Benchmarking Cell Type Annotation by Large Language Models with AnnDictionary

George Crowley[1], Tabula Sapiens Consortium[2], Stephen R. Quake[1,3,4]

1. Department of Bioengineering, Stanford University, Stanford, California, USA
2. Chan Zuckerberg BioHub, San Francisco, CA, USA
3. Department of Applied Physics, Stanford University, Stanford, CA, USA
4. Chan Zuckerberg Initiative, Redwood City, CA, USA

Abstract
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 Scanpy 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 benchmark all major LLMs at de novo cell-type annotation in Tabula Sapiens. 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%. 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.

Introduction

Single cell transcriptomic sequencing (scRNA-seq) analysis has enabled discovery of novel cancer targets, rare cell types, and deepened our understanding of cell phenotype and function. One of the largest bottlenecks in scRNA-seq is the annotation of cell type. Until recently, this step has required input from human experts. Large language models (LLMs) have emerged as a promising tool to automate single cell analysis based on marker genes.(*1*) What's more, LLMs have shown satisfactory agreement with classical biological inference tools (i.e. Gene Ontology term analysis), and so additionally hold promise for automating interpretation downstream of cell type annotation.(*2*)

LLMs are primarily accessed through a commercial provider via a provider-specific interface. While some are open source and can be downloaded for local use, the size and complexity of doing so can be restrictive. Therefore, we built AnnDictionary (https://github.com/ggit12/anndictionary/), an LLM-provider-agnostic python package built on top of Scanpy and LangChain that can be "plugged in" to any available LLM by just changing one line of code (e.g. any model provided by OpenAI, Anthropic, Google, Meta, or available on Amazon Bedrock). The aim of this package is to consolidate both automated cell type annotation and biological process inference into a single python package that interfaced natively with Scanpy. Furthermore, while handling smaller datasets with ease, AnnDictionary includes optimizations to allow the LLM-based annotation of atlas-scale data. We therefore used this package to benchmark the ability of cell type annotation of all 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). These benchmarks will be displayed as a running leaderboard at https://singlecellgpt.com/celltype-annotation-leaderboard.

Additionally, as scRNA-seq experiments continuously increase in size and complexity, so do their analyses. For example, it can be desirable to handle a dataset stratified at the donor, tissue, or cell type level, and independently operate on each of these groups of the data (i.e. normalize each tissue in Tabula Sapiens independently). AnnDictionary aims to create a formal backend for independent processing of multiple anndata in parallel. This functionality is a key building block in various scRNA-seq and spatial transcriptomic tasks (e.g. de novo annotation of cell type, label transfer, data integration, cell segmentation), and we expect AnnDictionary to serve as a useful, flexible backend for their analyses. We therefore provide notebooks of how AnnDictionary can simplify implementation of these tasks.

Results

AnnDictionary is a parallel backend for processing anndata

Our first step in benchmarking LLMs involved building a backend that could handle the parallel processing of many anndata through a simplified interface. The current state of the art is to manually create a dictionary of anndata objects and loop over them. We aimed to formalize this concept by: 1) defining a class called AnnDictionary (i.e. a dictionary of anndatas); and 2) providing an essential family of workhorse functions—adata_dict_fapply—that operates conceptually similar to R's lapply() or python's map(), **Figure 1**. The adata_dict_fapply is multithreaded by design, and is what allows the atlas-scale annotation of tissue-cell types by 15 LLMs in a tractable amount of time). However, multithreading can also be turned off for non-thread-safe operations.

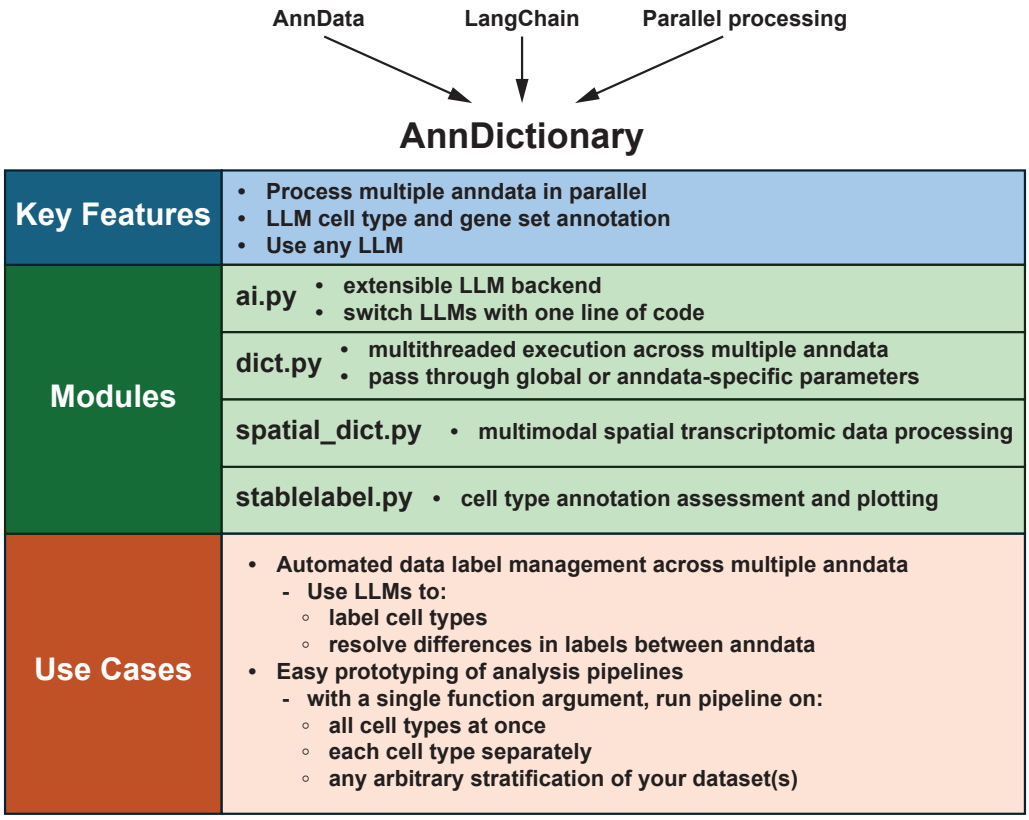

Figure 1. Overview of AnnDictionary—a python package built on top of LangChain and AnnData, with the goal of independently processing multiple anndata in parallel.

In AnnDictionary, we also include a module of wrappers to common Scanpy functions. Furthermore, we provide centralized wrappers for label transfer pipelines (i.e. via logistic regression with Universal Cell Embedding), and data integration techniques (i.e. harmony), and for common spatial transcriptomic libraries (i.e. squidpy). While this is only a small sample of available methods, AnnDictionary is extensible and can grow to accommodate additional methods. Finally, AnnDictionary functions can typically take single arguments to be broadcast to all anndata, or arguments can be provided as a dictionary, with a separate parameter for each anndata. For example, one could normalize 3 different datasets to 3 different values with a single AnnDictionary function call.

AnnDictionary consolidates common LLM integrations under one roof

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.

Cell type annotation

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. We note that current LLMs do not seem good enough to reliably produce reasonable resolutions, but still may be a useful first pass. Then, we provide several functions for cell type annotation via different methods. The following methods can generally be tissue-aware at the user's discretion. These include 1) based on a single list of marker genes, 2) by comparing several lists of marker genes, 3) by attempting to derive cell subtypes by comparing several lists of marker genes. To simplify the process of label management (for example, when resolving syntactic differences in labels used across different studies, or for resolving differences in labels produced by different researchers or LLMs), we provide several functions that use LLMs to unify labels between anndata and within an anndata. These functions can also be used to create label hierarchies (i.e. from cell subtype all the way to compartment). Furthermore, we provide ways to assess label agreement (all using LLMs to manage label comparison). At this stage, we note that, as a design principle, AnnDictionary returns relevant LLM output so that the user can manually verify cell type mappings, annotation, etc.

Gene annotation

We wrote several functions to assist with gene processing. These include annotating sets of genes and adding these annotations to the metadata, for example if an "is heat shock protein" column in the gene metadata is desired. We also wrote functions that use an LLM to attempt to infer the biological process represented by a list of genes.

AnnDictionary can plug in to any LLM with a single line of code

With a parallel processing backend and LLM integrations in place, the next step was to create a simple interface to allow the use of any LLM with AnnDictionary. This flexible design was desired to allow ease of use and future-proofing as new LLMs become available. To accomplish this, we built on top of LangChain to design a configurable LLM backend that we could call from the LLM integration functions without reference to a specific underlying model. The result is that the functions in AnnDictionary can be used with any LLM with just a single line of code—a function called "configure_llm_backend".

This aspect of flexibility and incorporation of provider-specific handling, including rate limits and message formatting, enables AnnDictionary to be used to annotate Tabula Sapiens v2 with 15 different LLMs.

Claude 3.5 Sonnet had the highest agreement with manual annotation

Tabula Sapiens v2 contains over 1.2 million cells with human expert annotations, more than half of which have not yet been publicly released (this is relevant for LLM annotation, because LLMs are trained on published marker gene data). To perform the de novo LLM-based cell type annotation of the entire atlas, we first applied standard scRNA-seq analysis on a *per-tissue* basis. Starting from raw counts, we 1) normalized, 2) log-transformed, 3) set high variance genes (top 2000 genes), 4) scaled, 5) PCA, 6) calculated neighborhood graph 7) calculated UMAP. Next, we used our agent-based function to derive a first pass of appropriate cluster resolutions (Leiden) for each tissue, and manually corrected instances

where the LLM agent failed. This step was done using GPT-4o, as not all LLMs allow images as input, and rate limits for other APIs were restrictive.

Using the manually corrected cluster resolutions, we calculated differentially expressed genes for each cluster (independently for each tissue). We then labelled cell types with a two-step approach. First, clusters were independently annotated with an LLM based on the top 10 marker genes.(*1*) Second, labels were merged with an LLM to account for redundant labels (i.e. "Macrophage", "macrophage", and "macrophage.") and sporadic cell subtyping (i.e. "T cell" and "Cytotoxic T cell"). We performed correction for subtyping because we noticed that, during package development, broad cell type labels were easy to consistently elicit, but more specific labels were not consistent. We therefore performed agreement assessments at this coarser cell type level. This may not speak to LLM performance specifically, and could be due to conflicting or undecided literature on cell subtypes in general.

We performed the two-step annotation process with each of the LLMs listed in **Table 1**.

Table 1. List of LLMs studied.

Company	Provider	Endpoint
Meta	Bedrock	meta.llama3-1-8b-instruct-v1:0
Meta	Bedrock	meta.llama3-1-70b-instruct-v1:0
Meta	Bedrock	meta.llama3-1-405b-instruct-v1:0
Amazon	Bedrock	amazon.titan-text-express-v1
Amazon	Bedrock	amazon.titan-text-lite-v1
Cohere	Bedrock	cohere.command-r-plus-v1:0
Mistral	Bedrock	mistral.mistral-large-2407-v1:0
Google	Google	gemini-1.5-pro
Google	Google	gemini-1.5-flash
OpenAI	OpenAI	gpt-4
OpenAI	OpenAI	gpt-4o
OpenAI	OpenAI	gpt-4o-mini
Anthropic	Anthropic	claude-3-5-sonnet-20240620
Anthropic	Anthropic	claude-3-opus-20240229
Anthropic	Anthropic	claude-3-haiku-20240307

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. We opted to use Claude 3.5 Sonnet for downstream comparison because, during testing, we observed that it had the best reasoning capabilities and sufficiently high API rate limits. As a first step, we used an LLM (Claude 3.5 Sonnet) to create a unified set of cell type labels across all annotations (manual and LLM-based). We then assessed label agreement in two ways, by comparing LLMs to 1) manual annotation 2) each other.

Agreement with Manual Annotation

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). Using the manual annotations as the grouping variable, we then 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,

Figure S1. 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 2**, and visualized the rates of all match qualities, **Figure 2A**, **B**.

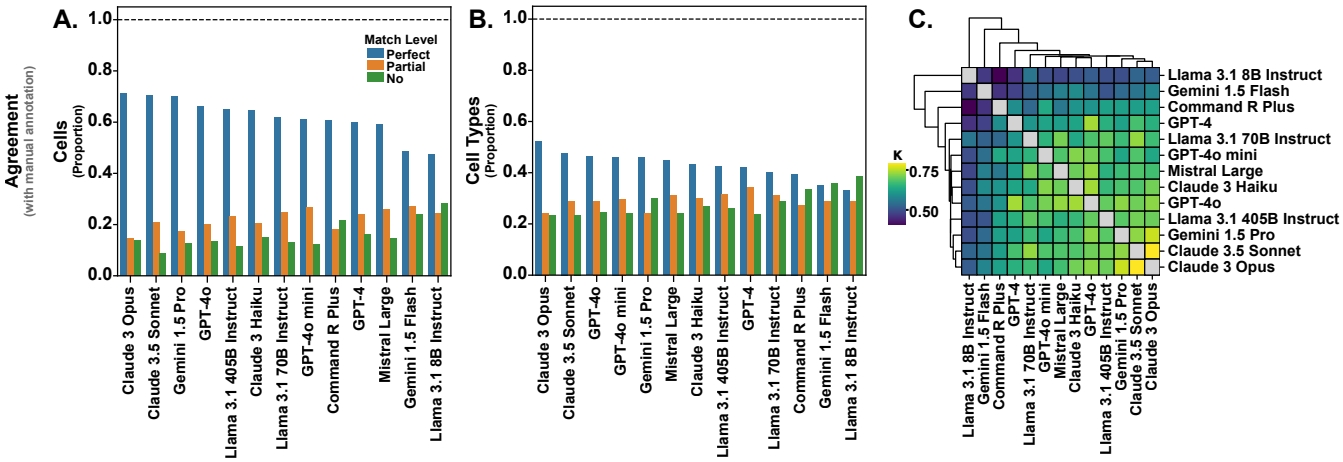

Figure 2. LLM cell type annotation quality as compared to manual annotation, rated by an LLM at three levels: 1) perfect, 2) partial, and 3) non-matching, and two resolutions: **A.** cells and **B.** cell types. **C.** Inter-rater reliability measured as pairwise kappa between each LLM.

Table 2. LLM performance. Agreement with manual annotations measured by yes/no, and quality of match. Kappa with manual annotation and average kappa of the given model with every other model.

	Binary (%)		Perfect Match (%)		Kappa	
Model	**Cells**	**Cell Types**	**Cells**	**Cell Types**	**With Manual**	**Average with LLMs**
Claude 3.5 Sonnet	81.14	65.03	70.29	47.74	0.679	0.675
Gemini 1.5 Pro	79.88	63.68	70.21	45.97	0.679	0.653
Claude 3 Opus	79.01	65.22	71.15	52.36	0.689	0.670
GPT-4o	77.58	63.52	66.33	46.35	0.639	0.678
GPT-4	76.94	65.32	59.97	41.98	0.570	0.614
Llama 3.1 405B Instruct	75.69	61.51	65.02	42.39	0.622	0.648
Claude 3 Haiku	73.41	59.48	64.50	43.16	0.620	0.656
Llama 3.1 70B Instruct	72.46	54.32	62.04	40.00	0.594	0.640
Mistral Large	72.15	62.18	59.21	44.65	0.567	0.646
GPT-4o mini	71.70	61.53	60.98	46.14	0.582	0.643
Command R Plus	70.14	56.34	60.54	39.47	0.577	0.575
Gemini 1.5 Flash	62.44	51.98	48.80	35.25	0.460	0.532
Llama 3.1 8B Instruct	55.34	46.35	47.45	32.97	0.441	0.496

Claude 3.5 Sonnet had the highest binary agreement with manual annotations at 81.14% of cells, followed closely by Gemini 1.5 Pro, Claude 3 Opus, and GPT-4o. Claude 3 Opus had the highest binary agreement by cell type at 65.22% of cell types, and the highest proportion of perfect matches at 71.15% of cells and 52.36% of cell types. As expected, the lowest performing models were the lightweight

models with smaller numbers of parameters. We omit Amazon's Titan models from assessment because they could not reliably follow directions well enough to annotate cell types.

Inter-LLM Agreement

The second way we assessed LLM annotation was via consistency between the LLMs. First, we measured kappa between each LLM and the manual annotation, **Table 2**. Of the LLMs tested, Claude 3 Opus was the most consistent with manual annotation (κ = 0.689). Then, we measured kappa values pairwise between all LLM annotation columns, **Figure 2C**, and each model's average kappa with all other models, **Table 2**. On average, GPT-4o was the most consistent with all other LLMs (κ = 0.678), and Claude 3 Opus and Claude 3.5 Sonnet were the most consistent LLM pair (κ = 0.791).

LLMs excel at annotating major cell types

Overall, there is roughly a 15-20% performance difference between annotating cells vs. cell type, indicating that the models consistently agree with large, common cell types. Among the 10 largest cell types, LLMs consistently scored highly (>80-90%), except for Stromal Cells and Basal Cells, **Figure 3A**. We then looked at how the best-performing LLMs annotated these cell types. Cells that were manually annotated as basal cells were, in large part, annotated as epithelial cells by the top performing LLM (Claude 3.5 Sonnet), **Figure 3B**. Basal and epithelial cells are closely related in lineage. Based on a small number of canonical marker genes (CDH1, EPCAM, and KRT8 for epithelial cells, and KRT5, KRT14, and TP63 for stromal cells), it seems that, while there may be a subpopulation of manually annotated basal cells that have a more epithelial phenotype, basal cells dominate this group **Figure 3C**. Meanwhile, stromal cells came entirely from the ovaries, and the LLMs derived cell type names for subclusters of this population, **Figure 3D**. However, known marker genes (specifically DCN and LUM) were expressed broadly across cells manually annotated as stromal cells, **Figure 3E**.(*3*) Additionally, it appeared that cells that were manually annotated as Neutrophils were consistently labeled as Macrophages by LLMs, **Figure S2**.

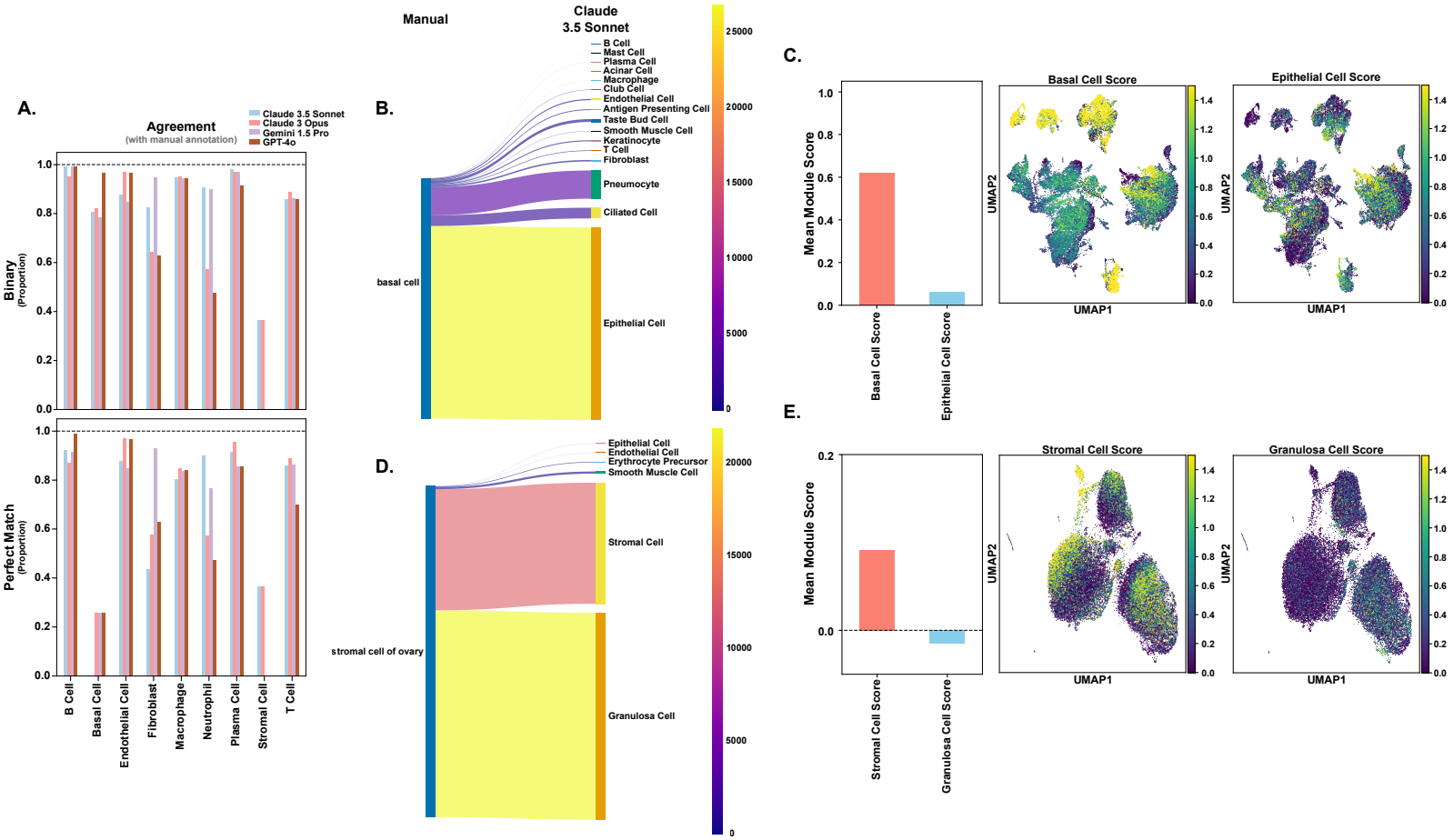

Figure 3. **A.** Agreement with manual annotation of top performing LLMs for the ten largest cell types by population size in Tabula Sapiens v2. As in **Figure 2**, agreement was assessed at two levels: binary (yes/no, top) and perfect match (bottom). For the two large cell types that disagreed with manual annotation the most: LLM annotations for cells manually annotated as **B.** basal cells and **D**. stromal cells (of ovary); and gene module scores for marker genes of the manually annotated cell type vs. marker genes for the mode LLM annotation **C.** Basal cell and Epithelial cell scores. **D.** Stromal Cell and Granulosa Cell scores.

To further understand LLM annotation behavior across all cell types, we plotted inter-LLM agreement vs. agreement with manual annotation for each cell type. Among other uses, this plot was designed to allow us to identify cell types that were consistently rated by the LLMs but disagreed with the manual annotations, **Figure 4**. We separate this plot into 4 quadrants, with cell types in the 1) top-left: LLMs agree with each other, but disagree with the manual annotation; 2) bottom-left: LLMs disagree with each other and with the manual annotation; 3) bottom-right: LLMs disagree with each other but agree with the manual annotation; and 4) top-right: LLMs agree with each other and with the manual annotation. This plot is designed to qualitatively assess label confidence.

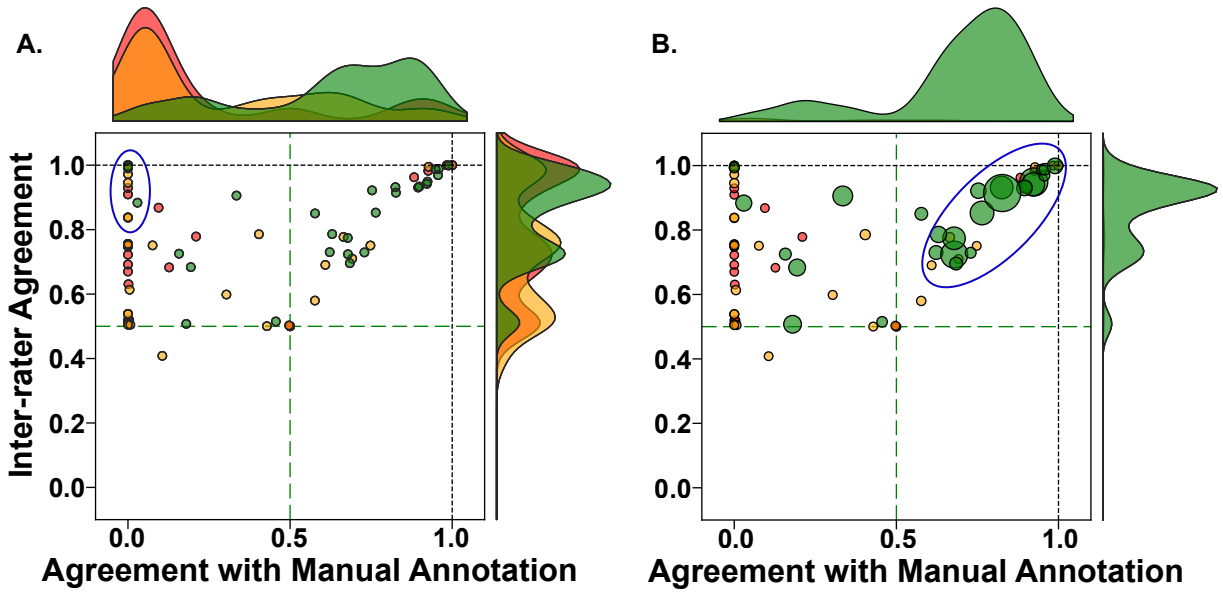

Figure 4. A. Inter-rater agreement within the top 4 performing LLMs vs. agreement with manual annotation for each manual cell type annotation, with marginal kernel density estimates stratified by tertile of cell type population size. Red, yellow, and green represent the bottom, middle, and top tertile 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 circle two groups of interest: (A) the cell types with the highest inter-rater agreement and lowest agreement with manual annotation, and (B) the cell types with the highest inter-rate agreement and highest agreement with manual annotation.

Nearly all cell types had greater than 50% agreement between the LLMs, suggesting that the rate of completely spurious annotation by these LLMs is generally low across the atlas. To see how cell type population size may have affected annotation consistency, we divided cell types into tertiles of population size. Generally, major cell types had high agreement both between LLMs and with respect to manual annotation, while smaller cell types still had moderate (>50%) agreement between LLMs, but did not agree with the manual annotation. To see where the majority of cells lie, we weighted the density estimates by the number of cells in each cell type, **Figure 4B**. Here, it is clear that the majority of cells are consistently rated by the LLMs and agree with manual annotation. The differences from this trend were primarily basal and stromal cells. As previously discussed, these cells were labeled less consistently by the LLMs than most other major cell type annotations

We then investigated the group of (mostly) smaller cell types in the upper-left of this scatterplot, **Figure 4A**. For the 10 cell types that were closest to the top-left corner of the scatterplot, we plotted a confusion matrix to see the correspondence of manual annotations with annotations by the top-performing LLM (Claude 3.5 Sonnet), **Figure 5A**. The largest of these cell types were manually annotated as mononuclear phagocytes (n~5,000), and LLM-annotated as macrophages. Based on visualization of canonical marker genes and associated module scores, there is evidence that this cluster is mostly composed of macrophages, but also contains monocytes and dendritic cells, **Figure 5B, C**. We note that there are other cell populations in the stomach that are manually labelled as monocytes and macrophages. Taken together, these data suggest that, in this case, the manual annotations may be technically correct, but the LLM annotation may be more pragmatic, as this cluster overall has highest expression of macrophage marker genes. That is to say that the label “mononuclear phagocyte” is a

useful description of phenotype across several cell types, but ultimately represents a different depth of annotation than other labels in the same set. Finally, we might expect to record disagreement for small cell types whose clusters were not reproduced in the present study due to parameter decisions.

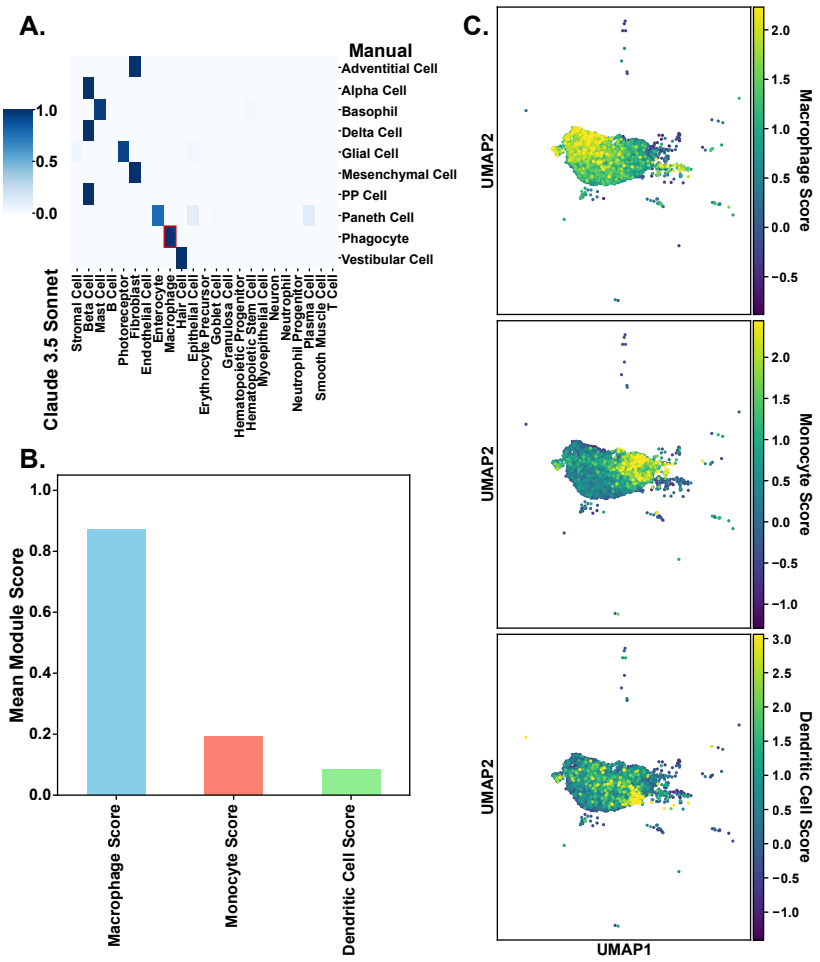

Figure 5. A. For the 10 cell types closest to the top-left corner of scatterplot in **Figure 4A,** a confusion matrix of top performing LLM annotations and corresponding manual annotations, with a red box around the largest cell type present in this group (phagocytes). **B.** Macrophage, monocyte, and dendritic cell module scores derived using canonical marker genes for cells manually annotated as phagocytes. **C.** UMAP visualization of the module scores in (B).

Discussion

This study represents the first comprehensive benchmark of LLMs at cell type annotation, and we plan to maintain a leaderboard of LLMs at this task as measured on Tabula Sapiens. Previous work has assessed GPT-4's capability using Tabula Sapiens as a benchmark dataset, and found 58% agreement with manual annotation.(*1*) On this larger version of Tabula Sapiens, we report 76.94% agreement of GPT-4 with manual annotation, and even higher agreement for several models, including GPT-4o, Claude 3.5 Sonnet, Claude 3 Opus, and Gemini 1.5 Pro. Our improvements could be due to several factors, including improvements of the models themselves, in the prompt engineering in AnnDictionary, or in data pre-

processing pipelines. It could also be due to increased permissiveness in terms of what counts as agreement between labels.

Beyond saving time, effort, and cost, one major advantage of using LLMs to annotate cell types is that the LLMs seem to be able to annotate at a more consistent depth than achieved manually. However, large-scale cell type annotation with LLMs highlighted the potential pitfalls of cell type annotation in general. The apparent shortfall of LLM annotation for cell types such as basal cells and mononuclear phagocytes, may actually represent an artifact of dichotomizing continuous expressional gradients in transcriptomic data, and not speak to the performance of LLMs themselves. This is supported by the fact that the LLM annotations for these cells were, for the most part *nearly* correct, representing nearby points in the lineage of each of these cell types (i.e. a large portion of basal cells were annotated as epithelial cells and mononuclear phagocytes were annotated as macrophages). In contrast, the case of stromal cell subcluster annotation by LLMs could be an artefact of potential over-clustering in the preprocessing pipeline used in the present study, or limitations of the marker gene selection method. We also observed the difficulty inherent in distinguishing immune cell types, such as neutrophils, macrophages, and monocytes based on a small number of differentially expressed marker genes.

In the present study, we opted not to assess intra-LLM kappa (i.e. a model's consistency with itself upon repeated, independent prompts), because we believe this to be a trivial assessment of the temperature parameter of the model. The temperature parameter is a feature of many LLMs and controls the "determinism" of the responses.

In addition to comprehensive cell type annotation benchmarking, we also include biological inference functions (e.g. gene set annotation). While we do not benchmark them here, previously published work has established this capability in LLMs.(*2*) It is convenient to consolidate these tasks into one package.

Conclusion

In summary, we developed a parallel backend that simplifies processing of several anndata at once. This package is flexible to allow users to build their own additions. We also have wrapped LLM-backend configuration and switching into a single line of code, thereby simplifying the use of LLMs for annotation tasks. Beyond the benchmarking of LLMs at cell type annotation based on marker genes described here, we plan to maintain an LLM leaderboard of this task at https://singlecellgpt.com/celltype-annotation-leaderboard.

Acknowledgements
We thank Jaeyoon Lee for his insights and feedback on the python package and the manuscript. We thank Mira N. Moufarrej for her feedback on the python package. We thank Madhav Mantri for his discussion of the manuscript. We thank Robert C. Jones and Jaeyoon Lee for help accessing the Tabula Sapiens v2 dataset.

References

1. W. Hou, Z. Ji, Assessing GPT-4 for cell type annotation in single-cell RNA-seq analysis. *Nat Methods*, (2024).
2. M. Hu *et al.*, Evaluation of large language models for discovery of gene set function. *ArXiv*, (2024).
3. X. Fan *et al.*, Single-cell reconstruction of follicular remodeling in the human adult ovary. *Nat Commun* **10**, 3164 (2019).