Autonomous Discovery for BCR Signaling and Cell Fate Decisions

Bayard Walsh
bkwalsh@uchicago.edu
University of Chicago
Chicago, Illinois, USA

ABSTRACT

Using principles from Autonomous Labs to consider BCR Signaling and Cell Fate Decisions pulling documents from Semantic Scholar literature. Using the data processing pipeline offered in class, Hypotheses, Experimental Plans, and Synthetic Data are generated in order to determine viable future hypotheses within the BCR signaling field. Through using cosine similarity grouping, Tree of Thought Prompting, synthetic data, and correlation metrics, this pipeline can rank the quality of 194 generated hypotheses, with experimental plans, high-level experimental protocols, and synthetic data.

KEYWORDS

B cell receptor, BCR Signaling, Cellular Activation, Imaging Pathway

ACM Reference Format:
Bayard Walsh. 2024. Autonomous Discovery for BCR Signaling and Cell Fate Decisions. In *Proceedings of cmsc35350 (Autonomous Labs).* ACM, New York, NY, USA, 7 pages. https://doi.org/05.2024

1 INTRODUCTION

This paper explores BCR Signaling and Cell Fate Decisions within the Semantic Scholar literature by aggregating papers on significant keywords. Using the data processing pipeline offered through code shared from class, along with additional data processing scripts, Hypotheses, Experimental Plans, and Synthetic Data are generated in order to determine viable future hypotheses within the BCR signaling field. BCR signaling was selected as a topic of choice because the topic was specific yet broad enough for there to be substantial literature on the topic without the queries being too general. Additionally, BCR signaling is a relevant field in vaccine development and medical research.

2 DATA PIPELINE

2.1 Part 1: Literature Retrieval and Processing

To begin the automation pipeline, I used the following query into ChatGPT3 to generate my initial list of keywords on BCR signaling for `eth.txt`:

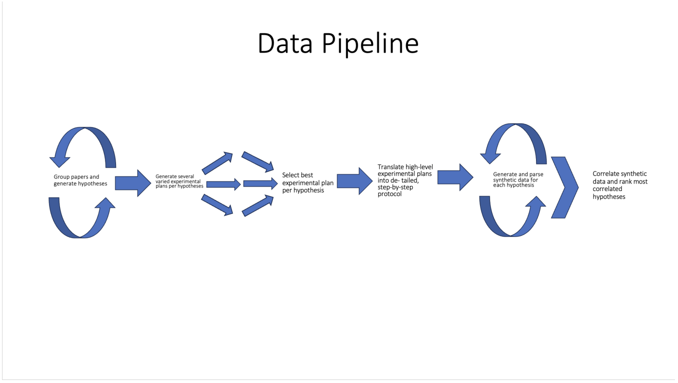

Data Pipeline
Group papers and generate hypotheses
Generate several varied experimental plans per hypotheses
Select best experimental plan per hypothesis
Translate high-level experimental plans into de- tailed, step-by-step protocol
Generate and parse synthetic data for each hypothesis
Correlate synthetic data and rank most correlated hypotheses

Figure 1: High Level View of Data Pipeline

"I want to retrieve documents from Semantic Scholar that relate to the areas of B cell antigen receptor (BCR) signaling and how BCR dependent processes regulate specific cell fate decisions. I am more interested in papers that describe B cell images or patterns in cellular detection. Please suggest to me 100 sets of possible keywords. Output each set of keywords on a separate line, without any numbers."

I followed the scripts outlined in the GitHub repository [1] to pull the papers and generate hypotheses, however, I added a small section of code to `bulk_fetch_viakeyword.py` that capped the amount of papers pulled per query to 40. My mindset is that if a query has more than 40 queries, it is likely too general to my subject to help, however selecting the process of selecting the first 40 arbitrarily could have been improved on, however I was limited by the scope of the project and compute power. Additionally, because `check_relevance.py` had a nontrivial runtime, I wanted to make sure the number of papers I was running my code on was manageable, so reducing the larger queries to small ones heuristically saved runtime on less relevant papers. I used the following prompt below in the script for checking relevance, which was a modified version of the initial prompt but that centered on BCR signaling:

*"Read the following abstract carefully. After reading, answer the following questions to determine if the abstract pertains to research related to the areas of B cell antigen receptor (BCR) signaling and how BCR dependent processes regulate specific cell fate decisions, giving a score of 1 if true and 0 if false, and reporting each in a separate line with the form <HEADING>: <numeric score> <explanation>: **Topic Relevance**: Does the abstract mention 'B cells','activation mechanisms', 'cell fate tracking', or 'pathway visualization'? List any terms used in the abstract that relate to these keywords. **Research Focus**: Is the primary focus of the research on identifying BCR dependent processes that regulate specific cell fate decisions? Specify what the*

*research aims to achieve or discover. **Outcome Mention**: Does the abstract discuss any results or potential outcomes regarding the efficiency, selectivity, or improvement of BCR process identification and cell fate outcomes? Briefly describe these outcomes. **Innovation Highlight**: Does the abstract indicate any novel approaches, techniques, or materials being investigated or used for BCR imaging or identification? Detail any innovative aspects mentioned. **Aggregate Score**: Add the four scores to get an aggregate score of from 0 to 4, and report that score in the form **Aggregate Score**: <score>, along with a sentence of about 50 words explaining your overall score. [BEGIN ABSTRACT " +* chunk *+ " END ABSTRACT]"*

Outside of the mentioned parts all aspects of my data pipeline were identical to the code offered in class.

2.2 Part 2: Information Extraction with LLMs

For this section of the project, I followed the code offered in class exactly. In terms of papers generated, I ran check relevance on 5480 papers and produced 57 papers with a score of 4, 169 with a score of 3, and 253 papers with a score of 2. Looking to take the papers with the most relevant information, I began using these 479 papers (every paper with a score of 2 or greater to check relevancy) for my hypothesis generation.

2.3 Part 3: Hypothesis Generation and Extension

At this point, my approach deviated from the code provided. The first key decision I made was grouping my papers to generate hypotheses on common topics. My rationale for grouping was two-fold; first I wanted to reduce the amount of data instances I was working with, and second I wanted to prevent the model from recreating hypotheses that were hyper-specific to certain papers. While I wasn't working with the same context window that I was using to generate the hypotheses, I was giving the model those hypotheses as inputs, so I believed that synthesizing several of these topics would be a better way to generate more original ideas. Additionally, on some level this is how science is achieved without language models; researchers read technical papers to inform or contribute to that space. I did this by creating groups of a max size of 3 papers; this is because the context window of the LLama model on the network couldn't fit more information than the prompting and the 3 papers into one query, and because of the architecture of the querying system through the network, it wasn't possible to feed more papers into the context window.
In terms of how I formally split papers, I used the Keyword section from the summary section of each paper. From there I clustered papers together with a cosine score of at least .25, and then I split each paper into groups of 3. I wanted a cosine score of at least .25 because querying on individual papers still worked in the system, and having too low of a threshold would reduce the minimum similarity score in groups and make the hypotheses too general. For splitting I picked the groups in a sequential range, with the mindset that all of the papers were similar in the group and could be aggregated together. From this aggregation step, I aggregated the 479 papers into 321 groups.
I used the following prompts before and after the summaries when I queried the LLAMA model on the network:

Read the following summaries of scientific papers carefully. Given these paper summaries, generate a new scientific hypothesis related to the topics mentioned in the paper. This hypothesis should be new (not an exact copy of the shown hypotheses or another paper), scientifically testable, and either provable or disprovable. Please generate A SINGLE question and nothing else, just the hypothesis with the question mark. Here are the summaries to analyze: + insert summaries + *HYPOTHESIS: Implementing a consistent format with a specific tag for hypotheses in responses will improve the efficiency of parsing and processing these hypotheses.*

After initially grouping the papers on Keywords from the summaries, I gave the model the entire summary file contents for the prompt and ran the query above for each group to generate the new hypotheses.
For the work done in this step of the pipeline, refer to `hypothesis_gen.ipynb` and the data file `hypothesis_out.csv`

2.4 Part 4: Experimental Plan Generation

2.4.1 Expanding Data: In this phase of the data pipeline, I focused on developing an experimental plan to test the generated hypotheses, using the network LLM to suggest high-level experimental plans. As this is early on in the pipeline, and many steps in the pipeline involved feeding queries into the network in aggregate and saving the results of those queries, I wanted to experiment with Tree of Thought prompting to see the viability of applying the strategy for future steps in the pipeline. I considered Chain of Thought [2] prompting, however because of the network setup, where we were sending and returning prompts in independent context windows, it was difficult to configure the context window in a way to benefit from the consecutive prompting approach utilized in Chain of Thought prompting. Therefore, I chose an approach similar to Tree of Thought [3], with some modifications specific to this problem. In essence, I wanted to create several potential plans and then choose the best one for each hypothesis. This involved some specific design considerations. First, as the model being used through the network was deterministic, rerunning the same prompts would result in the same output. Therefore, in order to generate better or worse outputs, I would need to vary the prompts. I achieved this by creating several options for prompting the model and then running all permutations through the LLM. The parameters are as follows:

papers_info = "Here are the summaries of the papers this hypothesis was generated from for context:" + add papers

disprove_info = "To check if this hypothesis is true, come up with an experiment designed to DISPROVE THIS hypothesis to verify if it is true or not."

expensive_info = "This experiment has an unlimited budget in terms of time, resources, and financial support."
cheap_info = "This experiment has a limited budget in terms of time, resources, and financial support and should be implementable within a reasonable time frame and for a reasonable price."

Through querying the model with each combination of these supplementary prompts (and queries without them), there were 12 total queries per hypothesis. I decided to offer the context of the papers to see if the papers used to generate the hypothesis would lead to "better" experimental plan generation. To disprove the hypothesis, I wanted to see how asserting the negative would contribute to generating experimental plans. Finally, I wanted to see how adding a high or low perceived budget to the plan generation would impact how the model outputted a plan, as the reminder of a limited budget could theoretically generate plans that are more applicable to the reality of fiscal bounds on laboratory experiments. Those prompts above were added as context to the following prompts:
"Read the following scientific hypothesis carefully: " + hypothesis + varied prompts + "Given this hypothesis, develop an experimental plan to test it using existing scientific methods and equipment. These instructions should outline a high-level experimental approach that could theoretically be carried out in a laboratory."
After running these prompts, I expanded my data from 321 instances to 3852 instances. I found that because of the cost of generating code and constant calls to the network, this query took a significant amount of time (roughly 22 hours), meaning that the upside would have to be considerable to reuse this approach of expansive prompt generation in future steps in the pipeline.
Aside from varying the prompts, another approach I could have taken to expand the data into several experimental plans would be to vary the temperature or `gpt_assistant_prompt` variable (from something other than "You are a super smart AI that knows about science. You follow directions and you are always truthful and concise in your responses.") to see if that created better or worse outcomes, however, I was curious to see the results from changing the text inputs to the model exclusively.

For the work done in this step of the pipeline, refer to `high_level_gen.ipynb` and the data file `high_level_out.csv`

2.4.2 Shrinking Data: Now that the pipeline has generated 3852 instances, I needed a way to shrink the data before the next step in the pipeline. With this I encountered the issue of how to rank the experimental plans in terms of their quality; I wanted to select the best version for each hypothesis before the next step. Additionally, I found that the context window was too small to feed every experimental plan instance into the window in one query, so I couldn't have the model select which of the twelve plans the model liked the most. Therefore I asked the model to rank each plan individually, with the idea of using these numerical rankings to select the best plan in the next step. To do this, I ran every query through the following prompt:

"Read the following scientific experimental plan carefully and rank it from 0-100 based on how well it addresses the hypothesis. The hypothesis will be given after <HYPOTHESIS>." + insert hypothesis + experimental plan + *"Remember, only return a single number; the ranking of the scientific plan. This number should be between 0-100 and should assess the feasibility of the experiment."*

With this query, I generated a numerical score for every plan. Even though this step in the processing loop had the same amount of calls to the network (3852) it was far faster, and took roughly 4 hours to run, which indicates that having less text generated far increased the speed of processing through the network. These scores were initially saved as text in the "score" column and then parsed into numbers (by taking the first instance of a valid number in the string) into the "first_number" column.
However, I quickly ran into a key issue; the scores for the hypotheses were very skewed toward certain points, specifically 85, 80, and 92 had the vast majority of the scores. This can be a result of a couple of aspects; first, if the majority of the plans are very good, it is sensible that the model would score them highly; secondly, it is plausible that the model struggles to differentiate the quality between one score and another. In terms of shrinking the data, I selected the experiment plan with the highest score according to the model, and in the case of ties (which were most instances), I selected the first instance. This approach could have been improved on and certainly favors the attributes of the earliest rows processed in the query, which would be a way to improve this pipeline given more time. While these scores made selecting the "best" experimental plan difficult (as there were many tiebreakers for a given hypothesis, we can use the output to see the attributes of the 74 highest-scoring plans.
Interestingly, these figures suggest that the highest-scoring papers

Value	Count
85	2980
80	794
92	61
90	9
95	4
60	3
82	1

Table 1: Counts of each Score ranked by Language Model

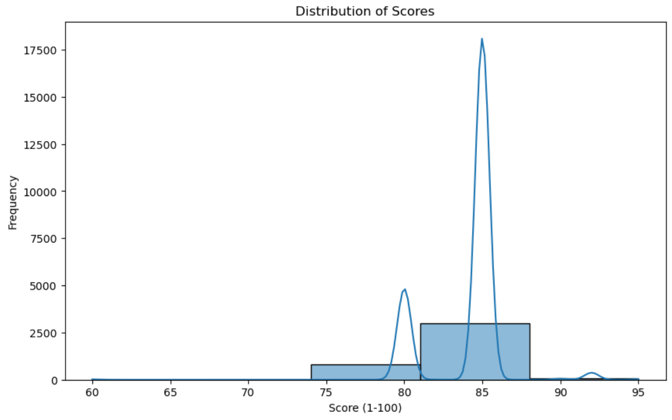


Figure 2: Distribution of each Score ranked by Language Model

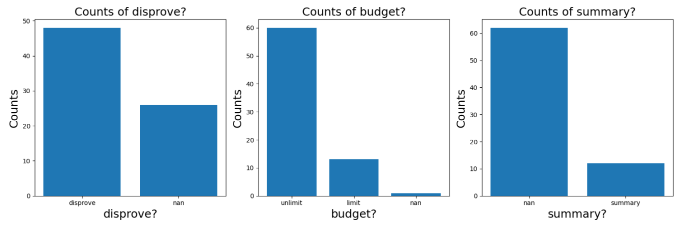


Figure 3: Distribution of Prompt Attributes for Plans with Score >= 90

had less summary and favored an unlimited budget. This would imply that giving more information to the model when assessing the generated hypotheses could cause the model to score lower, meaning that there could be reduced creativity in the language model when given more context. We can also consider the lowest-scoring papers. Because only 3 papers score below 80, we will use 80 and lower as a threshold for "low scoring" papers.

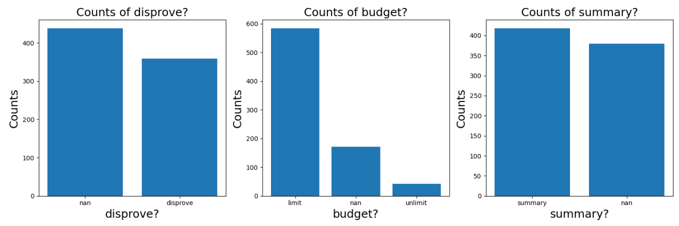


Figure 4: Distribution of Prompt Attributes for Plans with Score <= 80

Here we see that the disprove and summary counts seem to have relatively similar scores, however, it is clear that putting a budget limit on the experimental plan generation far reduces the score from the model; implying that with a limit on the budget, the model will create a plan that performs far worse. This suggests that putting a bound on the potential resources for an experiment reduces the model's creativity.

For the work done in this step of the pipeline, refer to `rank_high_level_gen.ipynb` and the data file `scoring_high_level.csv` and `best_scoring_high_level.csv`. The second CSV takes the top score for each hypothesis and is used in the following step of the pipeline.

2.5 Part 5: Experiment Execution Plan

Here I focused on translating high-level experimental plans into detailed, step-by-step protocols. It's important to note here that many of the experimental plans generated previously already had step-by-step protocols, I believe this in large part because when I was querying the model I used the assistant prompt of "You are a super smart AI that knows about science. You follow directions and you are always truthful and concise in your responses." Also, because of the long run time of the Tree of Thought style approach I applied in the previous step in the pipeline, and because of the similar score given to each experiment, I determined that replicating this process with the formulation of specific plans from the general plans would not be worthwhile. Therefore I simply ran the query on each plan to translate it from a general to a specific plan. I used the following query. As I was only processing 321 high-level plans and generating another 321 high-level plans, this query took roughly 2 hours to run.

Read the following scientific experimental plan carefully, and translate the high-level experimental plans into detailed, step-by-step protocols. Consider the requirements for replicability and specificity in experimental science, and how these steps could be communicated to someone (or something, like a robot) executing the experiment. Here is the plan:+ insert high level plan +*HYPOTHESIS: Remember, return detailed, step-by-step protocols for specific science experiments.*

For the work done in this step of the pipeline, refer to `convert_high_level_to_specific.ipynb` and the data file `specific_protocol.csv`.

2.6 Part 6: Data Analysis and Hypothesis Testing

Here I used a framework of synthetic data as a framework for ingesting experimental data and formally evaluating hypotheses.

2.6.1 Generating Synthetic Data: To generate synthetic data, I once again queried through each row of the model and asked the language model to create a CSV of data generated by varying the inputs of the experiment and simulating it. Of course, this framework is very general and not applicable to every hypothesis, however, given that asking the model directly which experiments are most viable returns a skewed distribution of variables, making it far harder to distinguish the best experiments, I wanted to approach assessing the hypothesis through another form of empirical testing. I was worried that generating the CSVs would take a considerable runtime, however, all these queries took a total of roughly 2 hours to run. I also sought to limit the rows in the CSV to 25 instances, as the number of columns of the generated data varied across hypotheses, and I thought setting this bound for rows would help generate data within the output bounds for each experiment. To generate the data I used the following query:

"Read the following scientific step-by-step protocol carefully and generate synthetic data to simulate the outcomes of the experiment. Use the materials outlined in the protocol to run several tests, varying the amounts of each material, and attempting to replicate the results in a laboratory setting. Provide a dataframe where each row represents a different experimental configuration, with columns indicating the amounts of materials used and the outcome of the experiment. Simulate 25 experiments, resulting in a dataframe with 25 rows and columns corresponding to each material and the experiment outcome. Please return only the dataframe." + insert hypothesis and step-by-step protocol + *"Remember, aim to simulate 25 experiments. There should be a dataframe returned with 25 rows and several columns, and nothing else. Do not return any other words or code, just the dataframe. Do not write 'Here is the simulated dataframe', just return the dataframe."*

For the work done in this step of the pipeline, refer to `create_synthetic_data.ipynb` and the data file `generated_data.csv`.

2.6.2 Analyzing Synthetic Data: To analyze the synthetic data, I parsed the data from the previous step in the pipeline into a CSV and then applied a simple correlation metric across the data frame to get the "correlation score" for a given data frame from all the previous columns correlated to the last column, giving a single value that represents the average correlation between the last column of the DataFrame and all other columns, providing a summary statistic for the relationship between the last column and the rest of the data. This assumes the last column for every data frame contains the outcome of the experiment, which was the approach taken in generating these CSVs initially. Note that some issues arose in the synthetic data: some data instances were strings, some CSVs parsed poorly or had no CSV output at all, and some CSVs were all zeroes. For strings, I set each distinct value in a given column to discrete whole numbers starting from 1, which functioned as a way of converting labels to numbers, though this approach certainly skewed the correlation factor and could have been improved given more time. Any other CSV that had a parsing issue or all zeroes was set to correlation 0. The most important benefit of the correlation is it provides a ranking between hypotheses that is a real number, unlike the previous approach of ranking numbers through the model. Additionally, I also applied the absolute value function to each correlation, as some experiments were designed to disprove a hypothesis (one of the options in the experimental plan generation) and I wanted to score the most likely hypotheses based on their magnitude. This framework allows the user to empirically rank hypotheses and view which few synthetic data instances and which ideas are the most promising, among the 321 instances. From this, we have the following figures:

Correlation	Count
Zero	127
Nonzero	194

Table 2: Counts of each Correlation for Synthetic Data

After parsing and analyzing the data frames, we have over 60% of instances with a nonzero correlation score, and considering that LLMs are not designed to explicitly generate or format synthetic data from English and the experiments were all unique, 60% is acceptable performance. Given more time it would be fruitful to check which CSVs didn't parse correctly, compared to which CSVs had 0 correlation, which would be a way to measure the worst hypotheses.

The graphs shown indicate that several synthetic datasets had a high correlation. Using correlation as a rank, we can formally rank hypotheses with few ties. Next, we will consider which aspects of the prompt design contributed to the highest and lowest scoring correlations. With the process of taking the highest-scoring first experiment, and with so many experiments scoring at 85 or 80, there is a clear skew towards the earlier entries in the permutation, which happened to be the disprove clause.

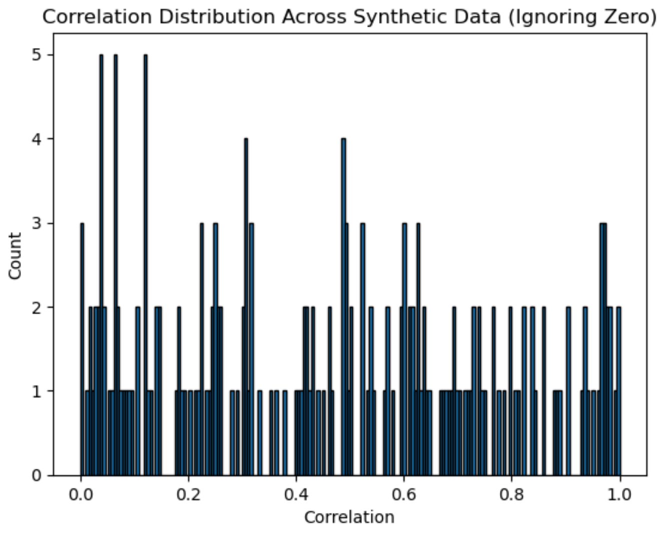


Figure 5: Distribution of Correlation Across Synthetic Data

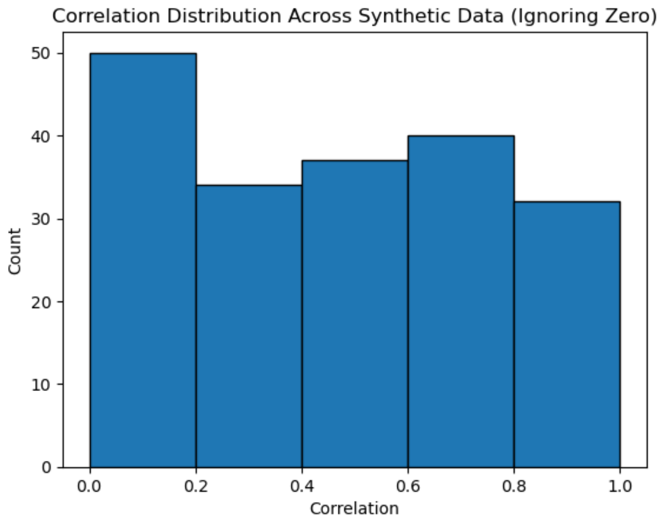


Figure 6: Distribution of Correlation Across Synthetic Data (range bins)

Additionally, when we measure the distribution of prompt features across high and low-scoring correlations we see a similar distribution to the general distribution, suggesting the synthetic data is not very correlated with any specific approach to forming hypotheses taken previously in the pipeline.

Another aspect to analyze the results is the number of papers grouped, compared to the highest-scoring correlations.

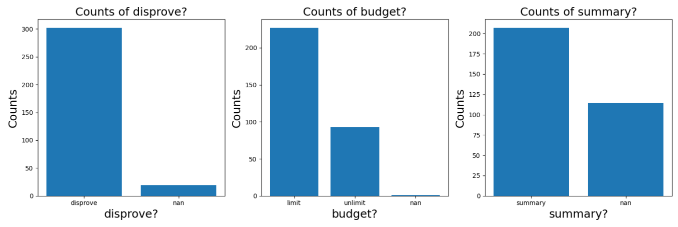


Figure 7: Distribution of Prompt Features Across All Execution Plans

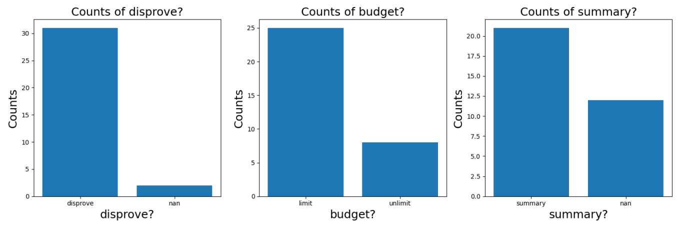


Figure 8: Distribution of Prompt Features Across Execution Plans with Correlation >= .8

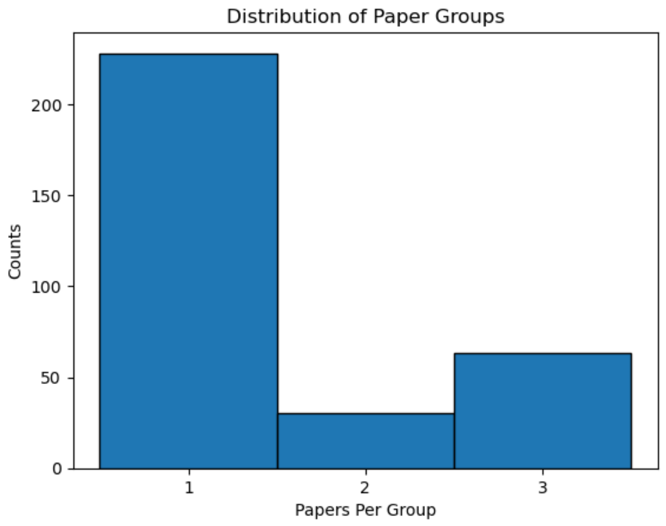
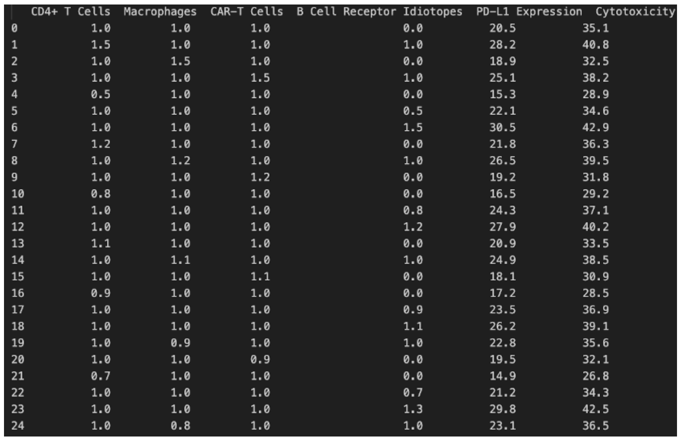


Figure 9: Distribution of Numbers of Papers Per Group

Again, it appears that the distribution of high correlation papers is also not very correlated with the number of grouped papers.

While it is hard to discern specific trends from the generation of hypotheses to the correlation of the synthetic data, we can consider the highest-performing hypotheses themselves:
Highest 5 ranked hypotheses by correlation:
Correlation : 1.0

Does the integration of BAFF, TLR, and RGC-32 signals in atypical B cells promote the production of VH4-34 autoantibodies, contributing to the development of autoimmune responses in SLE patients?



	CD4+ T Cells	Macrophages	CAR−T Cells	B Cell Receptor Idiotopes	PD−L1 Expression	Cytotoxicity
0	1.0	1.0	1.0	0.0	20.5	35.1
1	1.5	1.0	1.0	1.0	28.2	40.8
2	1.0	1.5	1.0	0.0	18.9	32.5
3	1.0	1.0	1.5	1.0	25.1	38.2
4	0.5	1.0	1.0	0.0	15.3	28.9
5	1.0	1.0	1.0	0.5	22.1	34.6
6	1.0	1.0	1.0	1.5	30.5	42.9
7	1.2	1.0	1.0	0.0	21.8	36.3
8	1.0	1.2	1.0	1.0	26.5	39.5
9	1.0	1.0	1.2	0.0	19.2	31.8
10	0.8	1.0	1.0	0.0	16.5	29.2
11	1.0	1.0	1.0	0.8	24.3	37.1
12	1.0	1.0	1.0	1.2	27.9	40.2
13	1.1	1.0	1.0	0.0	20.9	33.5
14	1.0	1.1	1.0	1.0	24.9	38.5
15	1.0	1.0	1.1	0.0	18.1	30.9
16	0.9	1.0	1.0	0.0	17.2	28.5
17	1.0	1.0	1.0	0.9	23.5	36.9
18	1.0	1.0	1.0	1.1	26.2	39.1
19	1.0	0.9	1.0	1.0	22.8	35.6
20	1.0	1.0	0.9	0.0	19.5	32.1
21	0.7	1.0	1.0	0.0	14.9	26.8
22	1.0	1.0	1.0	0.7	21.2	34.3
23	1.0	1.0	1.0	1.3	29.8	42.5
24	1.0	0.8	1.0	1.0	23.1	36.5

Figure 10: Sample Data Frame of Generated Data

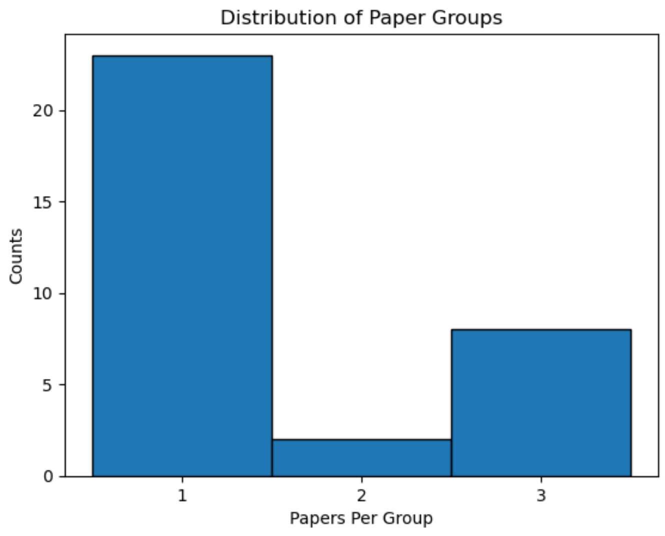


Figure 11: Distribution of Numbers of Papers Per Group with Correlation >= .8

Correlation : 0.997
Can the controlled covalent attachment of a targeting ligand to gold nanoparticles also enhance the delivery of immunotherapeutic agents to cancer cells, leading to increased anti-tumor immune responses?

Correlation : 0.990
Do CXCL13+CD4+ T cells in skin TLSs of pemphigus patients also regulate Treg function through a feedback loop, thereby modulating the chronic blister microenvironment?

Correlation : 0.984

Can the co_expression of tTRII_I7R and TYK2 in CAR_T cells enhance their anti_tumor efficacy and prevent tumor recurrence in B cell lymphoma by converting immunosuppressive TGF_b signaling into immune_activating IL_7 signaling and restoring IFNa_mediated immune surveillance?

Correlation : 0.981
Does the cell-specific expression of transposable elements in immune cells influence the development of autoreactive B cell subsets in systemic lupus erythematosus, leading to the production of pathogenic autoantibodies?

For the work done in this step of the pipeline, refer to `analyze_data.ipynb` and the data file `final_data.csv`.

3 REFLECTION

Regarding the challenges and limitations of using LLMs for automated discovery in scientific research:

During this project, the main limitation I found was working with evaluating the outcomes of generated hypotheses. This is a general challenge in the field of prompt engineering (and broadly scientific discovery), and arguably has no solution, however with an extensive amount of data and with a limited ability to implement projects in practice, I wanted to find a better way of analyzing and ranking hypotheses. I used the approach of ranking hypotheses through the model as well as generating synthetic data and analyzing synthetic datasets as a way to rank hypotheses. The second approach offered a better outcome, however, both had their limitations; namely how can we ensure the quality of this ranking? With synthetic data, where is the data being pulled from; Another table, or just a general pattern decided by the model? However, I think providing a surrogate program to evaluate the model, generate data, or pull similar real data for simulations outside the general LLM would improve the viability of the ranking methodology. After completing this project I feel that LLMs have the potential to create novel ideas, and with this sense, they can contribute to automated discovery. However, beyond creating ideas, the framework of testing through LLM synthesis feels far too broad. Even though the approach I proposed is cheap, easy, and applicable, it lacks the certainty that a formal simulator could offer.

Another complication I found was the way that we were accessing the language model, as the network removed the viability of the context window. In much of the contemporary research on prompt engineering successive prompting and analyzing the previous outcomes within a context window has led to significant results, and allowed us to improve the pipeline, I would look for a way to connect with the language model over the network for a longer period, to write and read successive prompts. This would enable more precise Chain of Thought or Tree of Thought applications and could improve the performance of the model at generating and evaluating hypotheses.

With these limitations and complications specific to the pipeline in mind, there is a clear viability to the idea of scientific aggregation and generation across scientific literature; however, instead of framing LLMs as a closed loop for scientific creation and experimentation, similar to the physical autonomous labs discussed in class, I believe it would be better to consider their usage in suggesting novel ideas to an experienced researcher. With this in mind, the prompt engineering and data wrangling for hundreds of papers feel less practical than fine-tuning a language model on scientific papers directly, and then simply prompting the model for novel hypotheses in direct queries. This approach of fine-tuning would also allow the model to have a large breadth of knowledge across literature and not limit it to summaries within a chat window. Keep in mind that this is an active field of research, for example with the BigBirdPegasus model (large) fine-tuned on scientific papers[4].

4 UTILS

Graphs were created through `graphing_utils.ipynb`. To see the titles of retrieved papers, extracted information, generated hypotheses, experimental plans, and their relevancy break down, look at the `eth` folder and the `eth_scores` folders. `final_data.csv` has the most complete versions of the data used for analysis however the CSVs at the bottom of each step of the pipeline show the updated version of generated data at that point. Not all papers used in relevancy testing are included in the GitHub because of the limit on files in a repository, so only the papers used in the pipeline are included. These are in the `papers_trained_titles.csv`. There were too many summaries to include all the files. See GitHub here (or at https://github.com/bkwalsh/Autonomous-Labs-Final-)

5 ACKNOWLEDGMENTS

Thank you to Professors Ian Foster and Rick Stevens as well as TA Marcus Schwarting for all the support and guidance throughout the course!

REFERENCES

[1] Ian Foster. 2024. Cmsc35350 repository. Accessed: 2024-05-24. (2024). https://github.com/uchicago-cs/cmsc35350/tree/master?tab=readme-ov-file.
[2] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903. https://arxiv.org/abs/2201.11903 arXiv: 2201.11903.
[3] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: deliberate problem solving with large language models. (2023). arXiv: 2305.10601 [cs.CL].
[4] Manzil Zaheer et al. 2021. Big bird: transformers for longer sequences. (2021). arXiv: 2007.14062 [cs.LG].

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009