

GPTCelltype: Reference-free and cost-effective automated cell type annotation with GPT-4 in single-cell RNA-seq analysis

Wenpin Hou^{1,*}, Zhicheng Ji^{2,*}

¹Department of Biostatistics, Columbia University Mailman School of Public Health Health

²Department of Biostatistics and Bioinformatics, Duke University School of Medicine

* corresponding authors

Introductions

Cell type annotation is an essential step in single-cell RNA-seq analysis. However, it is a time-consuming process that often requires expertise in collecting canonical marker genes and manually annotating cell types. Automated cell type annotation methods typically require the acquisition of high-quality reference datasets and the development of additional pipelines. We demonstrated that GPT-4, a highly potent large language model, can automatically and accurately annotate cell types by utilizing marker gene information generated from standard single-cell RNA-seq analysis pipelines in this manuscript. We developed this software, **GPTCelltype**, to provide reference-free, cost-effective automated cell type annotation using GPT-4 for single-cell RNA-seq analysis.

Installation

GPTCelltype can be installed by following the instruction on Github.

```
remotes::install_github("Winnie09/GPTCelltype")
```

GPTCelltype depends on the R package `openai`. Please install and load `openai` before running `GPTCelltype`.

OpenAI Key

GPTCelltype integrates the OpenAI API into the software. To connect to OpenAI API, a secret API key is required. You can generate your API key in your OpenAI account webpage: log in to OpenAI, click on “Personal” on the upper right corner, click on “View API keys” in the break-down list, and then click on “Create new secret key” which directs you API key page. Copy the key and paste it on a note for further use. Users need to pass their secret API key to GPTCelltype functions as one of the inputs.

Run GPTCelltype

First of all, please load the packages.

```
library(GPTCelltype)
library(openai)
```

We demonstrate how to run GPTCelltype as follows. The main function is `gptcelltype()`. It can annotate cell types by OpenAI GPT models in a Seurat pipeline or with a custom gene list. If `gptcelltype()` is used in a Seurat pipeline, Seurat `FindAllMarkers()` function needs to be run first and the differential gene table generated by Seurat will serve as the input. If the input is a custom list of genes, one cell type is identified for each element in the list.

Among the input arguments, `input` can either be the differential gene table returned by Seurat `FindAllMarkers()` function, or a list of genes. `tissuename` (optional) is a tissue name. `openai_key` is your

OpenAI key obtained from API key page (see above section). **model** is a valid GPT-4 or GPT-3.5 model name listed on Models page. Default is 'gpt-4'. **topgenenumber** is the number of top differential genes to be used when the input is Seurat differential genes. The output is a vector of cell types.

Example 1: Seurat object as input You have to install Seurat package first to run this example

```
library(Seurat, quietly = TRUE)
```

```
## Attaching SeuratObject
```

We use the embedded data pbmc_small from Seurat as an example.

```
data("pbmc_small")
all.markers <- FindAllMarkers(object = pbmc_small)
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

```
gptcelltype(all.markers,
             tissuename = 'human PBMC',
             openai_key = yourkey, ## Note: Please use your OpenAI key.
             model = 'gpt-4'
)
```

```
##              0              1              2
## "Monocytes, Macrophages"      "Neutrophils"      "B Cells"
```

Then we can obtain the output “Monocytes, Dendritic cells” “Monocytes” “B cells”.

Example 2: a list of genes as input For example, if we provide a list of two gene vectors: the first one contains *CD4* and *CD3D*, and the second one contains *CD14*, then we can call the function in this way:

```
gptcelltype(
  input = list(cluster1 = c('CD4, CD3D'), cluster2 = 'CD14'),
  tissuename = 'human PBMC',
  openai_key = yourkey, ## Note: Please use your OpenAI key.
  model = 'gpt-4'
)
```

```
##      cluster1      cluster2
## "T helper cells"      "Monocytes"
```

Then we can obtain the output “T helper cells” “Monocytes”.

Session Info

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS 10.16
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
```

```

## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] SeuratObject_4.1.3 Seurat_4.3.0.1      openai_0.4.1      GPTCelltype_1.0
##
## loaded via a namespace (and not attached):
## [1] Rtsne_0.16           colorspace_2.1-0      deldir_1.0-9
## [4] ellipsis_0.3.2       ggribges_0.5.4        rstudioapi_0.14
## [7] spatstat.data_3.0-1  leiden_0.4.3          listenv_0.9.0
## [10] ggrepel_0.9.3        fansi_1.0.4           codetools_0.2-19
## [13] splines_4.0.2        knitr_1.42            polyclip_1.10-4
## [16] jsonlite_1.8.4       ica_1.0-3             cluster_2.1.4
## [19] png_0.1-8           uwot_0.1.14          shiny_1.7.4
## [22] sctransform_0.3.5    spatstat.sparse_3.0-1 compiler_4.0.2
## [25] httr_1.4.6          assertthat_0.2.1      Matrix_1.5-4.1
## [28] fastmap_1.1.1       lazyeval_0.2.2        limma_3.46.0
## [31] cli_3.6.1           later_1.3.1           htmltools_0.5.5
## [34] tools_4.0.2         igraph_1.4.2          gtable_0.3.3
## [37] glue_1.6.2          RANN_2.6.1            reshape2_1.4.4
## [40] dplyr_1.1.2         Rcpp_1.0.10           scattermore_1.1
## [43] vctrs_0.6.2         spatstat.explore_3.2-1 nlme_3.1-162
## [46] progressr_0.13.0    lmtest_0.9-40         spatstat.random_3.1-5
## [49] xfun_0.39           stringr_1.5.0         globals_0.16.2
## [52] mime_0.12           miniUI_0.1.1.1        lifecycle_1.0.3
## [55] irlba_2.3.5.1       goftest_1.2-3         future_1.32.0
## [58] MASS_7.3-60         zoo_1.8-12            scales_1.2.1
## [61] promises_1.2.0.1    spatstat.utils_3.0-3  parallel_4.0.2
## [64] RColorBrewer_1.1-3  curl_5.0.0            yaml_2.3.7
## [67] reticulate_1.28     pbapply_1.7-0         gridExtra_2.3
## [70] ggplot2_3.4.2       stringi_1.7.12        rlang_1.1.1
## [73] pkgconfig_2.0.3     matrixStats_0.63.0    evaluate_0.21
## [76] lattice_0.21-8      ROCR_1.0-11           purrr_1.0.1
## [79] tensor_1.5          patchwork_1.1.2       htmlwidgets_1.6.2
## [82] cowplot_1.1.1       tidyselect_1.2.0      parallelly_1.35.0
## [85] RcppAnnoy_0.0.20    plyr_1.8.8            magrittr_2.0.3
## [88] R6_2.5.1            generics_0.1.3        pillar_1.9.0
## [91] fitdistrplus_1.1-11 survival_3.5-5         abind_1.4-5
## [94] sp_1.6-0            tibble_3.2.1          future.apply_1.10.0
## [97] KernSmooth_2.23-21  utf8_1.2.3            spatstat.geom_3.2-1
## [100] plotly_4.10.1       rmarkdown_2.21        grid_4.0.2
## [103] data.table_1.14.8    digest_0.6.31         xtable_1.8-4
## [106] tidyr_1.3.0         httpuv_1.6.11         munsell_0.5.0
## [109] viridisLite_0.4.2

```