because of proprietary concerns, yes or no?”.

Despite these benefits, GLLMs also have drawbacks when compared to non-generative textual analysis approaches. The primary disadvantage is that GLLMs are large and require more processing resources, making it more expensive and slower to use. Non-generative NLP techniques will be substantially cheaper and faster for simpler tasks that don’t require the power of GLLMs. Furthermore, evaluating the construct validity of GLLMs can be more challenging due to the flexibility of the natural language aspect and the black-box nature of very large models. Smaller non-generative models, or rule-based approaches, are generally more explainable and easier to evaluate. Finally, non-generative models can be easier to run and train without the need for using third-party APIs or access to expensive hardware.

Taken together, GLLMs can solve any problem that traditional NLP methods can solve. However, for simpler problems it can be slower and more expensive to use a GLLM. The costs of using a GLLM usually scale linearly with the number of observations, which can make it unnecessarily expensive for certain tasks. For example, Griffin et al. (2021) study discussions about wildfires in 10-K filings using a simple list of wildfire related keywords. One could feed all 10-K through a GLLM model and ask, “Does this text discuss wildfires, yes or no?” and it will likely do a marginally better job than the keyword list. However, it would also be much more expensive and much slower than simply counting the keywords, which is likely already good enough for the task at hand. Using a GLLM involves a cost-benefit trade-off, and it is only worth using for problems that would be challenging to solve otherwise. A general rule-of-thumb is that GLLMs are a viable consideration for problems that would otherwise require a machine learning
model or human coding. Problems with a clear and distinct set of keywords, or an existing keyword list, wouldn’t require an ML approach or human coding and don’t require a GLLM.

With this trade-off in mind, GLLMs and non-generative methods can also be combined to provide the best of both worlds. One such combination is to use GLLMs to construct or extrapolate a training dataset for a non-generative (e.g., BERT) machine learning model that is cheaper and faster to scale. For example, Li (2010) creates a training dataset by manually classifying 30,000 forward-looking statements (FLS) along the tone and content dimensions. Using a GLLM to perform these initial training classifications would save significant researcher time. Another combination is to narrow down the sample using a non-generative method and then use a GLLM to perform the final challenging task. For example, Bozanic et al. (2018) provide a keyword list to identify forward-looking statement sentences. One could then feed these FLS sentences through a GLLM to extract or identify any more challenging properties, such as asking “What is the time-horizon of this forward-looking statement? Less than 1 year, between 1 to 5 years, or 5+ years?”.

Finally, GLLMs can be an effective tool to validate and improve the accuracy of an existing approach. For example, one could use the Bozanic et al. (2018) method to identify FLS and then feed all FLS sentences through a GLLM model to remove any bad matches and improve accuracy.

2.2.2. Human research assistants

Using GLLMs also has several advantages and disadvantages relative to human research assistants or performing the task manually yourself. The primary benefit of GLLMs is the potential for cost savings, as well as an increase in speed and scalability. GLLMs can also yield outputs that are more consistent as they are not subject to inter-rater variability. For example, Chuk et al. (2013) hand-collect management forecasts in earnings releases and compare it with a data aggregator. Going directly to the source documents is a valuable validation exercise. However doing so
manually is costly and time-consuming. GLLMs can offer an approach to automate and scale manual coding exercises, such as by training a GLLM to read an earnings release and perform a task like: “Extract all forecasts made in this press release and return it as a JSON list.”. Finally, GLLMs can exhibit super-human capabilities in terms of recalling and memorizing large quantities of information. For example, a research task that involves looking up facts in the FASB codification would be slow and challenging for most human research assistants. In contrast, a GLLM with access to that information (either through fine-tuning, or otherwise) could complete such a task in seconds.3

Despite these advantages, there are limitations to using a GLLM relative to human work. The primary limitation is that human RAs are more flexible, better at reasoning, and have incentives to try hard and produce good work. A human can be held accountable for their work while a machine cannot. A well incentivized human RA will provide high quality outputs with only minimal instructions. In contrast, a GLLM approach might require prompt engineering or fine-tuning before it starts generating high quality outputs. This makes human RAs a better alternative for complex tasks with only small samples. For example, Brown et al. (2020) manually inspect a small random sample to evaluate their approach of identifying material filing revisions. This is likely automatable using a GLLM. However, the fixed costs of doing so are only worth it for larger samples. Domain knowledge can be another limitation. GLLMs might need to learn domain knowledge through explicit examples, which might be hard to obtain or formulate. For example, Hail et al. (2018) use human coding to identify accounting scandals that are morally wrong versus legally wrong. It might be challenging to communicate to a GLLM what it means for something to be morally wrong.

3 Providing GLLMs access to large quantities of external information can require extra steps and is an area of rapid development, which is further discussed in Section 4.4.
3. How to use GLLMs?

Anand et al. (2020) and Bochkay et al. (2022) provide a comprehensive textual analysis framework. Their higher-level guidance on the importance of construct validity and their guidance on the data acquisition and preprocessing steps also apply to GLLMs. The generative natural language aspects of a GLLM, however, pose distinct challenges and require a new way of thinking that falls between interactively with a human (due to the natural language aspect) and a non-generative machine learning model (due to the machine aspect). Specifically, while it is possible to instruct a GLLM like one would instruct a human, that will often not work and can lead to unsatisfactory results. To achieve high quality, consistent, and efficient results researchers need to engineer GLLM-specific instructions, which is a skill. In this section, I provide a five-step framework that helps researchers implement GLLMs for their research projects. Additionally, Appendix C provides further technical guidance and code through a companion GitHub repository.

3.1. Define and understand your problem – Step 1

The first step is to define and understand the problem you are looking to automate. Doing so requires the evaluation of what information is necessary to know to complete the task. This may include specific domain knowledge, contextual information, or other relevant details that are necessary to generate the desired output. Any model, irrespective of how powerful it is, will not perform well if it is lacking the necessary information to make a prediction or classification. Any information, such as specific institutional knowledge, that is likely not embedded in the standard

---

4 People with this specific skillset are generally referred to as “prompt engineers”. The following WSJ article provides a more detailed discussion: https://www.wsj.com/articles/chatgpt-ask-the-right-question-12d0f035
5 GLLMs sometimes “hallucinate”, which is the generation of incorrect or made-up information. This behavior is, in most instances, the result of the model’s inability to refuse to provide generations on topics or tasks for which it does not have sufficient background information.
GLLM weights will need to be provided to the model, either through examples, instructions, or updating the model weights (i.e., fine-tuning).

One of the best ways to develop a deeper understanding of the task you are looking to automate is to perform the task manually yourself on a set of examples and ask yourself why you made the decisions you made. What information did you use, and is it plausible that the GLLM you plan to use will “know” that information or will you need to provide it with that information? For example, assume you want to identify footnote text relating to FASB ASU 2014-09. You might be able to complete that task easily because you know what FASB ASU 2014-09 relates to, but the GLLM might have no idea, or worse, a flawed understanding of it. Including an explicit description or definition of FASB ASU 2014-09 in the instruction might help with that.

3.2. Decide on the approach and model – Step 2

Once you understand your task and the information necessary to perform that task, you can decide on the GLLM approach and model. Different GLLM models have different strengths and limitations, so I recommend deciding your approach first and then choosing an appropriate model. There are three approaches that one can take to instruct and control the output of a GLLM: zero-shot, few-shot, and fine-tuning.

The zero-shot approach works by providing the GLLM only with your instructions and the data on which you would like to perform the task, without giving any examples. The few-shot approach works like the zero-shot approach, except you provide a few examples, usually one to five, as part of every prompt to complement your instructions. Finally, the fine-tuning approach
works by embedding a larger set of examples and the instructions into the model logic by updating the model weights through a training procedure called “fine-tuning”.6

When selecting your approach, you need to keep the following aspects in mind: your “token budget”, the complexity of your task, and your accuracy vs. effort trade-off. The token budget refers to the length of the instructions to complete a single task as well as the total number of tokens required to complete all your generations. The time and costs required to complete all your generations with a GLLM model scale linear with the number of tokens in your prompt and completion. Including examples as part of your prompt will result in more tokens required and used per task, which can limit you to models with larger context windows and increase the time and money required to run your task. Furthermore, the complexity of your task is important when selecting the appropriate approach. Simple tasks work well with the zero-shot or few-shot approach, while more complex or domain-specific tasks may require fine-tuning to achieve satisfactory results. For example, classifying whether a text discusses employees might constitute a simple task while classifying whether a paragraph should reference FASB ASU 2014-09 might constitute a complex task. Lastly, consider the accuracy-effort trade-off. While fine-tuning may yield better results, it also requires more effort, time, and expensive computational resources. To minimize cost and effort you are best to go with the simplest approach with the lowest token budget requirements that still yields satisfactory results.

Once you have settled on your approach you can evaluate which GLLM and provider would best suit your needs. The GLLM in this context refers to the exact model, such as GPT-3 or GPT-4, and the provider refers to the organization that is making the model available, such as OpenAI.

---

6 *Weights* in this context are the machine learning equivalent to *coefficients* in a regression context. Fine-tuning in this context is like fine-tuning non-GLLM models, although fine-tuning GLLMs generally require less training samples to yield similar performance.
The time required and costs increase with the size of the model, so it is best to select the smallest possible model for your requirements. Models and providers differ in their cost, speed, ability to fine-tune, their accessibility (API vs. local), and the amount of text they can handle at once. For a full overview of current models and providers, see Appendix C.

3.3. Develop your prompt (i.e., “prompt engineering”) – Step 3

A prompt is the input you provide to the model and a completion is the output the model will generate based on your prompt. Both are expressed in natural language and take the form of a string of text. The input prompt is the primary way to control the output generated by the GLLM as the prompt contains your instructions, any contextual information, and the focal data. The focal data represents the information that will change for each observation in your data. For example, a simple prompt would be “Does the following tweet relate to politics, yes or no? Tweet: <the focal tweet>”. Engineering a good prompt is the key to getting reliable and high-quality outputs out of a GLLM model.

When designing a prompt, the two primary objectives are (1) communicate to the GLLM what you expect the completion to be and look like (instructions + examples) and (2) provide the GLLM with the necessary information to generate the desirable completion (context + focal data). A model that misunderstands the instructions or that does not have access to the necessary information will generate completions of low quality. Any information that can be represented as text can be part of a prompt or completion. This includes not only narrative data but also, amongst others, tabular data, JSON, XML, XBRL, HTML, and programming code. It is hard to anticipate how GLLMs will respond to an instruction, just like with humans. You are unlikely to get it right

---

7 Zero-shot and few-shot prompts will consist of three to four building blocks: instructions, relevant context, examples, and the focal data. For fine-tuned models the instructions, relevant context, and examples are already embedded into the model weights, so the prompt generally only includes the focal data. In other words, a fine-tuned model already “knows” what the task is because we trained it for it, so we don’t have to repeat the instructions.
on the first try and it will require trial-and-error. The benefit of designing a GLLM instruction is that iteration is quicker than with human RAs. You can try an infinite number of times, GLLMs don’t get tired, distracted, or hungry after the $N^{th}$ try. Appendix D includes more tips and tricks for designing successful and efficient prompts.

When designing the completion portion of your instruction, the primary objective is to make the output machine readable and easy to parse. Completions are natural text which can be hard, or impossible, to parse with your code if you don’t design your completion with parsing in mind. GLLMs can generate completions that would be challenging to generate for human RAs. For example, we can instruct a GLLM to return the result as a JSON string. A JSON string would be much easier to parse using Python or R relative to a narrative answer. For example, let’s say we want to extract cashtag mentions out of Twitter messages and classify them as bullish or bearish. If we ask a GLLM to perform that task without being mindful of the output it might yield something like “The first cashtag is $AAPL$ and it is bullish and the second cashtag is bearish (MMM)”. This would be tedious and difficult to parse into a dataset. Instead, we could include an instruction to return a JSON list of the format “[(cashtag, sentiment), …]”. This would yield “[('$AAPL', 'bullish'), ('$MMM', 'bearish')]” which we can parse into a dataset with a single line of code. More prompt examples are shown in Appendix B.8

3.4. Evaluate the construct validity – Step 4

Evaluating the construct validity of any textual analysis approach is an important responsibility of the researcher (Bochkay et al. 2022). Evaluating generative completions comes with two primary challenges. The first challenge is that the zero-shot and few-shot approaches don’t require a training sample to evaluate against. The second challenge is that GLLMs can yield

---

8 The fine-tuning step happens after Step 3 if you choose to go that route. Details on how to fine-tune a GLLM model using the OpenAI API are provided in the companion GitHub repository linked in Appendix C.
completions that are not compatible with standard performance metrics. For example, it is easy to develop a prompt that asks many questions at once. However, how do we evaluate the quality of the completion if it consists of multiple individual elements? To safeguard and ensure the construct validity when using the GLLM approach, I recommend the following best practices.

1. Always create an evaluation set of outcomes generated by human raters, such as yourself, co-authors, research assistants, domain experts, or third-party sources. This sample needs to be large enough such that it appropriately represents your input population. For simple tasks this can be a hundred. For complex tasks this might require thousands. The standard best practices for working with human raters apply here, such as using multiple raters, checking inter-rater validity, performing the tasks independently from the GLLM completions, and appropriately incentivizing the human raters to ensure high quality work.

2. When possible, engineer your instructions in such a way that the outputs are compatible with evaluating your GLLM approach like one would evaluate a supervised machine learning approach. For example, returning a single explicit outcome (e.g., Yes / No) as part of your completion will make it easier to evaluate the performance for a classification task relative to returning a narrative outcome or many explicit outcomes at once. 9 Sometimes it is necessary to return multiple outcomes at once for cost or efficiency reasons, in which case you will need to evaluate and report each outcome independently as if they are separate classification tasks.

3. When it is not possible to evaluate your completion as a classification task, I recommend following the evaluation best practices for the closest non-GLLM approach that one would use for that specific task. For example, when using a GLLM for a clustering task one can follow

---

9 For example, when classifying sentiment using a GLLM one can design an instruction that yields a sentiment score in terms of a narrative description (“The tweet is mostly positive, but some parts are negative”), but that would be much more challenging to evaluate compared to a prompt that asks for discrete outcomes (e.g., -1, 0, 1).
the best practices for unsupervised machine learning (see e.g., section 4.3 in (Bochkay et al. 2022). Or when using a GLLM to make qualitative assessments, follow the standard best practices for evaluating qualitative interviews.

(4) Be skeptical of the completion quality. GLLMs are trained to mimic human responses as closely as possible. However, GLLM responses are not human responses and should never be treated as such. Outputs generated by GLLMs tend to look great at first inspection but can fall apart with closer inspection. As human researchers, we are susceptible to a confirmation bias where we want the completions to be of high quality, so it is easy to over-estimate the quality of a GLLM completion. Exercising a healthy level of skepticism helps to overcome that.

3.5. Run the predictions and parse the results – Step 5

Once a satisfactory construct validity is reached, the approach can be scaled up to generate completions for all the data observations. This step is referred to as the “inference” step. There are two ways to generate completions for your full sample: interacting with an external third-party API or running the model locally on a powerful (Linux) machine with one or more accelerators (generally GPUs or TPUs). Using a third-party API, such as the OpenAI API, will be easiest and fastest for most use cases. However, there are also GLLMs available that one can download and run locally. There are several advantages to running a model locally, for example in terms of replicability and data privacy, which are further discussed in Sections 4.2 and 4.3. The technical guides in Appendix C provide more guidance on using third-party APIs or running GLLMs locally.

4. Discussion

4.1. GLLM training data: biases, attributability, and recent events

Researchers using GLLMs should be aware of the potential biases, attributability concerns, and limitations regarding recent events in the GLLM training data. These models are trained on
extensive textual corpora and they may inadvertently learn and reproduce biases present in that data. For example, preliminary analysis suggests that job postings generated by ChatGPT are less inclusive and more biased relative to humans.\textsuperscript{10} These biases can also surface, for example, when using ChatGPT to classify job postings for a research project. Researchers need to evaluate the outputs for potential biases and, when necessary, develop strategies to mitigate them through their prompt engineering or fine-tuning procedures.

Attributability concerns arise from the fact that GLLMs are trained on a large diverse set of sources making it difficult to trace the origin of generated text. This may lead to issues regarding intellectual property, the need to provide attribution, and the accuracy of the information produced. To help with that it is better to control the source of important information by explicitly passing it to your model through your prompt or fine-tuning. Providing the information explicitly enables attribution to the source and safeguarding accuracy of the information. For example, generations by a GLLM might suggest that it understands the FASB taxonomy. However, it is unclear what source material was used to develop that understanding and it might be incorrect or out-of-date. Passing it the explicit portion of the FASB taxonomy that you would like it to use as part of your prompt resolves that concern and gives greater control.

Lastly, GLLMs are trained on historical data up to a certain point. Any events or changes outside of the training sample will not be reflected in the output of the model. Researchers must account for these limitations when interpreting the results and, if necessary, incorporate up-to-date information through their prompts or by fine-tuning the model using more recent data.

4.2. Replicability

Ensuring replicable research is a challenging but important task when conducting research, especially when working with complex data and procedures. The GLLM approach faces two specific replicability challenges. First, GLLMs are powered by large neural networks with billions of parameters, which makes it hard, if not impossible, to fully understand the decision criteria applied by the model. Second, the expensive computational resources required to run most GLLMs limit the accessibility of these models. The easiest way to access a GLLM is through a third-party API, which is convenient but comes with replicability concerns. Models accessed through a third-party API can change, become unavailable, or become more expensive over time. These concerns are analogous to working with human coders, who might not be available in the future either. There are several things a researcher can do to mitigate replicability concerns.

The first method is to not rely on third-party APIs and to interact with your GLLM model locally using your own hardware. The technical materials in Appendix C provide more guidance on doing so. Running the model on your own hardware yields more control and replicability. However, it does not resolve all replicability concerns. For example, access to the required specialized hardware can be prohibitively expensive, not possible, or require a prohibitive degree of technical knowhow and expertise.11 Furthermore, the large size and complexity of GLLMs make it hard to yield fully persistent results. Running the same model with the same prompt on different sets of hardware can create minor differences in the completions making exact replication challenging.12 For most projects the benefits of using a third-party API will outweigh the challenges that come with running a GLLM locally.

11 Currently, the majority of GLLMs can only run on powerful NVIDIA GPUs, some of which are unavailable in countries such as China due to export or import restrictions.
12 This limitation is not specific to GLLMs and applies to most large machine learning models.
Fortunately, there is a second method to improve replicability that works across all methods. The philosophy behind these best practices is to assume that one can lose access to the exact source of the generations. The exact source could be a GLLM through a third-party API, human coders, or an exotic non-generative machine learning model. These best practices include:

(1) Take detailed lab notes and extensively disclose your research choices and procedures. Being transparent in your empirical choices, such as your data, your task, your prompt design, your approach, and your model(s) goes a long way. Sharing your code, when possible, is also helpful to eliminate discrepancies in the research approach that might yield replication challenges.

(2) Always store and backup the raw prompts and completions as they come out of the model. Having access to the raw prompts and completions is the next best thing to having access to the source that generated them.

(3) Make your training data and/or evaluation data available, when possible. Sharing your evaluation data and the corresponding generations makes it easier to understand what the model is doing. Understanding what the model is doing makes replication much easier.

(4) Try to make your GLLM generations as deterministic as possible. GLLMs inherently exhibit a degree of randomness as this randomness is what enables these models to generate “creative” responses. This creativeness, however, can be managed through the “temperature” hyper-parameter. Setting this parameter to zero will make the GLLM pick the highest probability tokens, which makes the generations as close to deterministic as possible.\(^\text{13}\)

4.3 Data privacy and copyright

When using GLLMs researchers need to be mindful of the privacy of their data, specifically when using third-party APIs. Sharing research data as part of your prompts or fine-tuning can

\(^{13}\) Note that setting the temperature to zero does not necessarily make the perfectly deterministic. Small variations can still exist, especially when using a third-party API.
result in data privacy concerns. Firstly, you are sharing your research data with the organization running the API. Secondly, the API provider might use your prompts and generations to improve and re-train their models. This retraining can result in GLLMs unintentionally disclosing sensitive information in their future outputs to others.\textsuperscript{14} The same logic applies to data that is under strict copyright restrictions. To mitigate this risk researchers should implement data anonymization techniques when preparing their training and prompt datasets. Furthermore, it is best to avoid using GLLMs through third-party APIs when data sharing is not allowed or desirable, especially when the API providers state in their terms of service that prompts can serve as additional training data.

4.4. Current GLLM developments

GLLM developments are happening at a quick pace with several areas receiving notable attention and resources. Firstly, currently available GLLMs can only accept natural language data as input. However, multi-modal support is on the horizon and models like OpenAI GPT-4 will soon support image inputs in addition to text inputs. Similar developments are underway for audio inputs. Multi-modal support will result in an even greater range of tasks and problems that are automatable using GLLMs, such as those involving data visualizations, challenging PDF files, corporate presentations, video material, and audio transcripts.

Speed optimization and lower processing requirements are another area of rapid progress. These optimizations will make GLLMs more usable as well as more affordable and scalable. To illustrate, the ChatGPT API (\textit{GPT-3.5-turbo}) is already ten times cheaper relative to its predecessor, the GPT-3 Davinci API. Faster speeds and lower costs will make GLLMs attractive for a wider range of problems over time. Furthermore, there are multiple ongoing developments

\textsuperscript{14} For example, you can try to ask ChatGPT what it knows about your name, and it will generally yield a response detailing (some) public information about yourself. That information about you was part of the original training data collected from the internet. Any prompts you provide might be added to the training data and surface in the same way.
to improve open-source models to make them more competitive with closed-source models, such as the GPT-J, GPT-NeoX, and LLaMa models (Touvron et al. 2023; Wang and Komatsuzaki 2021; Black et al. 2022). Better open-source models will make it easier to run GLLMs locally, which will help reduce concerns around replicability and data privacy.

Also, multiple developments are underway to connect GLLMs to information sources beyond its original training data. These implementations, such as the ChatGPT Plugins by OpenAI, are showing potential to improve the world understanding and information lookup capabilities of GLLMs.¹⁵ This has the potential to allow the automation of textual analysis tasks that require large amounts of context or domain knowledge without the need to fine-tune the model. For example, one could create a plugin that can search the FASB taxonomy and hook it up to GLLM model to perform a domain-specific textual analysis task on 10-K footnotes. Connecting GLLM models to external sources has many applications. However, it also raises several questions related to reliability and replicability, which future researchers will need to be mindful of.

Finally, most currently available GLLM models focus on the English language. These English GLLMs do have basic capabilities in other languages. However, the state-of-the-art performance is currently limited to English. Multiple developments are in progress to provide GLLMs with other language focuses, such as the Ernie model for Chinese NLP tasks (Sun et al. 2021). These non-English GLLM models will make it easier to conduct textual analysis research on non-English documents.

4.5. GLLMs for writing & programming

While this paper focuses on using GLLMs for textual analysis tasks, it is worth discussing their increasing applications in other parts of the research process, such as AI copy editors and

programming assistants. Many of the above discussions apply to these applications as well, but there are a few specific challenges that I will discuss in more detail below.

When using GLLMs to adjust, check, or improve academic writing, it is critical to ensure proper attribution. Generated content requires fact-checking as GLLMs can produce plausible-sounding but false information or references. Researchers should verify the accuracy of the writing generated and exercise caution in relying on the model-generated content without thorough examination.

In the context of programming, GLLMs can serve as useful productivity tools that facilitate code development and debugging. However, the research and coding problems encountered in academia are often rare and unusual, differing from commonly used coding problems found on the internet. Current GLLMs excel at solving coding problems that are prevalent in the training data but can struggle with more complex, rare, or domain-specific coding tasks. Minor mistakes in the code base for a research project can have major impacts on the validity of the final results. Code review is essential when working with code written or modified by a GLLM. This is especially true when GLLM coding assistants write code that the researcher wouldn’t be able to write otherwise. To help with that, I echo the call by Bochkay et al. (2022) for young researchers to (still) invest time developing textual analysis and coding skills and for institutions to dedicate resources to provide the necessary training and guidance.

5. Conclusion

Generative Large Language Models (GLLMs) offer significant potential for accounting research involving textual data. GLLMs provide a powerful, versatile, and cost-effective alternative to traditional machine learning methods and human research assistants. This paper highlights the advantages of GLLMs, addresses challenges such as construct validity, biases,
knowledge limitations, replicability, data privacy, and attributability concerns, and offers practical resources and guidance for researchers. By providing a comparison, framework, and discussion, this paper equips researchers with the necessary knowledge and tools to harness the potential of GLLMs and assists editors and reviewers in evaluating papers that employ the GLLM approach.
References


Electronic copy available at: https://ssrn.com/abstract=4429658
# Appendix A: Glossary

<table>
<thead>
<tr>
<th>Terminology</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LLM</strong></td>
<td>Large Language Model</td>
</tr>
<tr>
<td><strong>GLLM</strong></td>
<td>Generative Large Language Model</td>
</tr>
<tr>
<td><strong>GPT</strong></td>
<td>Generative Pre-trained Transformer</td>
</tr>
<tr>
<td><strong>Token</strong></td>
<td>A token is the unit of observation for a GLLM. A single word generally consists of multiple tokens. One token is on average four characters in the English language.</td>
</tr>
<tr>
<td><strong>Prompt</strong></td>
<td>The text input provided to the GLLM.</td>
</tr>
<tr>
<td><strong>Prompt engineering</strong></td>
<td>The task of developing a prompt for your task.</td>
</tr>
<tr>
<td><strong>Completion</strong></td>
<td>The text output generated by the GLLM.</td>
</tr>
<tr>
<td><strong>Zero-shot</strong></td>
<td>Providing a prompt without any examples, just the instruction.</td>
</tr>
<tr>
<td><strong>Few-shot</strong></td>
<td>Providing a prompt with a few examples and the instruction.</td>
</tr>
<tr>
<td><strong>Fine-tuning</strong></td>
<td>Retraining (part of) the model weights for a specific task.</td>
</tr>
<tr>
<td><strong>Model weights</strong></td>
<td>The parameter values (“coefficients”) of the neural network.</td>
</tr>
<tr>
<td><strong>API</strong></td>
<td>The communication protocol used to interact with a service through the internet.</td>
</tr>
</tbody>
</table>
Appendix B: Examples

This appendix provides a few examples of how one would use GLLMs and prompt-engineering to automate various types of textual analysis tasks. To save space, the prompts are shown as their zero-shot version, but they would be easily adapted to the few-shot or fine-tuning approaches.¹⁶

Identifying statements about political risk:

Consider a research project that requires identifying statements in earnings calls related to political risk. A GLLM can solve this task by prompting it with:

Identify whether the earnings call transcripts below discuss political risks or not.

1. gaming industry is currently supporting a ballot initiative to amend the constitution to authorize an increase in the —BET— limits
2. Now on to the third quarter results. Cano Health continued to achieve steady organic growth in Q3. Total membership grew 4.6% from the second quarter to approximately 295,000 members, an increase of 40% year-over-year.
3. governments both President Obama’s budget proposal and separate legislation —PENDING— in Congress would provide funding to continue the Medicaid stimulus for another

Return the result in the following JSON format:

```json
[   {"index" : i, "discusses_political_risk" : Yes/No} ]
```

JSON =

Which yields the following output when passed to GPT-4:

```json
[   {"index" : 1, "discusses_political_risk" : "Yes"},
    {"index" : 2, "discusses_political_risk" : "No"},
    {"index" : 3, "discusses_political_risk" : "Yes"}
]
```

¹⁶ To make things easier to read I’ve added indents and newlines, but the GLLM does not need those so removing them will free up some token budget when ran at scale.
**Identifying CEO Humor:**

Consider a research project that requires classifying CEO statements in an earnings conference call as trying to be funny (yes) or not (no). A GLLM can solve this task by prompting it with:

Classify the CEO statements below as trying to be funny (yes) or not (no):

1. Thank you, everyone, for joining us today. I must say, our quarterly results are so good, even my dog started investing in our stock!
2. Thank you, everyone, for joining us today. I hope that you have a wonderful day!

Return the result in the following JSON format:

```json
[{
  "index" : i, 
  "funny" : Yes/No
}]
```

JSON =

Which yields the following output when passed to GPT-4:

```json
[
  {"index" : 1, "funny" : "Yes"},
  {"index" : 2, "funny" : "No"}
]
```

**Extracting EBITDA-like numbers for the MD&A section:**

Consider a research project that requires extracting EBITDA-like numbers from a company’s Management Discussion and Analysis (MD&A) section. A GLLM can solve this task by prompting it with:

Extract all EBITDA-related references and quantitative amounts from the paragraph below. Do not convert values and report them as reported.

> In the third quarter of 2021, our EBITDA increased to $1.2 billion, primarily driven by higher sales volumes and improved operational efficiency. Our Debt-to-EBIDAR increased by 20%. This growth was partially offset by increased raw material costs and unfavorable foreign exchange impacts.

Return the result in the following JSON format:

Electronic copy available at: https://ssrn.com/abstract=4429658
JSON =

Which yields the following output when passed to GPT-4:

```
[  
  {"name": "EBITDA", "value": "$1.2 billion"},
  {"name": "Debt-to-EBIDAR increase", "value": "20%"}
]
```

**Clustering task:**

Consider a research project that requires clustering company press release headlines into unsupervised topic categories. A GLLM can solve this task by prompting it with:

Cluster the following company press releases into common categories:

1. Company A announces a new partnership with Company B to develop innovative solutions for the healthcare industry.
2. Company B reports its financial results for the second quarter of 2021, highlighting a 15% increase in revenue compared to the same period last year.
3. Company C unveils its latest initiative to reduce their carbon emissions, a $2 Billion investment in an all-electric fleet of company vehicles.

Return the result in the following JSON format:

```
[  
  {"index": ..., "category": ...}
]
```

JSON =

Which yields the following output when passed to GPT-4:

```
[  
  {"index": 1, "category": "Partnerships"},
  {"index": 2, "category": "Financial Results"},
  {"index": 3, "category": "Sustainability Initiatives"}
]
```
Data cleaning task:

Consider a research project that needs to work with PDF documents that are challenging to convert to text. The extract text is often split and cut-off, making it hard to extract sentence. A GLLM can fix this problem by prompting it with:

Extract valid sentence from the following text that got split up:
In 2018, we entered the third year of an ambitious 10-year goal cycle. The 2025 Sustainability Goals we are now pursuing reflect a heightened commitment to going beyond compliance and thinking holistically about how our operations and products affect the world and every life in it. For additional information on the development of our 2025 Sustainability Goals, see global challenges.

Return the result in the following JSON format:

```
[
    {"i": ..., "sentence": ...}
]
```

Which yields the following output when passed to GPT-4:

```
[
    {"i": 1, "sentence": "In 2018, we entered the third year of an ambitious 10-year goal cycle."},
    {"i": 2, "sentence": "The 2025 Sustainability Goals we are now pursuing reflect a heightened commitment to going beyond compliance and thinking holistically about how our operations and products affect the world and every life in it."},
    {"i": 3, "sentence": "For additional information on the development of our 2025 Sustainability Goals, see global challenges."}
]
```
Appendix C: Practical resources

I provide several practical resources to make it easier to get started with GLLMs.

These resources include:

- A list of GLLM providers and models.
- A code file (Jupyter Notebook) with a full working example.
- A guide on fine-tuning GLLMs (*not yet available*).
- A guide on running GLLMs directly without an API (*not yet available*).

These materials can be accessed through the following companion GitHub repository:

[https://github.com/TiesdeKok/gllm_companion](https://github.com/TiesdeKok/gllm_companion)
Appendix D: Prompt engineering tips and best practices

**GLLMs generate text left to right, so their own generations become part of the prompt.**

The left-to-right aspect of GLLMs is useful to keep in mind as it (1) might create unexpected behavior when you are trying to perform multiple tasks at once, and (2) can be utilized to increase the quality of the completions. A common technique to improve completion quality is to ask the model to explain itself before generating a final outcome. Forcing the model to also generate an explanation will generally create better and more consistent output as the model is using its own explanation as part of the classification decision, which improves quality.

**Optimize the number of tokens used by your prompt and completion.**

Tools like tiktoken and the OpenAI tokenizer are helpful tools to understand how many tokens your prompt and completion are representing.¹⁷ Optimizing the number of tokens can save you substantial time and money. Reducing the number of tokens by 10% will generally yield a 10% speed up and a 10% cost savings. Another method to save on tokens is to include multiple observations in a single instruction, for example by having it return a list instead of a single item.

**Be specific and to the point, less is often more.**

When using a zero-shot, or few-shot, approach it can happen that the GLLM ignores certain parts of your instructions, which can make the completions hard to control. This behavior is usually the result of your instruction being too vague or ambiguous or because your instructions are too long. In my experience, the model will by default assign different weights to different instructions in a way that is not currently well understood. To alleviate this, it can help to remove less important instructions and to make the instructions shorter and more explicit. Another approach that one can take is to explicitly tell the GLLM what instructions are more important. For example, by ordering your instructions in order of importance and including a note to explicitly explain the order.

**Small differences in the prompt can make big differences in the completion.**

Small changes in the instructions usually don’t make a big difference when providing instructions to a human, however, they can make a big difference when instructing a GLLM. For example, including uncertain words such as “might”, “possibly”, “could”, or “can” often cause the model to

generate significantly different outputs relative to when they are excluded. The influence of a single word or sentence is hard to predict, but trial-and-error and experimentation are effective ways to tweak the prompt to improve the completion quality and improve consistency.

**If you can’t get your prompt to work, consider using a smaller model with fine-tuning.**

The largest GLLM models are best for zero-shot or few-shot approaches, however, sometimes a task is too complex or specific for a single prompt. In those cases, you can achieve better performance by taking a smaller GLLM model and fine-tuning it on a reasonable number of training examples. A fine-tuned GPT-3 Babbage model will generally outperform a zero/few-shot approach with ChatGPT (*GPT-3.5-turbo*) by a significant margin, even though the GPT-3 Babbage model is much smaller (~135 times fewer parameters) relative to ChatGPT.

**Combining prompts (i.e., prompt-chaining).**

For more complex tasks it might not be possible or effective to try and do everything through a single prompt. In this scenario a powerful technique is to chain a series of prompts together, called “prompt chaining”. For example, you might want to analyze the entire risk factor section of an annual filing, which likely won’t fit inside a single prompt due to the length. To deal with that, you can split up the risk factors, feed each risk factor into the GLLM with a prompt to summarize it, and then combine all the summarized risk factors together into the final prompt. Similarly, to increase the quality of your completions you can feed the models completions back into the model with a prompt asking it to check or improve the previous completion and generate a new completion in the process. The *LangChain* Python library provides functionality to make it easier to chain prompts together.\(^{18}\)

**Other useful prompt engineering resources:**

- [https://github.com/openai/openai-cookbook](https://github.com/openai/openai-cookbook)

\(^{18}\) [https://github.com/hwchase17/langchain](https://github.com/hwchase17/langchain)