nature portfolio

Peer Review File

Benchmarking Cell Type and Gene Set Annotation by Large Language Models with AnnDictionary

Corresponding Author: Dr Stephen Quake

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
This work presents a tool to benchmark cell type annotation. The topic is certainly very important and useful for the biomedical research community. However, the writing of the work is too abstract to judge. There are no details of methodology. The annotation data, as the authors wrote in the papers, are not released. So with mostly just accuracy reports, it is hard for me to judge the quality of this work. More detailed comments are below.

Major Comments
1. Source of Performance Improvement:
(1) The authors claim that on the same LLM basis, AnnDictionary increases annotation consistency from 58% to 78%. Please clearly state whether the parameters of the LLM used in the experiments are identical or if there are any differences.
(2) Please provide a detailed ablation study to analyze the sources of the performance gain. Clarify which steps or optimization measures contribute to the observed 20% improvement.
2. Impact of Dataset Differences on Results:
It is mentioned in the text that the 58% figure comes from the first edition of the 2022 Tabula Sapiens dataset, whereas the 78% figure is based on Tabula Sapiens v2. These two datasets differ significantly in terms of scale and diversity. To ensure fairness in the results, please conduct performance comparison experiments using the same dataset from Tabula Sapiens or Tabula Sapiens v2.
3. Comparison of Gene Annotation Methods:
Please supplement the manuscript with quantitative comparisons of AnnDictionary's 'gene annotation' methods against other similar tools.
4. Data Leakage and Bias Risk:
In the Tabula Sapiens v2 dataset used for model evaluation, more than half of the manual annotation information has not been published. Such unpublished information may have appeared in other published literature. If all or part of this information was included in the LLM training data, it could lead to data leakage, resulting in biased evaluation results.
5. Performance Issues on Unreleased Annotations:
The observed performance improvement may be misunderstood as a comprehensive enhancement, but the evaluation results focus solely on the portion of manually annotated markers that is publicly available. For the unreleased portion, model performance may decline. Please revise the relevant descriptions accordingly.
6. Comparison with Existing Tools:
A study published in 2024, "Reference-free and cost-effective automated cell type annotation with GPT-4 in single-cell RNA-seq analysis", also supports Python and multiple LLMs. Relevant documentation is available in the OmicVerse Tutorials (https://omicverse.readthedocs.io/en/stable/Tutorials-single/t_gptanno/).
7. Centralized Implementation in Scanpy:
On page 5, the manuscript states: "As previously mentioned, several LLM-based automations have been developed and tested, but there does not exist, to the authors' knowledge, a centralized implementation of them in python built on top of Scanpy, the predominant scRNA-seq analysis python package. Thus, we implemented a variety of annotation tools." However, OmicVerse already supports Python and integration with Scanpy.

Minor Comments

Insufficient Figure Legends:

Several figures lack adequate legends. For example:
(1) The numerical meaning in Figures 3B and 3D.
(2) The specific meaning of different colors in Figure 4.
(3) The specific meaning of the [0,1] values in Figure 5A.

It is recommended to add or improve these legends to help readers better understand the information presented in the figures.

(Remarks on code availability)
We downloaded and implemented the code. The setup works fine. However, there is no data to test the program. Thus, we are not able to validate the results of the submission.

Parallel analysis of large-scale data is generally conducted on HPC servers. It is recommended that a readily usable Singularity SIF image file be provided.

Reviewer #2

(Remarks to the Author)
In the manuscript, Crowley et, al. developed the software AnnDictionary, which is an open-source Python package for the cell-type annotation of single-cell RNA-seq data using common LLMs such as OpenAI's ChatGPT and Google's Gemini. Specifically, AnnDictionary can process multiple datasets at a time, annotate cell types and gene sets, and support multiple LLM models. They have also benchmarked different LLM models' ability to annotate cell types from the resolution of cells, cell types, and tissue-cell types, which has provided valuable information for LLM model selection. Moreover, they have also explored the inconsistency between manual annotation and LLM annotation in basal and stroma cells. They found that LLMs can identify the major cell types in the cluster population. They have also found that in the majority of cells, LLMs showed great consistency with each other and agreed well with manual annotation. The study provides a tool AnnDictionary, for practically cell type annotation using LLMs and gives suggestions on model selection by represents as a comprehensive benchmark study. It also provides things that need to be taken care of when using LLMs for cell type annotations. AnnDictionary may have limited novelty when there is GPTCelltype that uses GPT-4 for cell type annotation. The benchmark part can still be very helpful in practical cell type annotation using LLMs though it may have limitations. Below are my detailed comments:

Major comments:
1. The manuscript misses the methods part, which is very important in the process of peer review.
2. I suggest benchmarking the performance from the level of samples and the level of atlas. I suspect the evaluation is on the atlas level of Tabula Sapiens v2. Sample-level evaluation should also be performed in at least three different tissues. Sample-level benchmarking is more important since this is the practical performance of users. Also, I would like to know the performance of individual LLMs at different levels of cell type annotation (e.g. T cells / CD4 T, CD8T). Moreover, I would like to know the performance of each LLM when providing a set of cell types for their reference instead of using the strategy of zero-shot. This is also the common situation when users know or have expectations about what cells the sample contains.
3. One major concern for me is the use of GPT-4o to determine the best clustering resolution and Claude 3.5 Sonnet to correct cell type labels since both two methods are involved in benchmarking. For the resolution, you may try setting a resolution that is high enough to separate most cell types (e.g. resolution = 2). For the uniform set of cell types, I believe it can be manually corrected.

Minor comments:
1. For the instances that require manual correction, you might need to have a troubleshooting page for potential issues when using the tool.
2. The number of top marker genes is a concern for me. The tools should let users select the number of marker genes to use. The paper should also explore the ideal number of marker genes regarding different levels of cell-type annotation tasks.
3. Please explain the cell, cell-type, and tissue-cell type level resolution in the methods part.
4. I suggest providing an example of the original annotation results shown in the form of UMAPs from each LLM.
5. Please provide an example of how the gene annotation function may help. This sounds like the Gene Ontotation function which is pretty common.
6. In the abstract, it says We found that cell type annotation with AnnDictionary outperformed previous annotation with the same LLM by ~20% (58% vs 77%), and with other LLMs by ~25%. However, I didn't find any evidence to support the saying. For example, have you benchmarked with GPTCelltype?

(Remarks on code availability)

Reviewer #3

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career

Researchers who co-review manuscripts.

(Remarks on code availability)

Reviewer #4

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

(Remarks on code availability)

Reviewer #5

(Remarks to the Author)
General assessment of the work:

The use of LLMs in research has been of great interest and new packages that ease their use for biological data processing are very welcomed. Such packages ease the evaluation, comparison, and assessment of the usability of LLMs for various tasks and various types of datasets. The annotation of cell types in single cell (and spatial transcriptomics) experiments is one such task for which the usage of LLMs are explored. Further characterization of the cases and extent to which LLMs are appropriate for this task is needed, which in turn requires good software support. Finally, it is welcomed that the authors acknowledge and enable the interaction between human and LLM so educated decisions can be made in unclear cases.

Major comments:
1) The most important component of this manuscript is the underlying software. In the fast paced development of the field, it is of utmost importance to create extensible packages. The authors claim: “AnnDictionary is extensible and can grow to accommodate additional methods”, therefore special attention should be made to ensure these claims are valid. The standard way to ensure extensibility is by modular design and separation of responsibilities. The code base however does not show such characteristics at the abstraction level beyond the definition of functions. Please revise and refactor your code so that it supports maintainability and extensibility well. Pylint can help identify vulnerabilities and smells at the code level.
2) The authors describe the cell type annotation method as “First, we put the LLM in a feedback loop with Scanpy to create an agent that attempts to determine cluster resolution automatically from UMAP plots.” The human cell annotators use UMAP plots due to the limitations of human perception that requires visualizing the data in 2D, while machine learning models can work with the higher dimensional data (e.g. the 2000 most variable genes) to find clusters. Can the authors provide arguments for this method or references where UMAPs are used as input to (AI) models?
3) “To assess label agreement between the LLMs and manual annotations, we used an LLM to automate the comparison of cell type labels because this allowed us to meet the scale requirements of the present study, which involved assessing the agreement of ~10,000 unique pairs of labels”. Can the authors clarify why the usage of LLMs is necessary in this case? Comparison between annotations may be done programmatically without exposure to hallucinations.

Minor comments:
1) Subfigures in figure 4 feature ellipses one each. It is unclear why these ellipses were necessary and how they relate to the main text. Can the authors clarify these, preferably adding their meaning to the caption? Are these on Figure 4 A the top 10 cell types later shown in the confusion matrix at Figure 5 A?
2) The supplementary figures are missing the x axis label for tissues, which is understandable for Figure S1 due to size limit, but please consider adding it to Figure S2 for enabling visual investigations of model performances per tissue and cell type.
3) The authors hint several times at the disagreement between human expert and LLM labels. Limitations are also elaborated in the Discussion section. Although beyond the scope of this current manuscript, using their tool would potentially enable assessing the time spent with or without using LLMs for annotating cell types for researchers at different career stages which would be beneficial for the community.

(Remarks on code availability)
Installation was possible, however it would have been much more user friendly and up to current standards, to provide a package through PyPi. It would also have been appreciated to include a conda environment and/or a container to help installation of dependencies used for the tutorials.

I could not install cellxgene_census on my Fedora machine, so I was not able to run the basic tutorial. I tried running the cell type annotation tutorial with a small sample data instead. I was not able to complete the tutorial within 2 hours of troubleshooting. Last error message at which I gave up was at “set_high_variance_genes_adata_dict“ after I already installed tbb:

Error processing U11 on attempt 0: No threading layer could be loaded.
HINT:
Intel TBB is required, try:

$ conda/pip install tbb
Failed to process U11 after 0 attempts.

Note that the error message suggests installing tbb with conda/pip, however the readme says pip is not supported. Please update the error message accordingly.

It is strongly recommended to use linters (e.g. Pylint, Flake8) during development which can catch code smells before they cause an issue or hinder maintainability and further development. When pylint is enabled, among others, the following smells are found. “eval” is used in line 748 and 821 of ai.py which is an unsafe solution against code injection attacks. Too general “Exceptions” are caught in line 568 and 638 of ai.py which may hide issues and raise unhelpful error messages. Global variables are used or modified in line 404, 441, 442 and 446 of ai.py both within and outside of functions, which hinders readability and the robustness of the code. Inconsistent use of protected class members (i.e. variables starting with _) both as the previously mentioned globals in ai.py and in line 612 of dict.py. Possibly using variable 'cleaning_func' before assignment in line 1784 of dict.py due to missing a “else” statement in line 1765. Unused argument in line 1607 in dict.py, line 100 in spatial_dict.py, and line 222 and 527 of stablelabel.py. Cell-var-from-loop (https://pylint.readthedocs.io/en/latest/user_guide/messages/warning/cell-var-from-loop.html) issue is found in line 1029 of ai.py that likely causes unintended behaviour.

Furthermore, the code base lacks architecture considerations. Too many functions are within too generically named files, such as ai.py and dict.py with more than 1000 and almost 2000 lines of code, respectively. One side effect is that the documentation is organized into functions but no higher abstraction level which makes navigation difficult. As noted in the readme, using objects would be a beneficial next step. There is a complete lack of unit tests raising concerns over the testability and extensibility of the code base. All of these give the impression that the code is not mature yet, the design is not well thought-through, and the project likely suffers from high technical debt which will hinder future development.

Some suggestions for modularization are listed here. Initialization functions for LLM providers can be organized into their own file, you may consider inheritance for these. Provider mapping, providers and provider model dictionaries can be replaced with dataclasses or enums for more development support. Configuration can have its own module, as well as the user interface, API calls to the providers, and many other facets of the current tool can be explicitly moved to their own module. Examples specific for this manuscript and for other illustration purposes can be moved to their own subfolder so that it is not bundled with the main package. Many references can be found online, for example this one: https://nsworldinfo.medium.com/functions-and-modules-in-python-simplifying-code-organization-and-reusability-7e4626d0ce9e

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The authors thoroughly addressed our comments. The manuscript has been substantially improved through clarifications, additional analyses, and expanded documentation. The authors have addressed several important concerns regarding data integrity, technical claims, and figure clarity.
However, two major points remain to be addressed to further strengthen the manuscript: the lack of an internal ablation study and the absence of quantitative benchmarking for the gene annotation module.
Major Comments
1. Regarding the authors’ response to Major comments 1 and 2 (Source of Performance Improvement; Impact of Dataset Differences on Results)
Thank you for clarifying the differences in research objectives and for making updates to the manuscript. To further strengthen the credibility of your results, we still recommend including some form of ablation study or component analysis within the framework of your current work. For example:
1) Evaluate the contribution of key components in AnnDictionary to compare the performance when using AnnDictionary with and without specific core modules.
Even without direct comparison to prior work, such an internal analysis would better demonstrate which aspects of your system contribute to the observed results, and provide clearer insight into the source of performance improvements.

2. Regarding the authors’ response to Major comments 3 (Comparison of Gene Annotation Methods)
Thank you for the additional effort and for providing documentation and example code for the gene annotation module. However, from the perspective of evaluating AnnDictionary as a scientific tool, it is important to rigorously assess the performance of each functional component, so that researchers can properly evaluate its reliability and usability in real applications.

While the examples help illustrate how the function works, they do not fundamentally answer the question of whether this module is reliable. A few anecdotal examples are insufficient to demonstrate performance. Instead, some form of quantitative evaluation or benchmarking, even if limited in scope, is necessary to support the validity of this feature.

We encourage the authors to include a minimal benchmark—e.g., comparing a small set of gene sets annotated by

AnnDictionary to annotations from established tools—to provide at least preliminary evidence for the effectiveness of the method.

3. Regarding the authors' response to Major comments 4
This point has been satisfactorily addressed. We appreciate the authors' thorough effort to assess and document the risk of data leakage and potential bias.

4. Regarding the authors' response to Major comments 5
This point has been satisfactorily addressed. This point has been substantially addressed. We appreciate the authors' detailed response.

5. Regarding the authors' response to Major comments 6
This point has been satisfactorily addressed. This point has been substantially addressed. We appreciate the authors' detailed response.
6. Regarding the authors' response to Major comments 7
This point has been satisfactorily addressed.

Minor Comments
All minor figure legend issues have been addressed. No further concerns.

Remarks on code availability
All issues have been addressed.

(Remarks on code availability)

Reviewer #2

(Remarks to the Author)
The authors have addressed my comments.

(Remarks on code availability)

Reviewer #3

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

(Remarks on code availability)

Reviewer #4

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

(Remarks on code availability)

Reviewer #5

(Remarks to the Author)
Thank you for your answers and the modifications applied in the manuscript. These are sufficiently addressing my comments. I am really impressed by the complete reorganization and updates of the code base which now looks appropriate and installs without issues.

(Remarks on code availability)
The code base looks good. There is an appropriate structure, a good amount of unit tests, and sufficient documentation. The notebooks are providing examples of use cases.
I think it would be a good idea to set the version number to 1.0.0 indicating that the code is stable.

Version 2:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The authors have addressed my concerns. Congratulations for a nice work.

(Remarks on code availability)

Reviewer #3

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

(Remarks on code availability)
I have no questions about the code as it has been checked in the last version.

Reviewer #1 (Remarks to the Author):

This work presents a tool to benchmark cell type annotation. The topic is certainly very important and useful for the biomedical research community. However, the writing of the work is too abstract to judge. There are no details of methodology. The annotation data, as the authors wrote in the papers, are not released. So with mostly just accuracy reports, it is hard for me to judge the quality of this work. More detailed comments are below.

We apologize for this oversight. Detailed methodological description was previously consolidated in-line with the results. We have now moved this text to its own dedicated methods section.

Major Comments

1. Source of Performance Improvement:

(1) The authors claim that on the same LLM basis, AnnDictionary increases annotation consistency from 58% to 78%. Please clearly state whether the parameters of the LLM used in the experiments are identical or if there are any differences.

We thank the reviewer for their question. As the reviewer has correctly identified, the parameters used in LLM interactions can have impact on the results. The parameters used in our LLM interactions were previously viewable in the source code of our package, but we have now added them explicitly to the main text. The main text now reads:

> *LLM hyperparameters*
> For all LLM queries, we set the temperature to 0, which makes the LLM behave more deterministically. For all other hyperparameters, the defaults were used.

Additionally, we looked into how the LLM parameters differed between our study and prior work. In "Assessing GPT-4 for cell type annotation in single-cell RNA-seq analysis" by Hou and Ji, a temperature of 0.3 was used for LLM interactions when the LLM was accessed through the API.[1] Other LLM interactions in Hou and Ji were performed through the ChatGPT web interface, and it is thus unclear what the LLM hyperparameters and model version were.[1] For reasons we discuss in detail in response to the next point, we no longer find it suitable to include the hyperparameters of this other study in our manuscript.

(2) Please provide a detailed ablation study to analyze the sources of the performance gain. Clarify which steps or optimization measures contribute to the observed 20% improvement.

We thank the reviewer for their recommendation to provide a detailed study to analyze the differences in procedures and optimizations between the present study and prior work.

Upon detailed review of the methods used in prior work, namely "Assessing GPT-4 for cell type annotation in single-cell RNA-seq analysis" by Hou and Ji, we found that the metrics presented were based on curated gene lists derived from either literature searches or previously annotated groups of cells. Therefore, prior work by Hou and Ji provided important evidence that LLMs could identify cell types via marker genes, but did not assess the ability of LLMs to

annotate cell type in the face of the full complexity of gene lists calculated from unannotated clusters derived de novo from unsupervised analysis. Marker gene lists derived de novo can be more complex due to a variety of factors, including the presence of artifactual genes and cell types split among multiple clusters.[2,3] We have therefore revised our manuscript to reflect that we have performed the first benchmarking of LLMs at *de novo* cell type annotation. Because we were assessing a fundamentally different task than that presented in Hou and Ji, it would be inappropriate to directly compare the performances in the context of performance gains or losses. We have therefore removed from our manuscript language that presents the observed performance differences as a performance gain, and updated our manuscript to reflect how the goals of the present study differ from previous investigation by Hou and Ji.

The abstract now reads:

> We developed an open-source package called AnnDictionary (https://github.com/ggit12/anndictionary/) to facilitate the parallel, independent analysis of multiple anndata. AnnDictionary is built on top of LangChain and AnnData and supports all common large language model (LLM) providers. AnnDictionary only requires 1 line of code to configure or switch the LLM backend and it contains numerous multithreading optimizations to support the analysis of many anndata and large anndata. We used AnnDictionary to perform the first benchmarking study of all major LLMs at de novo cell-type annotation in Tabula Sapiens. LLMs varied greatly in absolute agreement with manual annotation based on model size. Inter-LLM agreement also varied with model size. We find that LLM annotation of most major cell types to be more than 80-90% accurate, and will maintain a leaderboard of LLM cell type annotation at https://singlecellgpt.com/celltype-annotation-leaderboard.

The introduction now reads:

> Previous work indicates that LLMs can reliably identify cell type from curated lists of marker genes—those identified via literature or calculated from previously identified cells of known type—but there has not yet been an assessment of LLMs at de novo cell type annotation, meaning annotation of gene lists derived directly from unsupervised clustering.[1] These gene lists crucially differ from curated gene lists, because they contain unknown signal and noise that may affect the annotation process.[2,3] De novo annotation is therefore a potentially more challenging task. The goal of this investigation is to assess the effectiveness of LLMs in this context. So, we used AnnDictionary to benchmark the de novo cell type annotation ability of the major commercially available LLMs (i.e. all models from OpenAI, Anthropic, Google, Meta, and all additional available text generation models on Amazon Bedrock, including Mistral, Titan, and Cohere).

The Discussion now reads:

> Previous work has assessed GPT-4's ability to annotate curated lists of genes, including lists derived from Tabula Sapiens v1[1] We build on this previous work by assessing LLMs' abilities to annotate the full complexity of gene lists derived from unsupervised clustering. In the present study, we used Tabula Sapiens v2—which contains more than double the number of cells compared to Tabula Sapiens v1. On this larger dataset, we find the annotation of major cell

types to be more than 80-90% accurate, making LLM-based annotation a viable option for first-pass cell type annotation. The flexibility of LLM-generated annotations solves a major problem of automated annotation procedures, which have historically lacked flexibility due to the need to use a reference set of annotations.[3] Furthermore, the LLM-based approach is reference-free and so does not require additional datasets, which could otherwise increase computational burden.

2. Impact of Dataset Differences on Results:

It is mentioned in the text that the 58% figure comes from the first edition of the 2022 Tabula Sapiens dataset, whereas the 78% figure is based on Tabula Sapiens v2. These two datasets differ significantly in terms of scale and diversity. To ensure fairness in the results, please conduct performance comparison experiments using the same dataset from Tabula Sapiens or Tabula Sapiens v2.

We thank you for your suggestion. As previously discussed, we have removed from our manuscript language that compares the evaluation results directly or presents the evaluation results of the present study as performance improvements. We have now further clarified how the objectives of the present study differ from those of prior work because the present study performed de novo annotation, whereas prior work did not. We have revised the discussion to describe how our present analysis builds on prior work, and clarify how the scope of each dataset differs. The discussion now reads:

> Previous work has assessed GPT-4's ability to annotate curated lists of genes, including lists derived from Tabula Sapiens v1[1] We build on this previous work by assessing LLMs' abilities to annotate the full complexity of gene lists derived from unsupervised clustering. In the present study, we used Tabula Sapiens v2—which contains more than double the number of cells compared to Tabula Sapiens v1. On this larger dataset, we find the annotation of major cell types to be more than 80-90% accurate, making LLM-based annotation a viable option for first-pass cell type annotation.

3. Comparison of Gene Annotation Methods:

Please supplement the manuscript with quantitative comparisons of AnnDictionary's 'gene annotation' methods against other similar tools.

We thank you for your critique. While the comprehensive benchmarking of gene annotation methods is certainly of importance and interest, it is beyond the scope of the present manuscript, which is to benchmark LLMs at de novo cell type annotation. Furthermore, detailed benchmarking of LLMs at this task has been previously studied.[4] We have clarified this point in the discussion to make it clear that we have not performed such benchmarking:

> In addition to comprehensive cell type annotation benchmarking, we also include biological inference functions (e.g. gene set annotation with biological processes). Previously published work has benchmarked LLM performance at this task.[4] It is convenient to consolidate these tasks into one package. A major limitation of current gene set enrichment analyses is their dependence on the sizes of the gene sets in the query database.[5] The use of LLMs—which do not rely on static lists of genes—to annotate pathways is therefore a promising solution to this issue.

However, we do agree that some demonstration of this feature would be useful. We have added example usage to the function's documentation, see (https://ggit12.github.io/anndictionary/api/annotate/cells/generated/anndict.annotate.ai_annotate_biological_process.html#anndict.annotate.ai_annotate_biological_process). In addition, we have now written a tutorial notebook to illustrate how gene set annotation would be performed, and include an example of annotating three known gene sets from the Gene Ontology Biological Process database. We have added this notebook to both the GitHub repository for AnnDictionary (https://github.com/ggit12/anndictionary/blob/main/nbs/tutorials/ipynb/biological_process_annotation_with_an_LLM.ipynb), and the AnnDictionary documentation (https://ggit12.github.io/anndictionary/tutorials/annotate/biological_process_annotation_with_an_LLM.html).

We have also included this example in the main text. We have written a description of the gene set annotation process in the Methods:

> *Biological process annotation*
> We used gseapy v1.1.8 to access the 2023 release of the Gene Ontology Biological Process database. Then, to make a brief example, we retrieved the gene lists associated with the first three terms in the database, and used AnnDictionary's ai_biological_process function to label the gene sets with claude-3-5-sonnet-20240620.

We have additionally edited the results to include this example. The results now read:

> *Biological process annotation*
> To demonstrate the use of LLMs to annotate gene lists with the biological process that they represent, we annotated three gene lists of known processes from the Gene Ontology Biological Process database. In these three cases, the LLM annotations, while slightly broader, generally agreed with the existing labels, **Supplementary Table 7**.

We have included these results as **Supplementary Table 7:**

Supplementary Table 7. Example biological process annotations of known gene lists using an LLM (Claude 3.5 Sonnet).

GO Biological Process Term	Genes	LLM annotation
'De Novo' AMP Biosynthetic Process (GO:0044208)	ATIC, PAICS, PFAS, ADSS1, ADSS2, GART	Purine biosynthesis pathway
'De Novo' Post-Translational Protein Folding (GO:0051084)	SDF2L1, HSPA9, CCT2, HSPA6, ST13, ENTPD5, HSPA1L, HSPA5, PTGES3, HSPA8, HSPA7, DNAJB13, HSPA2, DNAJB14, HSPE1,	Protein folding and chaperone-mediated quality control.

	DNAJC18, GAK, DNAJC7, DNAJB12, HSPA1A, ST13P5, HSPA1B, ERO1A, SELENOF, HSPA14, HSPA13, DNAJB1, CHCHD4, DNAJB5, DNAJB4, SDF2, UGGT1	
2-Oxoglutarate Metabolic Process (GO:0006103)	IDH1, PHYH, GOT2, MRPS36, GOT1, IDH2, ADHFE1, GPT2, TAT, DLST, OGDHL, L2HGDH, D2HGDH, OGDH	Mitochondrial tricarboxylic acid (TCA) cycle and related metabolic pathways.

4. Data Leakage and Bias Risk:

In the Tabula Sapiens v2 dataset used for model evaluation, more than half of the manual annotation information has not been published. Such unpublished information may have appeared in other published literature. If all or part of this information was included in the LLM training data, it could lead to data leakage, resulting in biased evaluation results.

To understand the risk of data leakage, we reviewed the release dates of all manuscripts that used Tabula Sapiens v2, and looked at these in the context of the model training knowledge cutoff dates. Prior to the present manuscript, the only work to use Tabula Sapiens v2 data was posted on bioArxiv on November 29th, 2023.

A few canonical memory vs. naïve B cell markers (IGHM, IGHD, YBX3, TNFRSF13B, CD27, ATXN1) were the only differentially expressed genes from Tabula Sapiens v2 that were mentioned, see supplemental figure s2 from this preprint. Given that these are all previously published B cell markers,[6-8] their mention in Rosen et al. does not provide noteworthy additional information in LLM model training compared to what already existed in the literature. For completeness, we assessed the presence of these genes in the full list of all marker genes used for annotation in this study. Of 676 clusters annotated, YBX3 was used in the annotation of 10 clusters, and none of the other 5 genes were used.

Any other preprints or publications that used Tabula Sapiens v2 were released after knowledge cutoff dates of models used in the present manuscript.

We now include the knowledge cutoff dates in **Table 2**:

Table 2. List of LLMs studied.

Company	Provider	Endpoint	Knowledge Cutoff Date
Meta	Bedrock	meta.llama3-1-8b-instruct-v1:0	December 2023[9]
Meta	Bedrock	meta.llama3-1-70b-instruct-v1:0	December 2023[9]
Meta	Bedrock	meta.llama3-1-405b-instruct-v1:0	December 2023[9]
Amazon	Bedrock	amazon.titan-text-express-v1	Not defined[10]
Amazon	Bedrock	amazon.titan-text-lite-v1	Not defined[10]
Cohere	Bedrock	cohere.command-r-plus-v1:0	February 2023[11]*

Company	Provider	Endpoint	Knowledge Cutoff Date
Mistral	Bedrock	mistral.mistral-large-2407-v1:0	Released July 24th, 2024[12]
Google	Google	gemini-1.5-pro-002	September 2024[13]
Google	Google	gemini-1.5-flash-002	September 2024[13]
OpenAI	OpenAI	gpt-4-0613	November 30th, 2023[14]
OpenAI	OpenAI	gpt-4o-2024-08-06	September 30th, 2023[14]
OpenAI	OpenAI	gpt-4o-mini-2024-07-18	September 30th, 2023[14]
Anthropic	Anthropic	claude-3-5-sonnet-20240620	April 2024[15]
Anthropic	Anthropic	claude-3-opus-20240229	August 2023[15]
Anthropic	Anthropic	claude-3-haiku-20240307	August 2023[15]
*not clear which model version is accessed on Bedrock			

We thus conclude that there is minimal risk of data leakage and bias.

Furthermore, we have amended the manuscript to include a section about the risk of data leakage, and added a step in our snakemake pipeline to record the usage of the 6 genes mentioned above (https://github.com/ggit12/benchmark_llms/blob/main/benchmark_pipeline/scripts/14_extract_DEGs.py).

In the methods section, we now state:

> *Risk of data leakage biasing the results*
> To understand the risk of data leakage, we reviewed the release dates of all manuscripts that used Tabula Sapiens v2, and looked at these in the context of the model training knowledge cutoff dates. Prior to the present manuscript (first posted in preprint form on bioaRxiv on October 13th, 2024), the only work to use Tabula Sapiens v2 data was posted on bioaRxiv on November 29th, 2023. Six canonical memory vs. naïve B cell markers (IGHM, IGHD, YBX3, TNFRSF13B, CD27, ATXN1) were the only differentially expressed genes from Tabula Sapiens v2 that were mentioned. Given that these are all previously published B cell markers,[6-8] their mention in Rosen et al. does not provide noteworthy additional information in LLM model training compared to what already existed in the literature. For completeness, we assessed the frequency of these genes in the full list of all marker genes used for annotation in this study. Of 676 clusters annotated, YBX3 was used in the annotation of 10 clusters, and none of the other 5 genes were used. Any other preprints or publications that used Tabula Sapiens v2 were released after knowledge cutoff dates of models used in the present manuscript, **Table 2**. There were 3 exceptions, which involved models that were in the lower range of performance. Specifically, Mistral Large 24.07 was released July 24th, 2024, but Mistral does not specify a knowledge cutoff date. The two Titan models also do not have a knowledge cutoff date, but these models were omitted from detailed benchmarking in this study as previously discussed. We thus conclude that there is minimal risk of data leakage biasing the annotation results.

The Discussion now reads:

> There are several potential limitations due to bias in the LLMs used in this study. We attempted

to fully characterize the risk of data leakage influencing benchmarking by denoting all information of the previous preprint that used Tabula Sapiens v2, cataloguing the use of relevant information in the present analysis, and making note of all this information in the context of the knowledge cut-offs of all LLMs used in the study.

5. Performance Issues on Unreleased Annotations:
The observed performance improvement may be misunderstood as a comprehensive enhancement, but the evaluation results focus solely on the portion of manually annotated markers that is publicly available. For the unreleased portion, model performance may decline. Please revise the relevant descriptions accordingly.
Thank you for your suggestion. We agree that the present results should not be taken as a performance improvement over previous results, because we assessed a fundamentally different annotation task. We have therefore removed language that presents our results as a performance improvement.

Prior work by Hou and Ji used manually annotated markers that were publicly available to assess LLM annotation of cell type based on these markers. To be clear about the present analysis, our evaluation results did not use only the portion of manually annotated markers that were publicly available. In fact, we used the entirety of the Tabula Sapiens dataset (v1 and v2, totaling over 1 million cells), and used differentially expressed genes derived de novo from unsupervised clustering of the data. Therefore, the gene lists were sourced from the entire transcriptome and all available data, more than half of which was not publicly released at the time of initial benchmarking. The dataset has since been released in its entirety, but, critically, the knowledge cutoffs of the majority of models used in the present study—including all the top-performing models—preceded the public release of Tabula Sapiens v2, see **Table 2**.

To directly address the concern that the presence of cells from Tabula Sapiens v1 may be driving LLM performance, we re-ran the benchmarking pipeline using only the portion of Tabula Sapiens v2 that was not collected as part of Tabula Sapiens v1. The performance on this portion of the dataset is marginally better than on the full dataset, which is perhaps to be expected because the size of the dataset is smaller. We have included these results as **Supplementary Table 2**. Because the Tabula Sapiens v2 dataset as a whole is of primary interest for downstream biological tasks, we still focus our investigation on the LLM annotations of the entire Tabula Sapiens v2 atlas, and use the agreement metrics calculated thereon because they are generally lower and therefore more conservative. We have updated the Results, Discussion, and Methods to address these points.

We have now included **Supplementary Table 2**:

Supplementary Table 2. LLM performance on only the portion of Tabula Sapiens v2 that not collected as part of Tabula Sapiens v1.

	Binary (%)		Perfect Match (%)		Extract String Match (%)		Kappa	
	Cells	By Cell Type	Cells	By Cell Type	Cells	By Cell Type	With Manual	Average With LLMs
Claude 3 Haiku	75.8 ± 2.7	66.2 ± 3.5	65 ± 7	51 ± 11	65 ± 7	51 ± 11	0.62 ± 0.07	0.67 ± 0.06
Claude 3 Opus	82.7 ± 2.1	68.2 ± 2.6	76 ± 5	57 ± 9	76 ± 5	57 ± 9	0.73 ± 0.04	0.70 ± 0.04
Claude 3.5 Sonnet	84.5 ± 0.8	68.0 ± 1.4	76.1 ± 2.7	55 ± 7	76.0 ± 2.9	54 ± 7	0.734 ± 0.025	0.69 ± 0.05
Command R Plus	76.5 ± 3.2	61.4 ± 2.5	66 ± 7	48 ± 12	66 ± 7	48 ± 12	0.63 ± 0.07	0.62 ± 0.06
GPT-4	76.6 ± 0.8	64.7 ± 2.5	66 ± 7	47 ± 11	66 ± 7	47 ± 11	0.63 ± 0.07	0.67 ± 0.09
GPT-4o	80.7 ± 2.3	69.4 ± 2.7	71 ± 6	55 ± 10	71 ± 6	55 ± 11	0.68 ± 0.06	0.71 ± 0.04
GPT-4o mini	73.6 ± 1.8	65.1 ± 1.5	65 ± 7	49 ± 10	65 ± 7	49 ± 10	0.62 ± 0.06	0.70 ± 0.05
Gemini 1.5 Flash	66 ± 5	59.1 ± 3.3	50 ± 7	43 ± 10	50 ± 7	42 ± 11	0.45 ± 0.06	0.57 ± 0.05
Gemini 1.5 Pro	78.1 ± 2.5	64.8 ± 2.3	68 ± 6	50 ± 9	68 ± 6	50 ± 9	0.65 ± 0.06	0.65 ± 0.05
Llama 3.1 405B Instruct	81 ± 4	66 ± 4	68 ± 6	50 ± 11	68 ± 6	50 ± 11	0.65 ± 0.06	0.65 ± 0.05
Llama 3.1 70B Instruct	69 ± 4	60.6 ± 2.9	61 ± 4	47 ± 10	61 ± 4	47 ± 10	0.576 ± 0.033	0.64 ± 0.04
Llama 3.1 8B Instruct	54 ± 9	53 ± 6	45 ± 8	42 ± 11	45 ± 8	42 ± 11	0.41 ± 0.06	0.50 ± 0.07
Mistral Large	75.4 ± 2.6	65.0 ± 3.4	63 ± 5	51 ± 10	63 ± 5	50 ± 10	0.59 ± 0.04	0.68 ± 0.04
Plurality Vote	78.6 ± 1.8	68.0 ± 2.4	72 ± 5	55 ± 9	72 ± 5	55 ± 9	0.70 ± 0.04	0.76 ± 0.04

The results now read:

> *Annotation performance was not driven by the presence of cells from Tabula Sapiens v1*
> More than half of Tabula Sapiens v2 had not been publicly released by the knowledge cutoffs of the majority of models benchmarked in this study, including all of the top-performing models. However, cells from Tabula Sapiens v1—which was released before the knowledge cutoffs—account for a substantial portion of Tabula Sapiens v2. To determine if the presence of Tabula Sapiens v1 cells was driving the annotation results in this investigation, we re-ran the LLM benchmarking pipeline after removing cells that were part of Tabula Sapiens v1. In this control experiment, the performance of the LLMs is marginally better, **Supplementary Table 2**. This is not surprising because this test dataset was smaller. Because the Tabula Sapiens v2 dataset as a

whole is of primary interest for downstream biological tasks, we still focus our investigation on the LLM annotations of the entire Tabula Sapiens v2 atlas, and use the agreement metrics calculated thereon because they are generally lower and therefore more conservative.

The Discussion now reads:

> Furthermore, we ensured that the observed performances are not due to overlap with the previously published Tabula Sapiens v1 by replicating performance in only the portion of Tabula Sapiens v2 that was not released in Tabula Sapiens v1. Because the Tabula Sapiens v2 dataset as a whole is of primary interest for downstream biological tasks, we still focus our investigation on the LLM annotations of the entire Tabula Sapiens v2 atlas, and use the agreement

The Methods now read:

> *Validation on Tabula Sapiens v2*
> Using the Tabula Sapiens v2 dataset obtained as described above in *Data access*, we took only donors that were not included in Tabula Sapiens v1, that is any donor after but not including TSP15. We then re-ran the entire LLM benchmarking pipeline on this subset of the data. We excluded pancreas from this analysis due to low cell abundance. The source code for this version of the pipeline is available in the benchmark_llms GitHub repository under the branch metrics_only.

6. Comparison with Existing Tools:
A study published in 2024, "Reference-free and cost-effective automated cell type annotation with GPT-4 in single-cell RNA-seq analysis", also supports Python and multiple LLMs. Relevant documentation is available in the OmicVerse Tutorials (https://omicverse.readthedocs.io/en/stable/Tutorials-single/t_gptanno/).
We thank you for bringing the OmicVerse package to our attention. We agree that the package supports multiple LLMs. In order to adequately revise our claims, we performed a detailed review of OmicVerse's annotation functionality. Ultimately, AnnDictionary is the first package to natively support multiple LLM providers, and contains substantial technical advances. Below, we discuss these points in detail.

We note that "Reference-free and cost-effective automated cell type annotation with GPT-4 in single-cell RNA-seq analysis" by Hou and Ji is the same study as previously discussed, "Assessing GPT-4 for cell type annotation in single-cell RNA-seq analysis" by Hou and Ji, but titled differently on its bioRxiv preprint. The OmicVerse package, presented by Zeng et al. in their 2024 manuscript titled "OmicVerse: a framework for bridging and deepening insights across bulk and single-cell sequencing" described its LLM annotation features as a direct port of GPTCelltype from R to Python. We interpret the reviewer's comment to refer to the multi-LLM support implemented by OmicVerse.

A true comparison between OmicVerse and AnnDictionary must be made by direct inspection of the source code. For this, we turn to the core annotation functions of OmicVerse (https://github.com/Starlitnightly/omicverse/blob/master/omicverse/single/_gptcelltype.py).

Upon review of the source code, it is clear that OmicVerse only supports LLMs to the extent that they happen to share the same API access methods and call parameters as OpenAI, see lines 18, 22-28, and 35-38, where the URL of the API endpoint is directly used through an OpenAI client. Crucially, this method relies on the use of identical query parameter formatting between the user's desired model and the format required by the OpenAI client. The use of OpenAI's client under-the-hood severely limits extensibility to support other LLMs. This happens to work for the two other LLM providers supported by OmicVerse.

In contrast, AnnDictionary provides substantial engineering around LLM querying. LLM management was a major effort achieved by AnnDictionary. Now housed in the sub-package anndict.llm, this functionality uses LangChain to support a multitude of providers with varying requirements in terms of query and input format. For example, the current implementation in OmicVerse would have to change substantially to extend coverage to commonly used providers like Google, Anthropic, or Amazon that use different query parameters and request formats. To exhaustively elucidate the technical shortcomings of the OmicVerse implementation, it is poor practice to incorporate LLM initialization, gene list extraction from a dataframe, and LLM query within the body of a single function, as this hinders both debugging and future development. Furthermore, OmicVerse does not implement retry mechanisms, rate limiters, customizable response parsing, or failure handling, all of which are required for a user-friendly package that can use LLMs to annotate at atlas scale. One of the major efforts of AnnDictionary was building in all of these technical features. We also note a lack of prompt optimization in OmicVerse, for example, line 50 of the source linked above, which contains repeated prompt elements:

> message = f'Identify cell types of tissuename cells in speciename using the following markers separately for each row. Only provide the cell type name. **Do not show numbers before the name. Do not show numbers before the name.** Some can be a mixture of multiple cell types.\n' + '\n'.join([f'k: v' for k, v in input.items()])

And finally, there is low-legibility code, specifically in lines 29, 30, and 36, where the API key variable is called "QWEN_API_KEY" in the source code, regardless of which provider's API key is actually bound.

Taking the full technical details into consideration, we have revised the claims in our manuscript. Given that we built on top of LangChain to achieve provider-agnostic LLM support across the major providers (OpenAI, Anthropic, Amazon Bedrock, and Google AI), AnnDictionary is the first package to natively support multiple LLMs. We have revised the manuscript to reflect these points. The results now read:

> ***AnnDictionary consolidates common LLM integrations under one roof***
> As previously mentioned, several LLM-based automations, including gene set annotation and data label management, have been developed and tested, but there does not exist, to the authors' knowledge, a centralized implementation of them in Python built on top of AnnData, the predominant data structure used in Pythonic scRNA-seq analysis. Thus, we implemented a variety of LLM-based tools.

AnnDictionary is the first package in this space to natively support multiple LLM providers, and contains substantial technical advances over previous work,[1,16] including few-shot prompting, retry mechanisms, rate limiters, customizable response parsing, and failure handling. All of these features contribute to a user-friendly experience when annotating datasets.

7. Centralized Implementation in Scanpy:
On page 5, the manuscript states: "As previously mentioned, several LLM-based automations have been developed and tested, but there does not exist, to the authors' knowledge, a centralized implementation of them in python built on top of Scanpy, the predominant scRNA-seq analysis python package. Thus, we implemented a variety of annotation tools."
However, OmicVerse already supports Python and integration with Scanpy.

Thank you for highlighting this section. OmicVerse implements cell type annotation with LLMs. We implemented several additional LLM-based features: multiple types of gene set annotation, including for cell type and biological process, as well as various data cleaning functions that use LLMs to process natural language data labels, easing manipulation of categorical labels from different datasets (as one example). What we mean to say is that it would seem useful to have this suite of functions available when working with AnnData, and that we have centralized their implementation. We have thus revised the manuscript to clarify this statement. We have also revised the statement to reflect that we built additional data structures on top of AnnData rather than Scanpy, because this more accurately reflects our engineering efforts. The results now read:

> ***AnnDictionary consolidates common LLM integrations under one roof***
> As previously mentioned, several LLM-based automations, including gene set annotation and data label management, have been developed and tested, but there does not exist, to the authors' knowledge, a centralized implementation of them in Python built on top of AnnData, the predominant data structure used in Pythonic scRNA-seq analysis. Thus, we implemented a variety of LLM-based tools.

And we have added a subsection to further elaborate:

> *Automated label management*
> We implemented several functions to assist with data label management in AnnData using LLMs. Use cases include resolving syntactic differences in labels used across different studies. Some functions are built to process category labels in a single column by cleaning them, merging them, or generating multi-column label hierarchies (i.e. from cell subtype all the way to compartment). Other functions are designed to handle common situations when dealing with datasets from multiple sources or annotations from multiple methods, such as differing notation for common cell types. Furthermore, we provide ways to assess label agreement (all using LLMs to manage label comparison).

Minor Comments

Insufficient Figure Legends:

Several figures lack adequate legends. For example:
(1) The numerical meaning in Figures 3B and 3D.
We thank the reviewer for pointing this out. The numerical values represent the number of cells. The revised Figures 3B and 3D now include the label "N Cells" on the scale bar.

(2) The specific meaning of different colors in Figure 4.
As stated in the figure legend, "Red, yellow, and green represent the bottom, middle, and top tertile of cell type by population size, respectively."

(3) The specific meaning of the [0,1] values in Figure 5A.
Thank you for drawing attention to Figure 5A. The numerical values represent the proportion of cells in each manual annotation category that are in each LLM annotation. The revised figure legend now states:

> The scale bar represents the proportion of cells from each category of manual annotation that are in each category of LLM annotation. Thus, each row sums to 1.

It is recommended to add or improve these legends to help readers better understand the information presented in the figures.
We thank the reviewer for this helpful recommendation. All relevant figure legends have been expanded to clarify scale bars, and the normalization of numerical values (see our specific responses above).

Reviewer #1 (Remarks on code availability):

We downloaded and implemented the code. The setup works fine. However, there is no data to test the program. Thus, we are not able to validate the results of the submission.
We now use the sample dataset included with Scanpy for tutorials. This ensures that users can successfully run basic tutorials as long as Scanpy installation is possible.

Additionally, the full Tabula Sapiens v2 dataset is available on CellxGene here (https://cellxgene.cziscience.com/collections/e5f58829-1a66-40b5-a624-9046778e74f5). The Tabula Sapiens v1 release can be downloaded from Figshare here (https://figshare.com/articles/dataset/Tabula_Sapiens_release_1_0/14267219).

Finally, we have provided an Anthropic API key for testing the LLM features.

Parallel analysis of large-scale data is generally conducted on HPC servers. It is recommended that a readily usable Singularity SIF image file be provided.

We thank the reviewer for the suggestion. We have now provided a Singularity image file to use for the benchmark_llms pipeline and added download instructions to the repository's README. This Singularity image contains an installation of AnnDictionary as well.

Reviewer #2 (Remarks to the Author):

In the manuscript, Crowley et, al. developed the software AnnDictionary, which is an open-source Python package for the cell-type annotation of single-cell RNA-seq data using common LLMs such as OpenAI's ChatGPT and Google's Gemini. Specifically, AnnDictionary can process multiple datasets at a time, annotate cell types and gene sets, and support multiple LLM models. They have also benchmarked different LLM models' ability to annotate cell types from the resolution of cells, cell types, and tissue-cell types, which has provided valuable information for LLM model selection. Moreover, they have also explored the inconsistency between manual annotation and LLM annotation in basal and stroma cells. They found that LLMs can identify the major cell types in the cluster population. They have also found that in the majority of cells, LLMs showed great consistency with each other and agreed well with manual annotation. The study provides a tool AnnDictionary, for practically cell type annotation using LLMs and gives suggestions on model selection by represents as a comprehensive benchmark study. It also provides things that need to be taken care of when using LLMs for cell type annotations. AnnDictionary may have limited novelty when there is GPTCelltype that uses GPT-4 for cell type annotation. The benchmark part can still be very helpful in practical cell type annotation using LLMs though it may have limitations. Below are my detailed comments:

We thank the reviewer for their comments that have helped improve the manuscript. We have revised the manuscript to better describe the novelty offered by the present study. While the publication that presented GPTCelltype assessed its ability to annotate curated lists of marker genes, we add novelty by presenting the first benchmarking result of LLMs on the full complexity of marker gene lists derived from unsupervised clustering. Furthermore, we have elaborated on the numerous technical advances over GPTCelltype to yield a user-friendly package that scales to annotate thousands of clusters.

Major comments:

1. The manuscript misses the methods part, which is very important in the process of peer review.

We apologize for this oversight. Detailed methodological description was previously consolidated in-line with the results. We have now provided further detail and moved this text to its own dedicated methods section which can be found after the Conclusion section. We have also broken the LLM benchmarking pipeline into a separate snakemake pipeline to further increase the interpretability of the methodology, validation of results, and reusability of the code. The repository is available here (https://github.com/ggit12/benchmark_llms) and includes the code for data preprocessing, annotation, and downstream interpretation.

2. I suggest benchmarking the performance from the level of samples and the level of atlas. I suspect the evaluation is on the atlas level of Tabula Sapiens v2. Sample-level evaluation should also be performed in at least three different tissues. Sample-level benchmarking is more important since this is the practical performance of users.

Thank you for highlighting this point. To clarify, in our current study, initial annotations were indeed generated separately for each tissue. This per-tissue annotation approach directly reflects the annotation strategy employed by Tabula Sapiens v2 when annotating samples,[17] and thus represents the annotation strategy that we may expect from an end user. Additionally, to explicitly address your request, we have now included supplementary analyses calculating agreement metrics at the tissue level. We include these results in **Supplementary Figure 3**. When viewing these results, we noticed that the lowest-agreement tissues coincided with those that had high abundance of low-agreement cell types. Thus, we still focus our analysis on understanding annotation differences at the cell type level.

The results now read:

> We also considered the LLM annotation performance within each tissue, **Supplementary Figure 3**. The tissues with the lowest average agreement between manual and LLM annotations were ear, muscle, and ovary. Viewing the tissue results in the context of the tissue-cell type level metrics, we can see that low tissue-level performance was driven by having a higher relative abundance of cell types that had low agreement with manual annotation in general. Specifically, the ear had 37% stromal cells, the muscle had 50% mesenchymal stem cells, and the ovary 72% stromal cells of ovary. So, we focus on understanding the annotations at the cell type level.

And we have included **Supplementary Figure 3**:

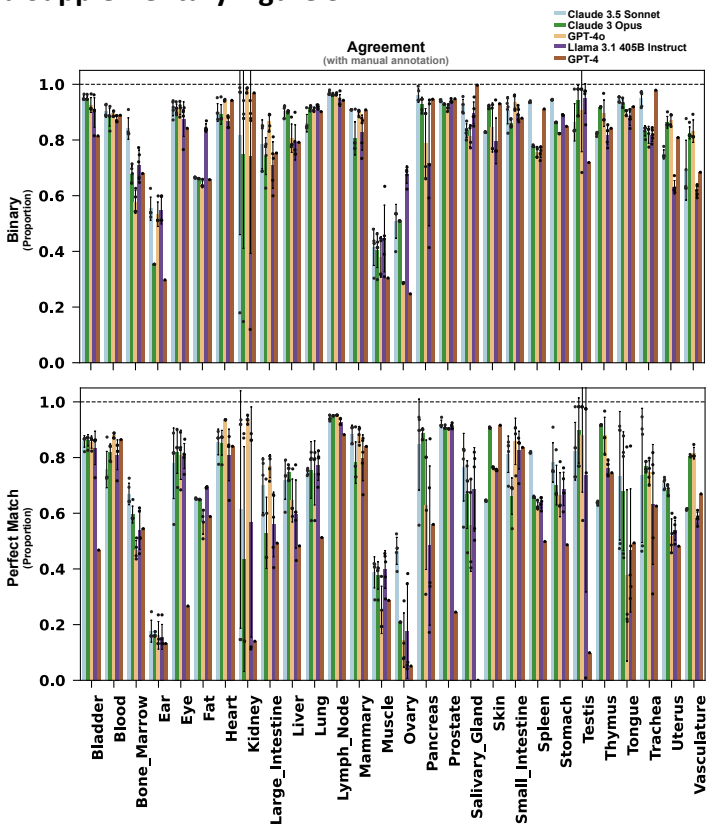


Figure S3. Agreement with manual annotation of top performing LLMs for the each tissue in Tabula Sapiens v2. As in **Figure 2**, agreement was assessed at two levels: binary (yes/no, top) and perfect match (bottom).

In response to the reviewer's minor comment (3), we have now elaborated on the different resolutions at which the agreement metrics were calculated, and added a description of the tissue-level resolution. The methods now include:

> **Resolutions.** We calculated the rate of agreement of each model with manual annotations at four resolutions: 1) cells, 2) cell types, **Table 2**, 3) tissue-cell types, **Figure S1**, and tissues, **Figure S3**. The first resolution, cells, corresponds to treating each cell individually, and seeing the rate of agreement across all cells. We consider this resolution to be the most representative of overall annotation performance, because it directly measures the overall proportion of cells that are correctly annotated across the dataset. The second resolution, cell types, is calculated by taking the mean of agreement of cells when grouped by the manual annotation column. The average of this metric across all cell types is presented in **Table 2**, and in specific cell types in Figures 3A and 3B. This metric is useful for uncovering annotation performance of the model at each cell type, but, unlike the cell resolution, does not take into account differential cell type abundance. The third resolution, tissue-cell type, is calculated as the mean agreement of within a single tissue-cell type. This metric was considered because cell types are known to have tissue-specific expression programs, and so it was possible that the resulting differentially expressed genes, and thus annotation performance, varied at the tissue-cell type level. Finally, agreement was calculated at the tissue level by taking the mean agreement of all cells in a single tissue.

Also, I would like to know the performance of individual LLMs at different levels of cell type annotation (e.g. T cells / CD4 T, CD8T).

The initial labelling of cell type is currently a major bottleneck in the single cell processing pipeline and is the problem that we aim to address in the present manuscript. While there are a few cell subtypes for which there is general agreement, generally, there is still much debate in the literature about how to identify and define cell subtypes in general. Thus, we don't think the literature has matured enough to call for the automation of this task, of which the labels produced and their downstream utility would be highly variable and depend on the practitioner and research question. Therefore, our goal was to provide in-depth benchmarking and study of an automated method to label broad cell types, and we intentionally omit benchmarking at finer detail. We have further elaborated on the goals of the present study and its limitations in the Discussion, which now reads:

> One goal of the present study was to build an annotation tool that could cheaply produce accurate first-draft annotations. The identification of finer-grained cell type annotations including subtype and state is often context-specific and dependent on the practitioner. So, we focus our efforts here on benchmarking annotations at a broad level, attempting to provide accurate coarse annotations to best facilitate downstream analysis. Thus, we have not considered cell type annotation beyond the broad level, but higher resolution annotations may be investigated in future work.

Moreover, I would like to know the performance of each LLM when providing a set of cell types for their reference instead of using the strategy of zero-shot. This is also the common situation when users know or have expectations about what cells the sample contains.

Thank you for the suggestion. In response, we developed a function to annotate gene lists based on a list of expected cell types called ai_annotate_from_expected_cell_types() with documentation available here (https://ggit12.github.io/anndictionary/api/annotate/cells/de_novo.html#annotation-based-on-expected-cell-types) and links to function source code available in the documentation. This function uses chain-of-thought reasoning to first present the list of expected cell types to the LLM, then asking it to compare and contrast the gene lists presented, then asking it to provide annotations for each list on list at a time, and finally uses partial string matching with a similarity threshold to map the LLM labels back to the initial set of expected labels. The threshold allows the LLM labels that are not similar enough to the expected labels to pass through as new labels. Because this function uses chain of thought reasoning to help the LLM assign cell types to multiple gene lists in a single conversation, it is substantially more expensive to run and requires higher rate limits. Therefore, we benchmarked the annotation from expected cell types on a subset of the LLMs used previously that had higher API rate limits (excluding Claude 3 Opus due to cost). Overall, the performance of this strategy is similar to the zero-shot strategy that is the main focus of the manuscript. We now include the results of this evaluation in **Supplementary Table 1:**

Supplementary Table 1. LLM performance when annotating from a set of expected cell types using chain-of-thought reasoning. Agreement with manual annotations measured by yes/no, quality of match, and exact string agreement. Kappa with manual annotation and average kappa of the given model with every other model. All values are mean ± standard deviation across five replicates.

	Binary (%)		Perfect Match (%)		Extract String Match (%)		Kappa	
	Cells	By Cell Type	Cells	By Cell Type	Cells	By Cell Type	With Manual	Average With LLMs
Claude 3 Haiku	71.9 ± 1.6	45.4 ± 0.8	45.5 ± 1.3	25.2 ± 0.7	45.5 ± 1.3	25.2 ± 0.7	0.437 ± 0.013	0.624 ± 0.010
Claude 3.5 Sonnet	73.66 ± 0.22	51.44 ± 0.34	53.11 ± 0.24	31.2 ± 0.7	52.98 ± 0.25	30.6 ± 0.5	0.5112 ± 0.0026	0.616 ± 0.018
GPT-4o	73.4 ± 0.5	52.9 ± 0.9	50.3 ± 0.9	33.0 ± 1.0	50.2 ± 0.9	32.5 ± 1.0	0.485 ± 0.010	0.639 ± 0.005
GPT-4o mini	68.5 ± 0.8	46.9 ± 0.9	43.44 ± 0.25	26.9 ± 1.2	43.17 ± 0.25	26.8 ± 1.2	0.4134 ± 0.0027	0.567 ± 0.006
Gemini 1.5 Flash	55.0 ± 0.6	37.3 ± 0.7	34.4 ± 0.9	16.9 ± 0.8	34.3 ± 0.9	16.0 ± 0.7	0.321 ± 0.009	0.545 ± 0.013
Gemini 1.5 Pro	63.7 ± 0.9	44.4 ± 0.8	42.6 ± 0.9	24.0 ± 1.0	42.3 ± 0.8	23.4 ± 1.0	0.401 ± 0.009	0.596 ± 0.007
Plurality Vote	72.0 ± 0.5	50.3 ± 1.2	48.4 ± 0.8	28.9 ± 1.7	48.4 ± 0.8	28.5 ± 1.7	0.465 ± 0.008	0.720 ± 0.006

The Methods now read:

> *Annotation from expected cell types*
> The annotation from expected cell types was carried out starting from the same differentially expressed gene lists as used in the previously mentioned annotation benchmarking. To annotate gene lists, we used the ai_annotate_from_expected_cell_types() function from AnnDictionary, passing the unique cell types present in the given tissue as the expected cell types. This function uses a chain-of-thought reasoning approach where first, the LLM is supplied the tissue and expected cell types as context in the system prompt, where supported, or a message from the user if system prompts are not supported. Then, all differentially expressed gene lists to be annotated are supplied, and the LLM is asked to compare and contrast them. Finally, each list is presented again and requested to be labeled. The last step of this function is to use partial string matching to map the LLM-supplied labels back to the expected cell types, with a similarity threshold to allow new labels from the LLM to pass through to the returned annotations. Because the returned annotations are generally mapped back to the original set of expected annotations, these annotations are directly used to calculate agreement metrics without the postprocessing step mentioned in the previous annotation pipeline (described in the Methods section titled "Annotation post processing"). The source code modifications for this analysis are available under the from_expected branch of the benchmark_llms code repository.

The Results now read:

> *Annotation from expected cell types*
> We also benchmarked an LLM annotation strategy based on expected cell types. To do so, we designed a function that uses chain-of-thought reasoning to first present the list of expected cell types to the LLM, then asks it to compare and contrast the gene lists presented, then asking it to provide annotations for each list on list at a time, and finally uses partial string matching with a similarity threshold to map the LLM labels back to the initial set of expected labels. The threshold allows LLM labels that are not similar to the expected labels to pass through as new labels. Because this function uses chain of thought reasoning to help the LLM assign cell types to multiple gene lists in a single conversation, it is substantially more expensive to run and requires higher rate limits. Therefore, we benchmarked the annotation from expected cell types on a subset of the LLMs used previously that had higher API rate limits (excluding Claude 3 Opus due to cost). Here, the performances of the LLMs were only marginally lower compared to the previous annotation performances which represented a coarser scale, **Supplementary Table 1**. Furthermore, some degradation of performance is to be expected given that in this annotation strategy, the labels returned from the annotation function were not post-processed before being compared between LLMs and with manual annotations.

The Discussion now reads:

> In addition to direct LLM labeling of single lists of differentially expressed genes, we also tested two other annotation strategies: annotation by the ensemble vote of several LLMs, and annotation by chain-of-thought reasoning of multiple marker gene lists in a single conversation with expected cell types for context. Both of these methods add substantial runtime, financial expense, and complexity, but did yield general performance increases. Therefore, of the three

methods tested, the straightforward annotation of a single gene list at a time was the best-performing, simplest, and most cost-effective approach to annotating broad cell type.

3. One major concern for me is the use of GPT-4o to determine the best clustering resolution and Claude 3.5 Sonnet to correct cell type labels since both two methods are involved in benchmarking. For the resolution, you may try setting a resolution that is high enough to separate most cell types (e.g. resolution = 2).

In initial analysis trials, we tried setting a high resolution and then merging the labels afterward. We observed that using high resolutions generally yields small clusters that are enriched for potential artefact (i.e. many clusters whose differentially expressed genes are only mitochondrial). This is a natural thing to try when automatically labelling clusters (i.e. "If I don't have to do literature searches for each cluster, let's just overcluster the data a bit, then merge labels later"). Given the notedly experimental nature of the LLM-based resolution determination, we opted to remove this potentially deleterious step, and use a standard resolution=0.5 for each tissue. Therefore, we have revised the text to elaborate on the manual selection of resolution, and added a warning about this common pitfall to the manuscript. The methods now read:

> All the following steps were performed with Scanpy functions, accessed through AnnDictionary wrappers to parallelize across tissue. Starting from raw counts, we 1) normalized to 10,000 counts per cell, 2) log-transformed, 3) set high variance genes (top 2000 genes), 4) scaled each gene to zero mean and unit variance, 5) performed PCA, retaining the top 50 principal components, 6) calculated neighborhood graph 7) clustered the neighborhood graph using the Leiden algorithm with resolution=0.5, 8) calculated the UMAP embedding, 9) calculated differentially expressed genes for each cluster using the t-test and Benjamini-Hochberg-corrected p-values. All of these preprocessing parameters are within the range of those commonly used. With regard to the resolution for Leiden clustering, we opted to use the same moderate-to-low resolution value across all tissues. We initially tested higher resolutions in the range 2-5 with the intent to merge clusters based on their cell type identities. But we observed that doing so often yielded clusters that were enriched for genes typically thought of as artifactual such as mitochondrial and ribosomal signal. Thus, we caution the user against the over-cluster-and-merge strategy.

For the uniform set of cell types, I believe it can be manually corrected.

Our goal is to build an automated assessment pipeline so that the benchmarking results can be updated as new models become available. Thus, the manual curation of the set of cell types is not a feasible solution to achieve the goals of this manuscript.

The use of LLMs in post-processing and assessing free text is standard practice; see Raffel et al. for a discussion and list of relevant benchmark tasks and datasets.[18]

To address the concern that the choice of post-processing model can introduce bias, we ran replicates of the post-processing step using a second LLM, GPT-4o, giving an independent assessment of the same set of annotations. This specific provider and model was chosen based

on having sufficiently high API rate limits and having reasoning capability and knowledge on the higher end of models available. We then computed the same set of performance metrics based on these corrected labels, **Supplementary Table 3**. The performance metrics as assessed by independent post-processing models are highly correlated, **Supplementary Table 4**.

We now include **Supplementary Tables 3 and 4**:

Supplementary Table 3. LLM performance when post-processed by GPT-4o

	Binary (%)		Perfect Match (%)		Extract String Match (%)		Kappa	
	Cells	By Cell Type	Cells	By Cell Type	Cells	By Cell Type	With Manual	Average With LLMs
Claude 3 Haiku	78.42 ± 0.29	66.2 ± 1.0	58.9 ± 1.3	36.3 ± 2.8	58.5 ± 1.6	35.2 ± 3.1	0.557 ± 0.017	0.620 ± 0.007
Claude 3 Opus	82.09 ± 0.22	68.3 ± 0.7	68.7 ± 0.9	44.8 ± 2.0	68.2 ± 1.4	43.5 ± 2.5	0.658 ± 0.014	0.677 ± 0.012
Claude 3.5 Sonnet	83.0 ± 0.6	68.6 ± 0.7	70.4 ± 1.1	45.2 ± 2.4	69.9 ± 1.5	43.6 ± 3.5	0.678 ± 0.016	0.653 ± 0.016
Command R Plus	76.3 ± 0.4	57.4 ± 1.1	61.3 ± 0.8	34.3 ± 2.0	60.4 ± 1.3	32.8 ± 3.0	0.575 ± 0.014	0.604 ± 0.014
GPT-4	79.4 ± 0.7	64.0 ± 0.7	57.9 ± 1.7	34.2 ± 2.7	57.2 ± 1.8	33.2 ± 3.5	0.542 ± 0.019	0.591 ± 0.024
GPT-4o	80.4 ± 0.5	66.8 ± 1.1	67.3 ± 0.6	45.0 ± 2.7	66.8 ± 1.0	43.5 ± 3.3	0.644 ± 0.011	0.691 ± 0.013
GPT-4o mini	76.13 ± 0.33	64.2 ± 1.3	60.2 ± 1.0	37.3 ± 1.9	59.9 ± 1.2	36.4 ± 2.4	0.571 ± 0.012	0.649 ± 0.014
Gemini 1.5 Flash	67.90 ± 0.30	58.8 ± 0.4	47.9 ± 0.5	32.5 ± 2.2	47.5 ± 0.7	30.2 ± 3.1	0.447 ± 0.007	0.529 ± 0.014
Gemini 1.5 Pro	79.3 ± 1.5	66.9 ± 1.1	63.1 ± 2.3	40.2 ± 3.0	62.7 ± 2.7	39 ± 4	0.602 ± 0.028	0.628 ± 0.021
Llama 3.1 405B Instruct	82.3 ± 1.2	66.0 ± 1.4	65.6 ± 0.5	36.8 ± 2.7	65.2 ± 0.4	35.6 ± 3.1	0.625 ± 0.005	0.659 ± 0.011
Llama 3.1 70B Instruct	72.6 ± 1.8	59.8 ± 2.2	58.7 ± 2.0	35.8 ± 2.0	58.3 ± 1.8	35.0 ± 2.5	0.557 ± 0.019	0.623 ± 0.015
Llama 3.1 8B Instruct	56.0 ± 2.1	47.5 ± 1.5	43.6 ± 2.1	27.3 ± 1.8	42.9 ± 2.0	25.3 ± 2.7	0.397 ± 0.020	0.488 ± 0.019
Mistral Large	77.56 ± 0.33	62.9 ± 1.3	61.4 ± 1.5	39.9 ± 2.2	61.0 ± 1.7	38.3 ± 3.0	0.587 ± 0.018	0.655 ± 0.010
Plurality Vote	80.47 ± 0.26	67.4 ± 1.3	69.2 ± 0.5	45.5 ± 1.8	68.8 ± 0.6	44.2 ± 2.5	0.665 ± 0.006	0.742 ± 0.009

Supplementary Table 4. Comparison of agreement metrics calculated using two different LLMs for post-processing of annotations (Claude 3.5 Sonnet and GPT-4o).

	Pearson's R	P
Overall Binary (% of Cells)	0.99	6.5e-13
Overall Binary (% of Cell Types)	0.97	9.2e-09
Perfect Match (% of Cells)	0.99	9.5e-12
Perfect Match (% of Cell Types)	0.98	3.0e-09
Exact String Match (% of Cells)	0.99	1.5e-11
Exact String Match (% of Cell Types)	0.98	7.1e-10
Categorical Agreement (% of Cells)_1.0	0.99	9.5e-12
Categorical Agreement (% of Cells)_0.5	0.94	7.1e-07
Categorical Agreement (% of Cells)_0.0	1.00	2.0e-16
Categorical Agreement (% of Cell Types)_1.0	0.98	3.0e-09
Categorical Agreement (% of Cell Types)_0.5	0.78	9.9e-04
Categorical Agreement (% of Cell Types)_0.0	0.98	7.0e-10
Kappa with Manual Annotations	0.99	1.9e-11
Average Kappa with Other LLMs	0.99	8.5e-12

We thus conclude that the choice of post-processing model does not bias the results. By performing this procedure, we have also presented each model's performance when post-processed by a different model, which further addresses the reviewer's concern that the performance presented of Claude 3.5 Sonnet is based on labels post-processed by the same model. For the final set of results in the main manuscript, we opt to present the performance results as post-processed by a single model for simplicity and fairness of interpretation. For this data, we used the model that had the higher graduate-level reasoning capabilities,[19] and performance at our specific task.

The Results now read:

> *Annotation performance is robust to the LLM used in label post-processing*
> To address concern that the choice of post-processing model can introduce bias, we ran replicates of the post-processing step using a second LLM, GPT-4o, giving an independent assessment of the same set of annotations. We then computed the same set of performance metrics based on these corrected labels, **Supplementary Table 3**. The performance metrics as assessed by independent post-processing models are highly correlated, **Supplementary Table 4**. We thus conclude that the results are robust to choice of post-processing model.

The limitations section in the Discussion now reads:

> We also ensured that performance results were not due to the specific post-processing model used by reproducing similar performance with an independent post-processing model.

The Methods now read:

> *Annotation post-processing*
> In order to compute inter-LLM consistency, we needed to have a shared set of labels across all LLM-generated and manual annotations. Therefore, we used an LLM to create a unified set of cell type labels across all annotations (manual and LLM-based). We opted to use Claude 3.5 Sonnet for label unification because, during testing, we observed that it had the best reasoning capabilities and sufficiently high API rate limits. We also compared to results when using GPT-4o for the same task to assess if the choice of model biased the results. GPT-4o was chosen for this step due to sufficient API rate limits and similar reasoning capabilities.[19]

Minor comments:

1. For the instances that require manual correction, you might need to have a troubleshooting page for potential issues when using the tool.

Thank you for your comment. We agree that manual correction is an important part of LLM annotation of cell types. We have thus written an additional function adt.cell_type_marker_gene_score(), (source available here: https://ggit12.github.io/anndictionary/api/annotate/cells/generated/anndict.annotate.cell_type_marker_gene_score.html#anndict.annotate.cell_type_marker_gene_score) to make an independent assessment of the cell type labels, and included instructions on how to use this

function in a separate tutorial page. This function retrieves marker genes for a given set of cell types, then uses these lists to calculate gene module scores. This function serves as an independent assessment of the veracity of labels, can identify cases that may need correction, and help correct these cases. We have added this tutorial to the package's documentation here (https://ggit12.github.io/anndictionary/tutorials/annotate/automated_cell_type_marker_gene_scores.html), and the notebook is available as a .ipynb file in the repository here (https://github.com/ggit12/anndictionary/blob/main/nbs/tutorials/ipynb/automated_cell_type_marker_gene_scores.ipynb).

2. The number of top marker genes is a concern for me. The tools should let users select the number of marker genes to use. The paper should also explore the ideal number of marker genes regarding different levels of cell-type annotation tasks.

All LLM-based annotation functions in AnnDictionary allow the user to specify the number of marker genes to use via the argument n_top_genes, and the default is set to 10. Previous literature investigated the ideal number of marker genes to supply to GPT-4 when annotating cell type, and found that using the top 10 marker genes gave the best performance when annotating gene lists.[1] The study also showed that performance generally lowered when increasing the number of marker genes. This result is unsurprising given that the specificity of a gene list decreases with the number of genes and expanding the list of marker genes risks the inclusion of genes with smaller effect size. Given this context and the financial cost of increasing the number of genes supplied to the LLM, we find 10 to be the suitable number of marker genes to use when annotating broad cell type. We have now included this rationale in the discussion.

The discussion now reads:

> Previous investigation involving curated marker gene lists found that annotations based on the top 10 differentially expressed genes gave better performance than using the top 20 or 30.[1] Here, we used 10 marker genes and find that this gives satisfactory performance while keeping token usage minimal. Furthermore, using longer lists of marker genes risks compromising the gene lists' specificity and including genes with smaller effect size.

3. Please explain the cell, cell-type, and tissue-cell type level resolution in the methods part.

We have now added a subsection to elaborated what is meant by each resolution. The methods now read:

> **Resolutions.** We calculated the rate of agreement of each model with manual annotations at three resolutions: 1) cells, 2) cell types, **Table 2**, and 3) tissue-cell types, **Supplementary Figure 1**. The first resolution, cells, corresponds to treating each cell individually, and seeing the rate of agreement across all cells. We consider this metric to be the most representative of overall annotation performance, because it directly measures the overall proportion of cells that are correctly annotated across the dataset. The second resolution, cell types, is calculated by taking the mean of agreement of cells when grouped by the manual annotation column. This metric is useful for uncovering annotation performance of the model at each cell type, but, unlike the cell

resolution, does not take into account differential cell type abundance. The third resolution, tissue-cell type, is calculated as the mean agreement of within a single tissue-cell type. This metric was considered because cell types are known to have tissue-specific expression programs, and so it was possible that the resulting differentially expressed genes, and thus annotation performance, varied at the tissue-cell type level.

4. I suggest providing an example of the original annotation results shown in the form of UMAPs from each LLM.

Thank you for the suggestion. We have now included a set of example annotations of all cells detected in blood of Tabula Sapiens v2 from each LLM as a UMAP in **Figure 1B**.

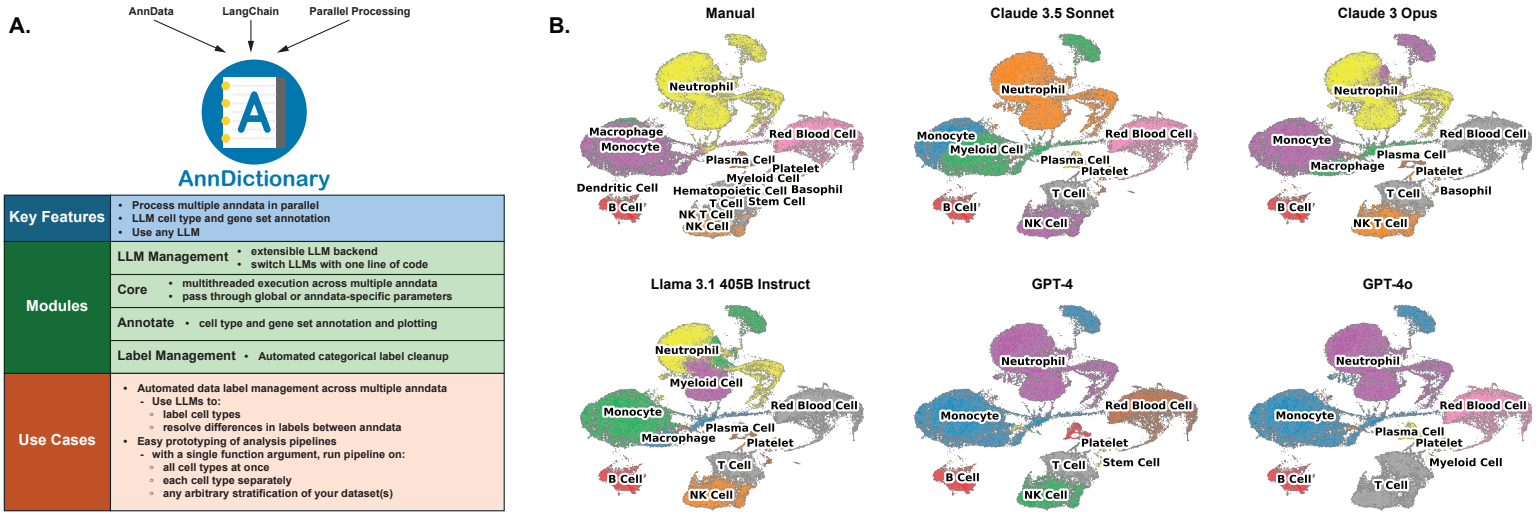


Figure 1. A. Overview of AnnDictionary—a python package built on top of LangChain and AnnData, with the goal of independently processing multiple anndata in parallel. **B.** Example LLM annotations of cell types and coarse manual annotations for all cells detected in blood of Tabula Sapiens v2.

The Results now read:

We show example LLM annotations of all cells detected in blood in **Figure 1B.**

5. Please provide an example of how the gene annotation function may help. This sounds like the Gene Ontotation function which is pretty common.

Thank you for your suggestion. These functions indeed produce similar output to other gene set annotation tools; the benefit of using LLMs is that they may be more frequently updated with literature than ontology databases, as well as being more flexible. Additionally, the outputs do not depend on specific databases of gene lists, which can yield results that are biased by gene set sizes and represents a substantial limitation of current gene set enrichment analysis.[5] We agree that some demonstration of this feature would be useful. We have added example usage to the function's documentation, see (https://ggit12.github.io/anndictionary/api/annotate/cells/generated/anndict.annotate.ai_ann

otate_biological_process.html#anndict.annotate.ai_annotate_biological_process). In addition, we have now written a tutorial notebook to illustrate how gene set annotation would be performed, and include an example of annotating three known gene sets from the Gene Ontology Biological Process database. We have added this notebook to both the GitHub repository for AnnDictionary (https://github.com/ggit12/anndictionary/blob/main/nbs/tutorials/ipynb/biological_process_annotation_with_an_LLM.ipynb), and the AnnDictionary documentation (https://ggit12.github.io/anndictionary/tutorials/annotate/biological_process_annotation_with_an_LLM.html).
We have also included this example in the main text. We have written a description of the gene set annotation process in the Methods:

> *Biological process annotation*
> We used gseapy v1.1.8 to access the 2023 release of the Gene Ontology Biological Process database. Then, to make a brief example, we retrieved the gene lists associated with the first three terms in the database, and used AnnDictionary's ai_biological_process function to label the gene sets with claude-3-5-sonnet-20240620.

We have additionally edited the results to include this example. The results now read:

> *Biological process annotation*
> To demonstrate the use of LLMs to annotate gene lists with the biological process that they represent, we annotated three gene lists of known processes from the Gene Ontology Biological Process database. In these three cases, the LLM annotations, while slightly broader, generally agreed with the existing labels, **Supplementary Table 7**.

We have included these results as **Supplementary Table 7:**

Supplementary Table 7. Example biological process annotations of known gene lists using an LLM (Claude 3.5 Sonnet).

GO Biological Process Term	Genes	LLM annotation
'De Novo' AMP Biosynthetic Process (GO:0044208)	ATIC, PAICS, PFAS, ADSS1, ADSS2, GART	Purine biosynthesis pathway
'De Novo' Post-Translational Protein Folding (GO:0051084)	SDF2L1, HSPA9, CCT2, HSPA6, ST13, ENTPD5, HSPA1L, HSPA5, PTGES3, HSPA8, HSPA7, DNAJB13, HSPA2, DNAJB14, HSPE1, DNAJC18, GAK, DNAJC7, DNAJB12, HSPA1A, ST13P5, HSPA1B, ERO1A, SELENOF, HSPA14, HSPA13, DNAJB1,	Protein folding and chaperone-mediated quality control.

	CHCHD4, DNAJB5, DNAJB4, SDF2, UGGT1	
2-Oxoglutarate Metabolic Process (GO:0006103)	IDH1, PHYH, GOT2, MRPS36, GOT1, IDH2, ADHFE1, GPT2, TAT, DLST, OGDHL, L2HGDH, D2HGDH, OGDH	Mitochondrial tricarboxylic acid (TCA) cycle and related metabolic pathways.

The Discussion now reads:

> In addition to comprehensive cell type annotation benchmarking, we also include biological inference functions (e.g. gene set annotation with biological processes). Previously published work has benchmarked LLM performance at this task.[4] It is convenient to consolidate these tasks into one package. A major limitation of current gene set enrichment analyses is their dependence on the sizes of the gene sets in the query database.[5] The use of LLMs—which do not rely on static lists of genes—to annotate pathways is therefore a promising solution to this issue.

6. In the abstract, it says We found that cell type annotation with AnnDictionary outperformed previous annotation with the same LLM by ~20% (58% vs 77%), and with other LLMs by ~25%. However, I didn't find any evidence to support the saying. For example, have you benchmarked with GPTCelltype?

Upon detailed review of the investigation mentioned in the above sentence, we found that the metrics presented were based on curated gene lists derived from either literature searches or previously annotated groups of cells.[1] Therefore, prior work by Hou and Ji provided important evidence that LLMs could identify cell types via marker genes, but did not assess the ability of LLMs to annotate cell type in the face of the full complexity of gene lists calculated from unannotated clusters derived de novo from unsupervised analysis. Marker gene lists derived de novo can be more complex due to a variety of factors, including the presence of artifactual genes and cell types split among multiple clusters.[2,3] We have therefore revised our manuscript to reflect that we have performed the first benchmarking of LLMs at *de novo* cell type annotation. Because we were assessing a fundamentally different task than that presented in Hou and Ji, it would be inappropriate to directly compare the performances in the context of performance gains or losses. We have therefore removed from our manuscript language that presents the observed performance differences as a performance gain, and updated our manuscript to reflect how the goals of the present study differ from previous investigation by Hou and Ji.

The abstract now reads:

> We developed an open-source package called AnnDictionary (https://github.com/ggit12/anndictionary/) to facilitate the parallel, independent analysis of multiple anndata. AnnDictionary is built on top of LangChain and AnnData and supports all common large language model (LLM) providers. AnnDictionary only requires 1 line of code to

configure or switch the LLM backend and it contains numerous multithreading optimizations to support the analysis of many anndata and large anndata. We used AnnDictionary to perform the first benchmarking study of all major LLMs at de novo cell-type annotation in Tabula Sapiens. LLMs varied greatly in absolute agreement with manual annotation based on model size. Inter-LLM agreement also varied with model size. We find that LLM annotation of most major cell types to be more than 80-90% accurate, and will maintain a leaderboard of LLM cell type annotation at https://singlecellgpt.com/celltype-annotation-leaderboard.

The introduction now reads:

> Previous work indicates that LLMs can reliably identify cell type from curated lists of marker genes—those identified via literature or calculated from previously identified cells of known type—but there has not yet been an assessment of LLMs at de novo cell type annotation, meaning annotation of gene lists derived directly from unsupervised clustering.[1] These gene lists crucially differ from curated gene lists, because they contain unknown signal and noise that may affect the annotation process.[2,3] De novo annotation is therefore a potentially more challenging task. The goal of this investigation is to assess the effectiveness of LLMs in this context. So, we used AnnDictionary to benchmark the de novo cell type annotation ability of the major commercially available LLMs (i.e. all models from OpenAI, Anthropic, Google, Meta, and all additional available text generation models on Amazon Bedrock, including Mistral, Titan, and Cohere).

The Discussion now reads:

> Previous work has assessed GPT-4's ability to annotate curated lists of genes, including lists derived from Tabula Sapiens v1[1] We build on this previous work by assessing LLMs' abilities to annotate the full complexity of gene lists derived from unsupervised clustering. In the present study, we used Tabula Sapiens v2—which contains more than double the number of cells compared to Tabula Sapiens v1. On this larger dataset, we find the annotation of major cell types to be more than 80-90% accurate, making LLM-based annotation a viable option for first-pass cell type annotation. The flexibility of LLM-generated annotations solves a major problem of automated annotation procedures, which have historically lacked flexibility due to the need to use a reference set of annotations.[3] Furthermore, the LLM-based approach is reference-free and so does not require additional datasets, which could otherwise increase computational burden.

Reviewer #3 (Remarks to the Author):

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

Reviewer #4 (Remarks to the Author):

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

Reviewer #5 (Remarks to the Author):

General assessment of the work:

The use of LLMs in research has been of great interest and new packages that ease their use for biological data processing are very welcomed. Such packages ease the evaluation, comparison, and assessment of the usability of LLMs for various tasks and various types of datasets. The annotation of cell types in single cell (and spatial transcriptomics) experiments is one such task for which the usage of LLMs are explored. Further characterization of the cases and extent to which LLMs are appropriate for this task is needed, which in turn requires good software support. Finally, it is welcomed that the authors acknowledge and enable the interaction between human and LLM so educated decisions can be made in unclear cases.

Major comments:

1) The most important component of this manuscript is the underlying software. In the fast paced development of the field, it is of utmost importance to create extensible packages. The authors claim: “AnnDictionary is extensible and can grow to accommodate additional methods”, therefore special attention should be made to ensure these claims are valid. The standard way to ensure extensibility is by modular design and separation of responsibilities. The code base however does not show such characteristics at the abstraction level beyond the definition of functions. Please revise and refactor your code so that it supports maintainability and extensibility well. Pylint can help identify vulnerabilities and smells at the code level.
Thank you for the helpful critique. We have extensively refactored the code base for modularity and to follow the best practices of python package design. We also addressed all pylint warnings. Please see our detailed descriptions of the major changes below in response to your Code Availability critique.

2) The authors describe the cell type annotation method as “First, we put the LLM in a feedback loop with Scanpy to create an agent that attempts to determine cluster resolution automatically from UMAP plots.” The human cell annotators use UMAP plots due to the limitations of human perception that requires visualizing the data in 2D, while machine learning models can work with the higher dimensional data (e.g. the 2000 most variable genes) to find clusters. Can the authors provide arguments for this method or references where UMAPs are used as input to (AI) models?
The original method was designed to mimic the human expert annotation as closely as possible. Thus, we included the iterative resolution selection process in the annotation pipeline. Note that in our initial pipeline, we included manual correct of the resolutions for each tissue.

Chart-image reasoning is an established task family for AI models, and several benchmark datasets exist to evaluate models directly on rasterized charts, including FigureQA, DVQA,

PlotQA, and ChartQA.[20-23] Therefore, we improved the sentence that you highlighted for clarity, and provide references to support this feature. The Results now read:

> *Cell type annotation*
> We designed an LLM agent that attempts to determine cluster resolution automatically from UMAP plots. Chart-based reasoning is an established task family of LLMs.[20-23] We note that current LLMs do not seem good enough to reliably produce reasonable resolutions, but still may be a useful first pass and this capability may eventually improve.

However, given the notedly experimental nature of the LLM-based resolution determination, we opted to remove this potentially deleterious step, and use a standard resolution=0.5 for each tissue. Therefore, we have revised the text to elaborate on the manual selection of resolution, and included a warning about setting artificially high resolutions, which is a potential pitfall when using LLMs to annotate clusters. The methods now read:

> All the following steps were performed with Scanpy functions, accessed through AnnDictionary wrappers to parallelize across tissue. Starting from raw counts, we 1) normalized to 10,000 counts per cell, 2) log-transformed, 3) set high variance genes (top 2000 genes), 4) scaled each gene to zero mean and unit variance, 5) performed PCA, retaining the top 50 principal components, 6) calculated neighborhood graph 7) clustered the neighborhood graph using the Leiden algorithm with resolution=0.5, 8) calculated the UMAP embedding, 9) calculated differentially expressed genes for each cluster using the t-test and Benjamini-Hochberg-corrected p-values. All of these preprocessing parameters are within the range of those commonly used. With regard to the resolution for Leiden clustering, we opted to use the same moderate-to-low resolution value across all tissues. We initially tested higher resolutions in the range 2-5 with the intent to merge clusters based on their cell type identities. But we observed that doing so often yielded clusters that were enriched for genes typically thought of as artifactual such as mitochondrial and ribosomal signal. Thus, we caution the user against the over-cluster-and-merge strategy.

3) "To assess label agreement between the LLMs and manual annotations, we used an LLM to automate the comparison of cell type labels because this allowed us to meet the scale requirements of the present study, which involved assessing the agreement of ~10,000 unique pairs of labels". Can the authors clarify why the usage of LLMs is necessary in this case? Comparison between annotations may be done programmatically without exposure to hallucinations.

Our goal is to build an automated assessment pipeline so that the benchmarking results can be updated as new models become available. Our rationale for using LLMs to make label comparisons is that LLMs are the state-of-the-art method for free text comparison and when assessing conceptual distances, especially because cell type labels can, at times, be semantically close but typographically far, for example "T-helper cell" and "CD4+ lymphocyte.", or typographically close but semantically far, for example "T cell" and "B cell".

Thus, we believe LLMs offer a powerful solution when assessing free text labels. The use of LLMs in comparing free text results is standard practice, and this approach was formally

investigated by Zheng et al. in their 2023 work, “Judging LLM-as-a-judge with MT-Bench and Chatbot Arena”, and also by Fu et. Al in their 2024 work, “GPTScore: Evaluate as You Desire”.[24,25] However, the reviewer’s concern about the exposure to hallucination is valid. To address this concern, we have included as an additional agreement metric the direct string comparison between labels. The direct string comparison metric almost exactly matches the rate of “perfect matches” in the dataset. This is perhaps unsurprising given that the function that queries the LLM for the quality of match itself contains a short-circuit to return “perfect match” in the case of a direct string match between query labels. These two metrics indicate that the LLM rarely rates different cell type labels as a perfect match if they aren’t exact string matches. So overall, the exposure to hallucination appears to be minimal.

We have now elaborated on the motivation behind this methodology in the methods section:

> **Agreement metrics.** For each unique pair of manual and LLM cell type labels, we used a function that asks the LLM if the two annotations agree, and then process the response to get a binary output (0 for no, 1 for yes). We also asked the LLM to rate the quality of the label match as perfect, partial, or non-matching and calculated the rate of “perfect” matches among individual cells and cell types, **Table 1**, and visualized the rates of all match qualities, **Figure 2A**, **B**. Finally, we also assessed the agreement via direct string comparison between the labels.
>
> We note that the direct string comparison is overly conservative and represents a lower bound on performance. This is because cell type labels can, at times, be semantically close but syntactically far, for example “T-helper cell” and “CD4+ lymphocyte.”, or syntactically close but semantically far, for example “T cell” and “B cell”. Our rationale for using an LLM to compare labels and rate the quality of matches was that LLMs were the state-of-the-art extension of any partial/fuzzy matching logic for strings, and it is not clear how well these label differences can be programmatically resolved. Finally, there is a standard ontology for cells. However, the first step of annotation typically uses free text labels, and most dataset annotations are not mapped to terms in the standard ontology. Furthermore, converting free text labels to standard terminology can lose specificity desired by individual research projects for their particular applications.[26] Annotation is also an iterative process that involves updating labels with the most up-to-date knowledge, and there can be lag in updating formal ontologies. The tool developed in this study is designed to be used in tandem with a researcher to increase the speed of draft annotation generation. Thus, we believe it is of most relevance from a practical perspective to assess the quality of free text labels provided by LLMs, as that is the type of label that will most likely be used by the majority of users, especially at early stages of projects.
>
> Finally, we measured Cohen’s kappa between each LLM label column and the manual label column as additional metrics of agreement.

We have added an exact string match column to **Table 1,** as well as all tables containing the same set of performance benchmarks (Supplementary Tables 1, 2, 3, and 5):

Table 1. LLM performance. Agreement with manual annotations measured by yes/no, quality of match, and exact string agreement. Kappa with manual annotation and average kappa of the given model with every other model. All values are mean ± standard deviation across five replicates.

	Binary (%)		Perfect Match (%)		Extract String Match (%)		Kappa	
Model	Cells	By Cell Type	Cells	By Cell Type	Cells	By Cell Type	With Manual	Average With LLMs
Claude 3 Haiku	78.3 ± 0.8	66.2 ± 1.5	61.8 ± 2.3	47 ± 4	61.8 ± 2.3	47 ± 4	0.589 ± 0.023	0.652 ± 0.023
Claude 3 Opus	82.3 ± 0.9	69.8 ± 1.0	72.8 ± 2.6	54 ± 4	72.7 ± 2.6	54 ± 4	0.704 ± 0.026	0.711 ± 0.020
Claude 3.5 Sonnet	84.0 ± 0.7	70.5 ± 1.2	74.4 ± 2.7	54 ± 4	74.3 ± 2.6	53 ± 4	0.721 ± 0.027	0.697 ± 0.026
Command R Plus	77.2 ± 1.0	59.4 ± 2.6	64.5 ± 2.6	40 ± 5	64.5 ± 2.6	40 ± 5	0.616 ± 0.027	0.646 ± 0.026
GPT-4	79.2 ± 0.9	64.3 ± 1.9	64 ± 4	44 ± 5	64 ± 4	44 ± 5	0.61 ± 0.05	0.65 ± 0.04
GPT-4o	80.9 ± 0.7	70.1 ± 2.8	70.4 ± 2.5	54 ± 6	70.4 ± 2.5	54 ± 6	0.680 ± 0.026	0.721 ± 0.021
GPT-4o mini	76.8 ± 1.0	66.2 ± 1.6	63.4 ± 3.0	47 ± 6	63.4 ± 3.0	47 ± 6	0.605 ± 0.031	0.681 ± 0.022
Gemini 1.5 Flash	68.8 ± 1.3	60.8 ± 2.7	51.0 ± 2.5	41 ± 5	51.0 ± 2.5	41 ± 5	0.478 ± 0.024	0.561 ± 0.020
Gemini 1.5 Pro	77.5 ± 1.8	67.9 ± 0.8	65.1 ± 2.4	50 ± 5	65.1 ± 2.4	50 ± 5	0.625 ± 0.024	0.658 ± 0.019
Llama 3.1 405B Instruct	82.0 ± 1.0	64.9 ± 2.7	69.5 ± 2.6	47 ± 5	69.3 ± 2.6	47 ± 5	0.667 ± 0.027	0.690 ± 0.021
Llama 3.1 70B Instruct	74 ± 4	61.6 ± 1.5	64 ± 4	46 ± 5	64 ± 4	45 ± 5	0.62 ± 0.04	0.665 ± 0.022
Llama 3.1 8B Instruct	59 ± 4	53 ± 4	47.7 ± 3.2	37 ± 6	47.6 ± 3.2	36 ± 6	0.440 ± 0.031	0.526 ± 0.030
Mistral Large	78.1 ± 1.5	66.2 ± 2.0	64.9 ± 2.7	50 ± 5	64.8 ± 2.8	49 ± 5	0.623 ± 0.027	0.696 ± 0.024
Plurality Vote	80.5 ± 0.9	69.4 ± 1.7	72.4 ± 2.1	55 ± 5	72.3 ± 2.1	55 ± 5	0.700 ± 0.022	0.770 ± 0.018

Additionally in the context of using LLMs to compare free text, Zheng et al. discuss self-enhancement bias—the phenomenon that LLMs may prefer their own answers to those produced by other models. To assess the presence of self-enhancement bias in the performance ratings presented in this study, we ran replicate experiments to compare the performances as rated by a second model, GPT-4o, **Supplementary Table 5**. This specific provider and model was chosen based on having sufficiently high API rate limits and having reasoning capability and knowledge on the higher end of models available. The performances of each LLM when assessed by Claude 3.5 Sonnet vs. GPT-4o were highly correlated, **Supplementary Table 6**, indicating that the effect of self-enhancement bias seems to be minimal.

Supplementary Table 5. LLM performance when rated by GPT-4o.

	Binary (%)		Perfect Match (%)		Extract String Match (%)		Kappa	
	Cells	By Cell Type	Cells	By Cell Type	Cells	By Cell Type	With Manual	Average With LLMs
Claude 3 Haiku	80.4 ± 1.4	71.2 ± 2.1	61.8 ± 2.3	47 ± 4	61.8 ± 2.3	47 ± 4	0.589 ± 0.023	0.652 ± 0.023
Claude 3 Opus	84.1 ± 1.9	72.5 ± 2.0	72.8 ± 2.6	54 ± 4	72.7 ± 2.6	54 ± 4	0.704 ± 0.026	0.711 ± 0.020
Claude 3.5 Sonnet	87.4 ± 2.9	74.9 ± 1.9	74.4 ± 2.7	54 ± 4	74.3 ± 2.6	53 ± 4	0.721 ± 0.027	0.697 ± 0.026
Command R Plus	78.0 ± 1.3	59.3 ± 3.4	64.5 ± 2.6	40 ± 5	64.5 ± 2.6	40 ± 5	0.616 ± 0.027	0.646 ± 0.026
GPT-4	80.3 ± 1.8	68.8 ± 2.3	64 ± 4	44 ± 5	64 ± 4	44 ± 5	0.61 ± 0.05	0.65 ± 0.04
GPT-4o	83.9 ± 2.6	73.7 ± 3.3	70.4 ± 2.5	54 ± 5	70.4 ± 2.5	54 ± 6	0.680 ± 0.026	0.721 ± 0.021
GPT-4o mini	80.4 ± 2.6	69.5 ± 2.1	63.4 ± 3.0	47 ± 6	63.4 ± 3.0	47 ± 6	0.605 ± 0.031	0.681 ± 0.022
Gemini 1.5 Flash	73.1 ± 2.8	65.7 ± 2.9	51.0 ± 2.5	41 ± 5	51.0 ± 2.5	41 ± 5	0.478 ± 0.024	0.561 ± 0.020
Gemini 1.5 Pro	80.5 ± 2.2	72.1 ± 1.9	65.1 ± 2.4	50 ± 5	65.1 ± 2.4	50 ± 5	0.625 ± 0.024	0.658 ± 0.019
Llama 3.1 405B Instruct	83.6 ± 1.3	70.0 ± 3.2	69.5 ± 2.6	47 ± 5	69.3 ± 2.6	47 ± 5	0.667 ± 0.027	0.690 ± 0.021
Llama 3.1 70B Instruct	77.5 ± 2.6	67.3 ± 2.1	64 ± 4	45 ± 4	64 ± 4	45 ± 5	0.62 ± 0.04	0.665 ± 0.022
Llama 3.1 8B Instruct	63.2 ± 3.4	58 ± 4	47.7 ± 3.2	37 ± 5	47.6 ± 3.2	36 ± 6	0.440 ± 0.031	0.526 ± 0.030
Mistral Large	81.4 ± 1.9	70.1 ± 2.6	64.9 ± 2.7	50 ± 5	64.8 ± 2.8	49 ± 5	0.623 ± 0.027	0.696 ± 0.024
Plurality Vote	83.5 ± 2.3	73.1 ± 2.5	72.4 ± 2.1	55 ± 5	72.3 ± 2.1	55 ± 5	0.700 ± 0.022	0.770 ± 0.018

Supplementary Table 6. Comparison of agreement metrics calculated using two different LLMs (Claude 3.5 Sonnet and GPT-4o) to rate annotations.

	Pearson's R	P
Overall Binary (% of Cells)	0.99	6.2e-11
Overall Binary (% of Cell Types)	0.96	6.9e-08
Perfect Match (% of Cells)	1.00	2.6e-37
Perfect Match (% of Cell Types)	1.00	5.4e-22
Exact String Match (% of Cells)	1.00	0.0e+00
Exact String Match (% of Cell Types)	1.00	0.0e+00
Categorical Agreement (% of Cells)_1.0	1.00	2.6e-37
Categorical Agreement (% of Cells)_0.5	0.99	7.6e-11
Categorical Agreement (% of Cells)_0.0	0.99	9.4e-13
Categorical Agreement (% of Cell Types)_1.0	1.00	5.4e-22
Categorical Agreement (% of Cell Types)_0.5	0.95	1.8e-07
Categorical Agreement (% of Cell Types)_0.0	0.97	1.7e-08
Kappa with Manual Annotations	1.00	0.0e+00
Average Kappa with Other LLMs	1.00	0.0e+00

The Methods now read:

> *Assessment of self-enhancement bias*
> To assess the presence of self-enhancement bias on the performance ratings presented in this study, we ran replicate experiments to compare the performances as rated by a second model, GPT-4o. For consistency, the same set of post-processed annotations were used here as were used in the main assessment of performance.

The Results now read:

> *Annotation performance is robust to the LLM used as rater*
> Self-enhancement bias refers to the potential behavior LLMs to prefer their own answers over those from other LLMs.[24] To assess the presence of self-enhancement bias in the performance ratings presented in this study, we ran replicate experiments to compare the performances as rated by a second model, GPT-4o, **Supplementary Table 5**. The performances of each LLM when assessed by Claude 3.5 Sonnet vs. GPT-4o were highly correlated, **Supplementary Table 6**, indicating that the effect of self-enhancement bias seem to be minimal.

The Discussion now reads:

> Finally, we showed that self-enhancement bias did not substantially influence the annotation ratings by reproducing similar performances with multiple independent models. The lack of observed self-enhancement bias is not surprising because the cell type annotations are short strings, and so it seems unlikely that they contain substantial stylistic information. While we used an LLM to rate the quality of matches, we also present these results alongside the much stricter exact string match agreement.

Minor comments:

1) Subfigures in figure 4 feature ellipses one each. It is unclear why these ellipses were necessary and how they relate to the main text. Can the authors clarify these, preferably adding their meaning to the caption? Are these on Figure 4 A the top 10 cell types later shown in the confusion matrix at Figure 5 A?

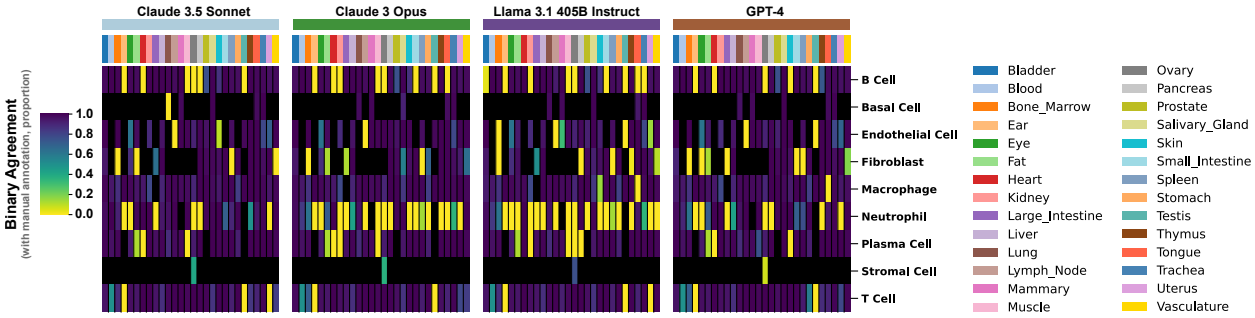
The ellipses were meant to highlight two potentially interesting regions of the plot. We have now elaborated on their purpose in the figure legend. Yes, the ellipse in 4A is meant to highlight the region of interest investigated further in Figure 5. The figure legend now reads:

> **Figure 4. A.** Inter-rater agreement within the top 4 performing LLMs vs. agreement with manual annotation for each manual cell type annotation, with marginal kernel density estimates stratified by tertile of cell type population size. Red, yellow, and green represent the bottom, middle, and top tertile of cell type by population size, respectively. **B.** Same set of axes as (A), with dot sizes scaled by their respective cell type populations size, and with kernel density estimates scaled by population size as well. The manually drawn ellipses outline two regions of interest: (A) the cell types with the highest inter-rater agreement and lowest agreement with manual annotation—which are the subject of Figure 5, and (B) the cell types with the highest inter-rate agreement and highest agreement with manual annotation—which includes the most abundant cell types discussed earlier.

2) The supplementary figures are missing the x axis label for tissues, which is understandable for Figure S1 due to size limit, but please consider adding it to Figure S2 for enabling visual investigations of model performances per tissue and cell type.

Thank you for the suggestion. We have now added tissue legends to both **Supplementary Figures 1 and 2**.

Supplementary Figure 1. Rates of agreement with manual annotation as measured by a binary response for the four large language models with the top performance based on this metric in this exemplar run. Data is plotted at the tissue-cell type resolution, where columns are tissues and rows are cell types, repeated for each large language model.



Supplementary Figure 2. Rates of agreement with manual annotation for the 10 largest cell types by number of cells available. As in **Supplementary Figure 1**, agreement is measured as a binary response, data is plotted at tissue-cell type resolution, and visualization is repeated for each of the four large language models with the highest overall binary agreement rate in this exemplar run.

3) The authors hint several times at the disagreement between human expert and LLM labels. Limitations are also elaborated in the Discussion section. Although beyond the scope of this current manuscript, using their tool would potentially enable assessing the time spent with or without using LLMs for annotating cell types for researchers at different career stages which would be beneficial for the community.

Thank you for this thoughtful suggestion; it raises an interesting point for future work.

Reviewer #5 (Remarks on code availability):

Installation was possible, however it would have been much more user friendly and up to current standards, to provide a package through PyPi. It would also have been appreciated to include a conda environment and/or a container to help installation of dependencies used for the tutorials.

Thank you for your feedback on the package installation experience and we appreciate your suggestions regarding installation methods. We have now made our package available through pip via PyPi (https://pypi.org/project/anndict/), which addresses your primary concern about aligning with current standards for software distribution.

We have now provided a Singularity image file to use for the benchmark_llms pipeline and added download instructions to the repository's README. This Singularity image contains an installation of AnnDictionary as well.

Finally, we have provided an Anthropic API key for testing the LLM features.

I could not install cellxgene_census on my Fedora machine, so I was not able to run the basic tutorial.

We now use the sample dataset included with Scanpy for tutorials. This dataset is both smaller and does not require the installation of cellxgene_census. This change should resolve the installation difficulties you encountered on your Fedora machine and ensure that users can successfully run the tutorials as long as Scanpy installation is possible.

I tried running the cell type annotation tutorial with a small sample data instead. I was not able to complete the tutorial within 2 hours of troubleshooting. Last error message at which I gave up was at “set_high_variance_genes_adata_dict“ after I already installed tbb:

> Error processing U11 on attempt 0: No threading layer could be loaded.
> HINT:
> Intel TBB is required, try:
> $ conda/pip install tbb
> Failed to process U11 after 0 attempts.

Note that the error message suggests installing tbb with conda/pip, however the readme says pip is not supported. Please update the error message accordingly.

Thank you for finding this issue, which we have now addressed in several ways:

1. We've rewritten our installation documentation to prevent this confusion. The error message you encountered is from the concurrent futures package, which incorrectly suggests using pip to install TBB. We now explicitly document that conda installation is required for TBB.
2. We've added platform-specific installation instructions at https://ggit12.github.io/anndictionary/installation.html, including:
    - Separate instructions for Linux and macOS
    - Details about our testing on Linux (v3.10, v4.18) and macOS (v13.5, v14.7)
3. We've added a dedicated "Multithreading Compatibility Note" section that:
    - Explains why we configure the Numba threading layer to TBB on macOS
    - Provides step-by-step troubleshooting instructions
    - Specifically warns users to ignore the misleading error message about pip installation

To directly address your error: If you encounter the TBB threading layer error again, please follow these specific steps:

```
bash
# First, remove existing installations
pip uninstall numba tbb intel-tbb
conda remove tbb numba


# Then install via conda (not pip)
conda install -c conda-forge tbb numba
```

We appreciate your feedback as it helped us improve our documentation and user experience. Please let us know if you encounter any other issues.

It is strongly recommended to use linters (e.g. Pylint, Flake8) during development which can catch code smells before they cause an issue or hinder maintainability and further development. When pylint is enabled, among others, the following smells are found.

> "eval" is used in line 748 and 821 of ai.py which is an unsafe solution against code injection attacks. Too general "Exceptions" are caught in line 568 and 638 of ai.py which may hide issues and raise unhelpful error messages. Global variables are used or modified in line 404, 441, 442 and 446 of ai.py both within and outside of functions, which hinders readability and the robustness of the code. Inconsistent use of protected class members (i.e. variables starting with _) both as the previously mentioned globals in ai.py and in line 612 of dict.py. Possibly using variable 'cleaning_func' before assignment in line 1784 of dict.py due to missing a "else" statement in line 1765. Unused argument in line 1607 in dict.py, line 100 in spatial_dict.py, and line 222 and 527 of stablelabel.py. Cell-var-from-loop (https://pylint.readthedocs.io/en/latest/user_guide/messages/warning/cell-var-from-loop.html) issue is found in line 1029 of ai.py that likely causes unintended behaviour.

Furthermore, the code base lacks architecture considerations. Too many functions are within too generically named files, such as ai.py and dict.py with more than 1000 and almost 2000 lines of code, respectively. One side effect is that the documentation is organized into functions but no higher abstraction level which makes navigation difficult. As noted in the readme, using objects would be a beneficial next step. There is a complete lack of unit tests raising concerns over the testability and extensibility of the code base. All of these give the impression that the code is not mature yet, the design is not well thought-through, and the project likely suffers from high technical debt which will hinder future development.

Some suggestions for modularization are listed here. Initialization functions for LLM providers can be organized into their own file, you may consider inheritance for these. Provider mapping, providers and provider model dictionaries can be replaced with dataclasses or enums for more development support. Configuration can have its own module, as well as the user interface, API calls to the providers, and many other facets of the current tool can be explicitly moved to their own module. Examples specific for this manuscript and for other illustration purposes can be moved to their own subfolder so that it is not bundled with the main package. Many references can be found online, for example this one: https://nsworldinfo.medium.com/functions-and-modules-in-python-simplifying-code-organization-and-reusability-7e4626d0ce9e

Thank you for your helpful suggestions regarding modularity and the best practices of python package design. We have extensively refactored the codebase into more modules, restructured modules to use object-oriented classes and dataclasses, and broken off the pipeline relevant to the present manuscript as its own Github repository. We have also addressed the pylint warnings as appropriate. See our detailed report below for specifics.

Resolution of the original lint findings

We have now corrected all linter warnings, or muted them where justified with targeted pylint: disable comments. All formerly unsafe eval calls were replaced with ast.literal_eval, broad except: clauses were

narrowed to specific exception classes, and error handling was re-worked in the core AdataDict class methods so genuine errors surface clearly, and could be silenced when running large batches where silent error catching is desired. The global variables in ai.py were folded into a dedicated LLMManager class. Inconsistent use of “protected” underscore members was resolved by renaming, or were handled in the broad refactor further detailed below.

Architectural overhaul

All previous modules, including the two former thousand-line modules, have been decomposed into sub-packages enumerated in the documentation. We have broadly refactored the core class—AdataDict—so that all relevant methods are defined within the class, and then we exposed these through modules that simply wrap AdataDict class methods if the user desires a functional interface. Furthermore, LLM provider configuration is now handled through a separate module, which contains the dataclass LLMProviderConfig, while a base class base_llm_initializer and a default class default_llm_initializer centralize defaults. The class LLMManager now handles all backend LLM configuration and client caching. We moved all annotation functions to their own subpackage—anndict.annotate—where each style of annotation has its own module. We also moved plotting functions to anndict.plot, categorical label management and cleanup functions to anndict.automated_label_management, adata_dict_fapply wrappers to common functions to anndict.wrappers—where wrappers are organized into modules based on the package that they wrap. All subpackages and modules discussed here have detailed documentation, organized hierarchically, and for many, we have added instructive examples and tutorials, which are housed as rendered notebooks in the documentation (https://ggit12.github.io/anndictionary/tutorials/index.html), or as .ipynb files on the github repository (https://github.com/ggit12/anndictionary/tree/main/nbs/tutorials/ipynb) , or as example blocks within the function documentation itself.

Documentation & contributor guidance

The Sphinx documentation was completely rebuilt and now offers a clear **Overview** (https://ggit12.github.io/anndictionary/overview.html) with an architectural narrative, and an expanded **Contribute** section (https://ggit12.github.io/anndictionary/contribute/index.html) that includes brief architectural considerations and style conventions. Furthermore, the modules are hierarchically documented as above mentioned.

Testing & continuous integration

We have written 372 unit tests. A new tests/ directory now tests every sub-package and has 87% line coverage, which is above the generally accepted 80% line coverage threshold.

Examples, tutorials, and manuscript-specific code

All end-user tutorials are delivered exclusively as Jupyter notebooks under a dedicated **Tutorials** section that Sphinx renders but which remain outside the import path of the library, ensuring example code does not bloat the distribution. Meanwhile, the full benchmarking workflow that underpins the associated manuscript has been extracted to a standalone repository (https://github.com/ggit12/benchmark_llms), and refactored to use a reproducible Snakemake pipeline. This repository also contains an environment template for safe API key management, helper scripts, pipeline-specific functions, and figure-generation logic.

References

1 Hou, W. & Ji, Z. Assessing GPT-4 for cell type annotation in single-cell RNA-seq analysis. *Nat Methods* (2024). https://doi.org:10.1038/s41592-024-02235-4

2 Kiselev, V. Y., Andrews, T. S. & Hemberg, M. Challenges in unsupervised clustering of single-cell RNA-seq data. *Nat Rev Genet* **20**, 273-282 (2019). https://doi.org:10.1038/s41576-018-0088-9

3 Luecken, M. D. & Theis, F. J. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol Syst Biol* **15**, e8746 (2019). https://doi.org:10.15252/msb.20188746

4 Hu, M. *et al.* Evaluation of large language models for discovery of gene set function. *ArXiv* (2024).

5 Karp, P. D., Midford, P. E., Caspi, R. & Khodursky, A. Pathway size matters: the influence of pathway granularity on over-representation (enrichment analysis) statistics. *BMC Genomics* **22**, 191 (2021). https://doi.org:10.1186/s12864-021-07502-8

6 Lee, R. D. *et al.* Single-cell analysis identifies dynamic gene expression networks that govern B cell development and transformation. *Nat Commun* **12**, 6843 (2021). https://doi.org:10.1038/s41467-021-27232-5

7 Dominguez Conde, C. *et al.* Cross-tissue immune cell analysis reveals tissue-specific features in humans. *Science* **376**, eabl5197 (2022). https://doi.org:10.1126/science.abl5197

8 Didonna, A. *et al.* Ataxin-1 regulates B cell function and the severity of autoimmune experimental encephalomyelitis. *Proc Natl Acad Sci U S A* **117**, 23742-23750 (2020). https://doi.org:10.1073/pnas.2003798117

9 Meta. *llama3_1/MODEL_CARD.md*, <https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/MODEL_CARD.md> (

10 Amazon. *Amazon Titan Text Lite and Titan Text Express - AWS AI Service Cards*, <https://docs.aws.amazon.com/ai/responsible-ai/titan-text/overview.html> (

11 Cohere. *Cohere's Command R+ Model (Details and Application)*, <https://docs.cohere.com/v2/docs/command-r-plus#unique-command-r-model-capabilities> (

12 AI, M. *Changelog*, <https://docs.mistral.ai/getting-started/changelog/> (

13 Google. *Gemini models*, <https://ai.google.dev/gemini-api/docs/models> (

14 OpenAI. Compare models.

15 Anthropic. *How up-to-date is Claude's training data?*, <https://support.anthropic.com/en/articles/8114494-how-up-to-date-is-claude-s-training-data> (

16 Zeng, Z. *et al.* OmicVerse: a framework for bridging and deepening insights across bulk and single-cell sequencing. *Nat Commun* **15**, 5983 (2024). https://doi.org:10.1038/s41467-024-50194-3

17 Stephen R Quake, T. T. S. C. Tabula Sapiens reveals transcription factor expression, senescence effects, and sex-specific features in cell types from 28 human organs and tissues. *bioRxiv* (2024). https://doi.org:https://doi.org/10.1101/2024.12.03.626516

18 Colin Raffel, N. S., Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* **21**, 1-67 (2020).

19 Anthropic. (2024).
20 Bengio, S. E. K. a. V. M. a. A. A. a. A. K. a. A. T. a. Y. FigureQA: An Annotated Figure Dataset for Visual Reasoning. *arXiv* (2018).
21 Kanan, K. K. a. B. P. a. S. C. a. C. DVQA: Understanding Data Visualizations via Question Answering. *arXiv* (2018).
22 Kumar, N. M. a. P. G. a. M. M. K. a. P. PlotQA: Reasoning over Scientific Plots. *arXiv* (2020).
23 Hoque, A. M. a. D. X. L. a. J. Q. T. a. S. J. a. E. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. *arXiv* (2022).
24 Lianmin Zheng, W.-L. C., Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv* (2024). https://doi.org:https://doi.org/10.48550/arXiv.2306.05685
25 Jinlan Fu, S.-K. N., Zhengbao Jiang, and Pengfei Liu. GPTScore: Evaluate as You Desire. *arXiv* (2023). https://doi.org:https://doi.org/10.48550/arXiv.2302.04166
26 Bastian, F. B. *et al.* Bgee in 2024: focus on curated single-cell RNA-seq datasets, and query tools. *Nucleic Acids Res* **53**, D878-D885 (2025). https://doi.org:10.1093/nar/gkae1118

REVIEWER COMMENTS

Reviewer #1 (Remarks to the Author):

The authors thoroughly addressed our comments. The manuscript has been substantially improved through clarifications, additional analyses, and expanded documentation. The authors have addressed several important concerns regarding data integrity, technical claims, and figure clarity.
However, two major points remain to be addressed to further strengthen the manuscript: the lack of an internal ablation study and the absence of quantitative benchmarking for the gene annotation module.

Thank you for providing additional comments and clarifications. We have now provided additional analysis to address your concerns.

Major Comments

1. Regarding the authors' response to Major comments 1 and 2 (Source of Performance Improvement; Impact of Dataset Differences on Results)

Thank you for clarifying the differences in research objectives and for making updates to the manuscript. To further strengthen the credibility of your results, we still recommend including some form of ablation study or component analysis within the framework of your current work. For example:
1) Evaluate the contribution of key components in AnnDictionary to compare the performance when using AnnDictionary with and without specific core modules.
Even without direct comparison to prior work, such an internal analysis would better demonstrate which aspects of your system contribute to the observed results, and provide clearer insight into the source of performance improvements.

Thank you for your comments. We have now included a systematic ablation study of the core annotation function of AnnDictionary. The results show that while the annotation performance is robust to several ablations, including detuning of the base prompt, randomization of gene order, and removal of the system prompt, the ablation of tissue context resulted in a moderate decline in performance. We thus conclude that the tissue context is an important piece of information when using LLMs to annotate cell type. We have amended the manuscript and code base accordingly. The full source code to run the prompt ablation pipeline is available in the benchmark_llms repository under the branch ablate.

In addition, the results now read:

Prompt ablation study

The unablated prompt of the annotation function contained the following key components: 1) a system prompt designed to decrease output token usage and set context 2) a base prompt designed to decrease output token usage and make the LLM return only a cell type label 3) tissue context intended to provide a weak suggestion of expected labels 4) marker genes sorted as returned by scanpy's rank_gene_groups function. To understand which components of the cell type annotation module impact annotation performance, we conducted a prompt ablation study using Claude 3.5 Sonnet, **Supplementary Table 7**. Detuning the base prompt caused annotations to be too long to use in the automated benchmarking pipeline, and so this ablation was removed from downstream comparisons. The performances amongst the remaining ablations seemed to be similar given their means and standard deviations across five replicates.

Standard deviations of metrics in the context of ablations were generally larger than in previous runs without ablation. Therefore, a dominating effect of the ablations was to decrease overall stability of the annotation pipeline. Specifically, we noticed a sharp increase in frequency of sporadic cell subtyping compared to that observed during testing of the unablated pipeline. Furthermore, without tissue context, the LLM tended to include spurious tissue information or cell types that would not reasonably be expected in the given sample. Being more stringent measures of agreement, the percent of labels that were rated as perfect matches, as well the percent of labels that were exact string matches, were more sensitive to the ablations overall than the less stringent binary rating.

The Methods now reads:

Prompt ablation study

We generated a set of ablated functions by taking the core annotation function (ai_cell_type) and independently ablating each component of the prompt. The first ablation was to detune the base prompt from, "In a few words and without restating any part of the question, describe the single most likely cell type represented by the marker genes:" to, "What cell type is represented by the marker genes:". The second ablation was to remove the tissue context from the prompt. The third ablation was to remove the system prompt, "You are a terse molecular biologist." The fourth and final ablation was to randomize the gene order in the gene list input. To test how these ablations impacted the agreement of LLM-generated annotations and manual annotations, we used claude-3-5-sonnet-20240620 on the full Tabula Sapiens v2. We followed the same data preprocessing and filtering steps as used in the main analysis, as well as the same label comparison procedure. The source code for the modifications used in the prompt ablation study are available in the benchmark_llms GitHub repository under the branch ablate.

And we now include **Supplementary Table 7:**

Supplementary Table 7. LLM performance (Claude 3.5 Sonnet) during prompt ablation.

	Binary (%)		Perfect Match (%)		Extract String Match (%)		Kappa
	Cells	By Cell Type	Cells	By Cell Type	Cells	By Cell Type	With Manual
Unablated	78 ± 4	66.1 ± 3.4	47 ± 17	30 ± 15	41 ± 21	26 ± 19	0.39 ± 0.21
Base prompt detuned	—	—	—	—	—	—	—
No tissue context	81.5 ± 2.2	59.3 ± 3.3	46 ± 14	21 ± 14	44 ± 16	20 ± 15	0.42 ± 0.16
No system prompt	78 ± 5	61 ± 7	42 ± 24	30 ± 17	38 ± 28	25 ± 21	0.36 ± 0.28
Randomize gene list order	80.2 ± 2.3	63 ± 4	48 ± 14	28 ± 15	42 ± 19	24 ± 18	0.40 ± 0.19

2. Regarding the authors' response to Major comments 3 (Comparison of Gene Annotation Methods)

Thank you for the additional effort and for providing documentation and example code for the gene annotation module. However, from the perspective of evaluating AnnDictionary as a scientific tool, it is important to rigorously assess the performance of each functional component, so that researchers can properly evaluate its reliability and usability in real applications.

While the examples help illustrate how the function works, they do not fundamentally answer the question of whether this module is reliable. A few anecdotal examples are insufficient to demonstrate performance. Instead, some form of quantitative evaluation or benchmarking, even if limited in scope, is necessary to support the validity of this feature.

We encourage the authors to include a minimal benchmark—e.g., comparing a small set of gene sets annotated by AnnDictionary to annotations from established tools—to provide at least preliminary evidence for the effectiveness of the method.

Thank you for your comments. In this context, we agree that it could be useful to assess the reliability of the biological process annotation module. The capability of LLMs at annotating gene sets for biological process has previously been studied in detail by Hu et al.[5] We therefore used their previously established methodology to show the satisfactory performance of AnnDictionary's biological process annotation function. Moreover, most of the LLMs in the present study have not been benchmarked at biological process annotation. So, we included these models in the minimal benchmarking study. In this benchmarking analysis, we used LLMs to annotate gene lists of known biological process retrieved from the Gene Ontology Biological Process database, and rate close matches of LLM labels to the GO term label based on cosine similarity of text embeddings of the labels. An LLM annotation of a gene list is considered a

close semantic match to the source GO term if the source GO term's embedding is closer to the LLM annotation's embedding than 95% of other GO term embeddings. Using a random sample of 500 gene sets from GO Terms with between 3 and 100 genes, Claude 3.5 Sonnet had the highest proportion of close matches to source GO terms (81.20 ± 0.32%), followed by Llama 3.1 405B Instruct (71.9 ± 0.5%), and Claude 3 Opus (71.0 ± 0.4%), **Table 1**. We have added this analysis to the manuscript, and added the source code to the benchmark_llms code repository.

The Abstract now reads:

> We developed an open-source package called AnnDictionary (https://github.com/ggit12/anndictionary/) to facilitate the parallel, independent analysis of multiple anndata. AnnDictionary is built on top of LangChain and AnnData and supports all common large language model (LLM) providers. AnnDictionary only requires 1 line of code to configure or switch the LLM backend and it contains numerous multithreading optimizations to support the analysis of many anndata and large anndata. We used AnnDictionary to perform the first benchmarking study of all major LLMs at de novo cell-type annotation. LLMs varied greatly in absolute agreement with manual annotation based on model size. Inter-LLM agreement also varied with model size. We find that LLM annotation of most major cell types to be more than 80-90% accurate, and will maintain a leaderboard of LLM cell type annotation at https://singlecellgpt.com/celltype-annotation-leaderboard. Furthermore, we benchmarked these LLMs at functional annotation of gene sets, and found that Claude 3.5 Sonnet recovers close matches of functional gene set annotations in over 80% of test sets.

The Introduction now reads:

> We also benchmarked these LLMs at functional annotation of gene sets.

The Results now read:

> *Benchmarking biological process annotation*
>
> To assess the performance of the LLM biological process annotation function in AnnDictionary, we followed previous methodology outlined in Hu et al[5] to define close matches between LLM-generated annotations of gene sets and the Gene Ontology terms labels from which the genes were taken, see **Methods**. All but one of the LLMs in this study—GPT-4—have not been previously benchmarked at this task. Based on annotation of 500 gene sets derived from GO Biological Process terms, Claude 3.5 Sonnet achieved the highest proportion of close matches to source GO terms (81.20 ± 0.32%), followed by Llama 3.1 405B Instruct (71.9 ± 0.5%), and Claude 3 Opus (71.0 ± 0.4%), **Table 1**.
>
> To demonstrate the biological process annotation function, we present sample LLM-annotations of gene lists of known processes from the Gene Ontology Biological Process database. In these three cases, the LLM annotations, while slightly broader, generally agreed with the existing labels, **Supplementary Table 8**.

The Discussion now reads:

Discussion

This study represents the first comprehensive benchmark of LLMs at de novo cell type annotation, and we plan to maintain a leaderboard of LLMs at this task as measured on Tabula Sapiens. We also present the first benchmarks of 14 LLMs at biological process annotation from gene sets of known process. Overall, our measures of performance indicate that large LLMs can provide reliable de novo cell type annotations at the broad cell type level, and reliable biological process annotation.

…

In addition to comprehensive cell type annotation benchmarking, we also include biological inference functions (e.g. functional gene set annotation with biological processes) and associated benchmarking. Most of the LLMs used in this study have not been previously benchmarked at this task, with GPT-4 being the one exception. We demonstrate that some LLMs have substantially higher performance compared to the best performances previously observed with other LLMs. Specifically, we observed Claude 3.5 Sonnet to achieve just over an 80% close semantic match rate when annotating curated gene sets, whereas the previous best performance was GPT-4, with a roughly 60% close semantic match rate. It is convenient to consolidate all these annotation tasks into one package.

Limitations

…

With regard to the benchmarking of LLMs at biological process annotation, a key limitation is that the benchmarking was performed on curated gene sets derived directly from GO terms. These lists likely differ from experimentally derived gene lists, and so further evaluation could be investigated. Difficulty in further evaluation includes establishing ground truth interpretations of gene lists, as genes are often used in many potentially independent contexts.

The Methods now read:

Benchmarking biological process annotation

To benchmark LLMs at biological process annotation, we followed the procedure designed by Hu et al.[5] We used gseapy v1.1.8 to access the 2023 release of the Gene Ontology Biological Process (GOBP) database (n=5,406 human terms). To generate gene lists of known biological process, we randomly selected 500 GOBP terms from the set of GOBP terms with between 3 and 100 genes (n=5,094). Each of these 500 gene lists were annotated by LLMs using the ai_biological_process function from AnnDictionary. To measure whether each LLM annotation was a close semantic match to the gene list's associated GOBP term, we computed text embeddings of all LLM

generated annotations and all 5,406 GOBP terms, and used the cosine similarity to measure the similarity between the embeddings of each LLM annotation and all GOBP terms. An LLM annotation was considered a close match to the source GOBP term if the source GOBP term was in the 95th percentile or higher of GOBP terms by cosine similarity of the text embedding to the LLM annotation. Text embeddings were computed using OpenAI's text-embedding-3-large model. We ran 5 replicates of each model to ensure stability, and also tested several random seeds to ensure that observed performance was not due to the specific set of GOBP terms sampled. Source code to reproduce this benchmarking analysis is available in the benchmark_llms GitHub repository as a Jupyter notebook in the nbs directory.

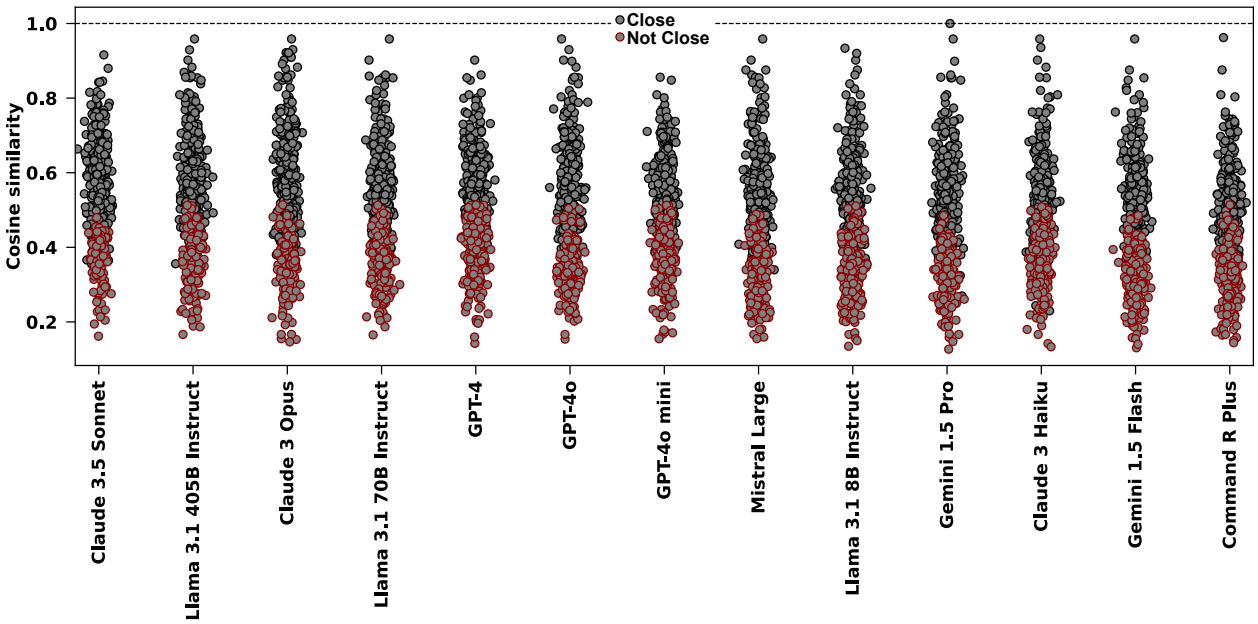
Then, to make a brief example, we retrieved the gene lists associated with the first three terms in the database, and used AnnDictionary's ai_biological_process function to label the gene sets with claude-3-5-sonnet-20240620.

We have added the results of the biological process benchmarking study to Table 1:

Table 1. LLM performance. Agreement with manual annotations measured by yes/no, quality of match, and exact string agreement. Kappa with manual annotation and average kappa of the given model with every other model. Biological process annotation of known gene lists. All values are mean ± standard deviation across five replicates.

	Cell Type								Biological Process
	Binary (%)		Perfect Match (%)		Extract String Match (%)		Kappa		Close Match
Model	Cells	By Cell Type	Cells	By Cell Type	Cells	By Cell Type	With Manual	Average With LLMs	% of Terms
Claude 3 Haiku	78.3 ± 0.8	66.2 ± 1.5	61.8 ± 2.3	47 ± 4	61.8 ± 2.3	47 ± 4	0.589 ± 0.023	0.652 ± 0.023	62.8 ± 0.4
Claude 3 Opus	82.3 ± 0.9	69.8 ± 1.0	72.8 ± 2.6	54 ± 4	72.7 ± 2.6	54 ± 4	0.704 ± 0.026	0.711 ± 0.020	71.0 ± 0.4
Claude 3.5 Sonnet	84.0 ± 0.7	70.5 ± 1.2	74.4 ± 2.7	54 ± 4	74.3 ± 2.6	53 ± 4	0.721 ± 0.027	0.697 ± 0.026	81.20 ± 0.32
Command R Plus	77.2 ± 1.0	59.4 ± 2.6	64.5 ± 2.6	40 ± 5	64.5 ± 2.6	40 ± 5	0.616 ± 0.027	0.646 ± 0.026	58.5 ± 0.7
GPT-4	79.2 ± 0.9	64.3 ± 1.9	64 ± 4	44 ± 5	64 ± 4	44 ± 5	0.61 ± 0.05	0.65 ± 0.04	65.24 ± 0.33
GPT-4o	80.9 ± 0.7	70.1 ± 2.8	70.4 ± 2.5	54 ± 6	70.4 ± 2.5	54 ± 6	0.680 ± 0.026	0.721 ± 0.021	67.04 ± 0.33
GPT-4o mini	76.8 ± 1.0	66.2 ± 1.6	63.4 ± 3.0	47 ± 6	63.4 ± 3.0	47 ± 6	0.605 ± 0.031	0.681 ± 0.022	64.8 ± 0.5
Gemini 1.5 Flash	68.8 ± 1.3	60.8 ± 2.7	51.0 ± 2.5	41 ± 5	51.0 ± 2.5	41 ± 5	0.478 ± 0.024	0.561 ± 0.020	60.52 ± 0.18
Gemini 1.5 Pro	77.5 ± 1.8	67.9 ± 0.8	65.1 ± 2.4	50 ± 5	65.1 ± 2.4	50 ± 5	0.625 ± 0.024	0.658 ± 0.019	66.32 ± 0.11
Llama 3.1 405B Instruct	82.0 ± 1.0	64.9 ± 2.7	69.5 ± 2.6	47 ± 5	69.3 ± 2.6	47 ± 5	0.667 ± 0.027	0.690 ± 0.021	71.9 ± 0.5
Llama 3.1 70B Instruct	74 ± 4	61.6 ± 1.5	64 ± 4	46 ± 5	64 ± 4	45 ± 5	0.62 ± 0.04	0.665 ± 0.022	70.8 ± 0.5
Llama 3.1 8B Instruct	59 ± 4	53 ± 4	47.7 ± 3.2	37 ± 6	47.6 ± 3.2	36 ± 6	0.440 ± 0.031	0.526 ± 0.030	61.7 ± 0.7
Mistral Large	78.1 ± 1.5	66.2 ± 2.0	64.9 ± 2.7	50 ± 5	64.8 ± 2.8	49 ± 5	0.623 ± 0.027	0.696 ± 0.024	62.76 ± 0.17
Plurality Vote	80.5 ± 0.9	69.4 ± 1.7	72.4 ± 2.1	55 ± 5	72.3 ± 2.1	55 ± 5	0.700 ± 0.022	0.770 ± 0.018	—

We also included Supplementary Figure 4, which shows an exemplar run of the distribution of cosine similarities of LLM annotations and the source GO term:



Supplementary Figure 4. Distribution of cosine similarities between Gene Ontology Biological Process term and LLM annotation of the genes in that term, colored by whether the LLM annotation was a close match to the source GO term as defined in the Methods, shown for one exemplar run.

We have added the source code to reproduce this analysis as a notebook in the benchmark_llms repository, available here: https://github.com/ggit12/benchmark_llms/nbs/benchmark_biological_process_annotation.ipynb.

3. Regarding the authors' response to Major comments 4

This point has been satisfactorily addressed. We appreciate the authors' thorough effort to assess and document the risk of data leakage and potential bias.

We thank the reviewer for their comments and feedback.

4. Regarding the authors' response to Major comments 5

This point has been satisfactorily addressed. This point has been substantially addressed. We appreciate the authors' detailed response.

We thank the reviewer for their comments and feedback.

5. Regarding the authors' response to Major comments 6

This point has been satisfactorily addressed. This point has been substantially addressed. We appreciate the authors' detailed response.

We thank the reviewer for their comments and feedback.

6. Regarding the authors' response to Major comments 7

This point has been satisfactorily addressed.

We thank the reviewer for their comments and feedback.

Minor Comments

All minor figure legend issues have been addressed. No further concerns.

We thank the reviewer for their comments and feedback.

Remarks on code availability

All issues have been addressed.

We thank the reviewer for their comments and feedback.

Reviewer #2 (Remarks to the Author):

The authors have addressed my comments.

We thank the reviewer for their comments and feedback.

Reviewer #3 (Remarks to the Author):

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

Reviewer #4 (Remarks to the Author):

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

Reviewer #5 (Remarks to the Author):

Thank you for your answers and the modifications applied in the manuscript. These are sufficiently addressing my comments. I am really impressed by the complete reorganization and updates of the code base which now looks appropriate and installs without issues.

We thank the reviewer for their comments and feedback.

Reviewer #5 (Remarks on code availability):

The code base looks good. There is an appropriate structure, a good amount of unit tests, and sufficient documentation. The notebooks are providing examples of use cases.
I think it would be a good idea to set the version number to 1.0.0 indicating that the code is stable.

Thank you for your comments. We are pleased to hear that the codebase has improved. We will consider incrementing the version number pending the conclusion of the review process and resolution of any related updates to the codebase.