Enhancing Author-Paper Assignment Accuracy through Fine-Tuning Lightweight Transformer Models

Anvesha Dutta
B.S. Data Theory
dutta.anvesha06@gmail.com

Sam Hopkins
B.S. Computer Science
samthehopkin@gmail.com

Kehan Li
B.S. Financial Actuarial Mathematics
kehan1230@gmail.com

Tsunming Liu
B.S. Mathematics of Computation
tsunmingliu2024@gmail.com

Adithi Ramesh
B.S. Computer Science
adithi.ramesh02@gmail.com

Andy Wang
B.S. Data Theory
andywang0321@gmail.com

Abstract

Inaccurate attribution of papers to authors remains a significant challenge in the academic community, complicating the management of scholarly resources. This study explores the efficacy of lightweight transformer models in detecting incorrect paper assignments. We initially conducted exploratory data analysis on an author-paper dataset, identifying potential correlations between paper attributes and erroneous assignments. Due to computational constraints, we fine-tuned Microsoft's Phi-1.5 model instead of more resource-intensive alternatives. Utilizing a weighted Area Under the ROC Curve (wAUC) for evaluation, our model demonstrated superior performance compared to existing baselines, achieving a wAUC of 0.80181. Despite limitations in computational resources and competition timing, our findings underscore the potential of lightweight models in improving academic database management and resource allocation.

1 Problem statement

In the academic community, accurate attribution of papers to authors is crucial for maintaining the integrity of scholarly records and ensuring proper credit for research contributions. However, the process is often fraught with errors due to name duplication, spelling variations, and other ambiguities. These inaccuracies can lead to misattributions that impact author reputation, research visibility, and resource allocation.

The task at hand is to develop robust models that can detect and correct incorrect paper assignments in a dataset comprising author names and their associated papers. The provided dataset includes detailed information such as titles, abstracts, author lists, keywords, venues, and publication years for each paper. The goal is to leverage this data to train a machine learning model capable of accurately identifying and flagging incorrect assignments, thereby improving the reliability of academic databases and publication records.

In this context, the study aims to: 1. Perform exploratory data analysis to uncover patterns and correlations that may indicate erroneous assignments. 2. Fine-tune a suitable large language model to effectively discern correct from incorrect paper assignments based on the given data attributes. 3. Evaluate the model's performance using appropriate metrics, specifically focusing on the weighted

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Area Under the Receiver Operating Characteristic Curve (wAUC) to ensure accuracy across various author profiles.

By achieving these objectives, the research seeks to enhance the efficiency and accuracy of author-paper attribution, providing a valuable tool for academic search engines, journals, publishers, research institutions, and libraries.

2 Literature review

In the ever-expanding academic landscape, accurate attribution of papers to authors is paramount, yet challenges persist due to errors in assignment. This review surveys the literature on detecting and addressing such errors.

Early efforts relied on matching algorithms based on author names, keywords, and content, but faced limitations like name duplication and spelling disparities. Recent advancements in machine learning and natural language processing have led to more effective methods rooted in deep learning and text mining.

Studies have explored techniques to enhance allocation systems and academic databases, introducing innovations such as author matching algorithms based on entity recognition and automatic paper assignment systems.

The WhoIsWho project has notably introduced a large-scale academic name disambiguation benchmark and toolkit to address the challenge of ambiguous author names. They offer a comprehensive set of tasks and competitions to spur method development.

The complexity of name disambiguation is attributed to non-uniform task designs and errors in noisy data. WhoIsWho's benchmarking process and toolkit aim to address these challenges, emphasizing a multimodal approach integrating semantic and relational features.

Community-driven and open-source, WhoIsWho welcomes contributions to advance name disambiguation methods. Future research should focus on more accurate detection methods, analysis of error mechanisms, and optimization of academic databases and allocation systems.

A proposed machine learning model aims to automate paper attribution by discerning relevant features from limited information such as abstracts, titles, publication dates, and journal names. This model holds potential for academic search engines, journals, publishers, research institutions, and libraries to enhance resource management and utilization.

In summary, advancements in paper assignment error detection offer promising avenues for enhancing the efficiency and accuracy of academic research attribution and resource management.

3 Methodology

3.1 Exploratory Data Analysis

We were given the author dataset with authors as the key and the papers as the values, a validation set, and a paper set that had papers as the key and information about the paper. The paper attributes provided in the dataset are Title, Abstract, Authors, Keywords, Venue, and Publication year. Before finetuning any model, we performed exploratory data analysis on the dataset. We converted the JSON files into pandas data frames using Python and merged the author and paper datasets. We found out the summary statistics and plotted some visualizations to see some attribute distributions. We especially focused on attempting to find any correlation between the attributes and outlier assignments or incorrectly assigned papers to authors in the author dataset. We also performed some statistical tests like the Chi-Squared test and the T-Test but none gave any significant results. We moved on to process much more complex models to understand the relationships within the dataset.

3.2 Train Test Split

Since we did not have access to labels for the evaluation dataset, we decided to split our dataset so that 10% of it is held aside as a test dataset. You can find our hold-out test set by the names

train_author_inference.json and **train_author_inference_labels.json** for the features and labels respectively in our code repository.

3.3 Model Fine-tune

We originally wanted to fine-tune the state-of-the-art open-source LLM **LLama3** by Meta, but we quickly learned that we have nowhere near the amount of compute power necessary. So after some further research into popular open-source LLMs, Microsoft's **Phi-1.5** caught our eye. The authors of **Phi-1.5** claims that it is able to perform on-par with many popular LLMs with more than 10 billion parameters using only 1.3 billion parameters, thus enabling a much more lightweight and efficient fine-tuning and inference setup.

To fine-tune the Phi-1.5, we took great inspiration from the ChatGLM code repository for IND and mostly reused its boilerplate code, which was very appropriately written to transform the given author/paper datasets into a format appropriate for a generative large language model. Wherever necessary, we made changes to the configuration files so that it works with the differences in model architecture between the two models.

To actually perform the model fine-tuning, we had to purchase a Google Colab Pro+ membership in order to gain access to more compute units, longer runtimes, more VRAM, and background execution. Since Google Colab is essentially a python jupyter notebook environment, we had to use magic commends (the ones denoted by the **!**) in order to execute shell commands to run our training script.

3.4 Inference on Test Split

To abide by the rules of the IND competition, we will be evaluating the performance of our model using the Weighted Area Under the Receiver Operating Characteristic Curve (wAUC), defined as follows:

$$wAUC = \sum_{i=1}^{M} AUC_i \times w_i, \tag{1}$$

where AUC_i denotes the AUC score of the i-th author for all M authors, and w_i is the weight of author i defined as

$$w_i = \frac{\textit{\# of errors of author i}}{\textit{\# of total errors}} \tag{2}$$

Since we do not have access to target labels for the validation dataset, we had to evaluate our model on the hold-out portion of the training set. To do so, we made use of the **inference.py** file from the ChatGLM paper, where instead of using the evaluation set as the argument, we used our own test split.

Additionally, since the provided ChatGLM model checkpoint came from training on the entire training dataset, we believe that using our hold-out test set for baseline evaluation is an unfair comparison. Additionally, in the documentation of the ChatGLM model, evaluation requires eight A100 GPUs to complete, which is far outside our budget. So as a baseline, we will use the Area under the ROC curve reported for the evaluation set in the ChatGLM documentation. Note that this assumes that the evaluation dataset is similar enough to our hold-out test dataset from the training set.

4 Results and Discussion

4.1 Exploratory Data Analysis

Through our exploratory data analysis, we created many interesting visualizations which helped us better understand the data we are working with and the task at hand. Not all of them are super relevant to the goal of this paper, though, so we have placed these visualizations in the appendix at the end. We recommend taking a look at them. If these data insights are interesting, more can be found in our python notebook titled **EDA.ipynb**.

We had carried out exploratory data analysis with looking for feature correlation in mind, so we carried out some statistical tests. Here are the results:

Statistical Test Results			
Feature	Method	Test Statistic	P-Value
Venues	Chi-Squared	$\chi^2 = 50050.49$	≈ 0
Number of AuthorsT-test	T-TestNumber of Authors	$\hat{t} = -8.46$	≈ 0
Year of PublicationT-test	T-TestYear of Publication	$\hat{t} = 13.85$	≈ 0

Figure 1: statistical tests on extracted features

Despite the p-values in Table 1 seemingly suggesting that the extracted features are really significant predictors, they should not be taken seriously. This is because the counts of normal papers for each author significantly out-number the counts for incorrectly-assigned papers, thereby violating the underlying assumptions of these tests. See the appendix for visualizations of the distributions of these features grouped by class.

4.2 Fine-tune Findings

Due to the previous challenge of insufficient compute resources to fine-tune LLaMa3, we began to question whether we really need 6 billion parameters to perform NLP tasks. This led us to explore Phi1.5, a transformer model with 24 layers, 32 heads (each with a dimension of 64), and rotary embeddings with a rotary dimension of 32. Phi1.5 has a context length of 2048, employs flash-attention for faster training, and uses byte-pair encoding for tokenization. With a total of 1.3 billion parameters, it is much lighter than the baseline ChatGLM, which has 6 billion parameters and a similar transformer-based architecture.

We are particularly interested in Phi1.5 due to its significant performance in multi-step reasoning, as highlighted in its paper. Phi1.5 outperforms LLaMa-2 7B in multi-step reasoning, suggesting it is capable of handling complex NLP tasks—precisely the quality we require. In contrast, ChatGLM is optimized for conversational purposes.

Given the similar architectural designs of both models and the availability of the necessary API from Hugging Face, we decided to fine-tune Phi1.5 using the provided code for the ChatGLM solution. Our goal is to determine if Phi1.5 can outperform our baseline model.

In the end, we ran the fine-tune script for Phi-1.5 for 2.4 epochs through our training split before we decided that the loss has converged. See the training loss plot over all epochs below:

As observed, the loss remained stagnant at around 0.03, leading us to conclude that the model has converged. Here is the same loss curve for the first 200 iterations of training:

To ensure that we have indeed reached an optimum in the parameter space, we also used a learning rate scheduler. Here is the learning rate used throughout the entire training process:

When viewed in conjunction with Figure 2, we notice that initially, the learning rate was way too low and so the loss remained high. But when the scheduler increased the learning rate, the loss began to decrease rapidly. When the loss became stagnant, the scheduler continuously decreased the learning rate in order to find the most optimal set of parameters in the local region in the parameter space. This shows that the model has indeed converged to an optimum.

Lastly, observe the norm of the gradient vectors for each gradient descent step of the training process in Figure 5

Here, we see that as training goes on, the gradient norms becomes smaller and smaller, suggesting that little changes could be made to further improve the performance of the model.

4.3 Test Split Performance

Now let us take a look at the performance of the model on our hold-out test split of the training set in Figure 6.

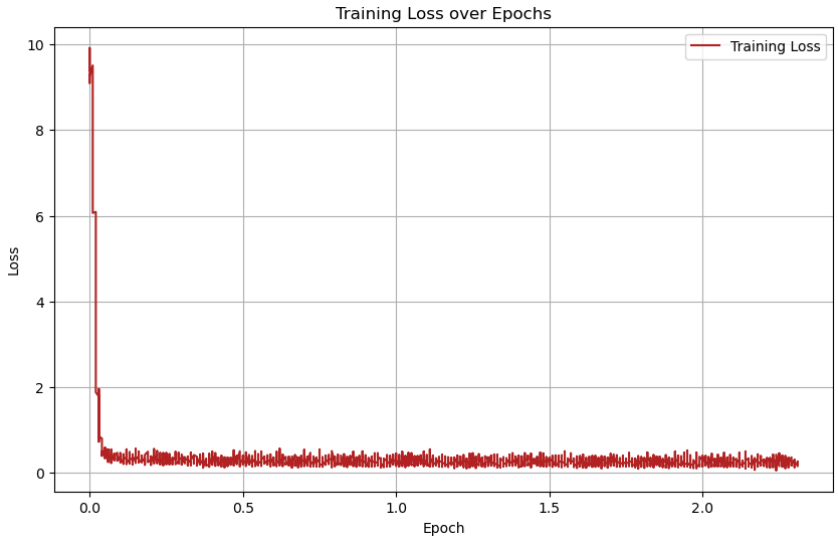

Figure 2: training loss of Phi-1.5 LLM on training split over 2.4 epochs

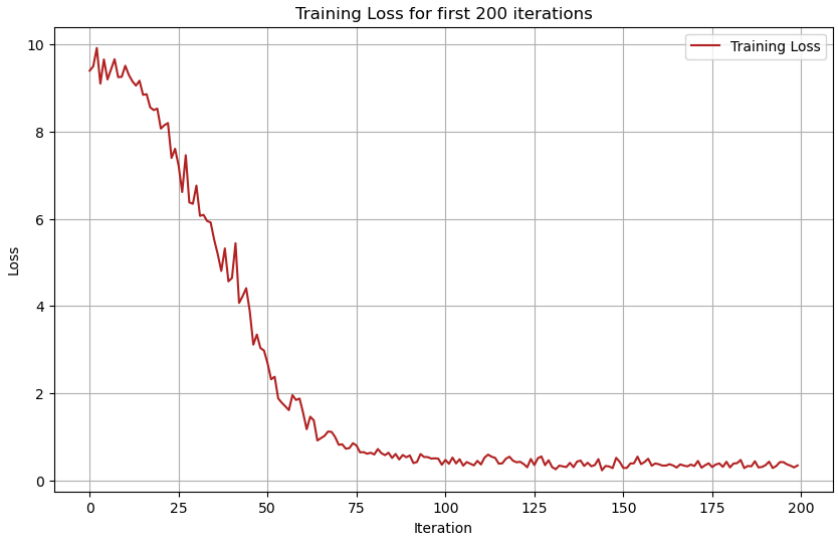

Figure 3: training loss of Phi-1.5 LLM on training split over 200 iterations

As shown, even with far fewer parameters than the best performing baseline model ChatGLM, Phi-1.5 achieved a higher weighted AUC score. Note that all baseline performance metrics were obtained from their respective documentations, which was obtained through inference on the evaluation dataset, whereas in our case, we obtained our weighted AUC score from inference on our hold-out test split of the training dataset. There are several reasons why we did not choose to run inference on the evaluation dataset:

1. The evaluation dataset was too large, we did not have enough time and compute to repeatedly run inference on it in order to improve our model.
2. Model fine-tuning took so long that we did not manage to finish training before the end of the competition, which means that even if we ran inference on the evaluation dataset, we would have had no way of evaluating the performance of our model.

On the other hand, we could have re-trained the baseline models on our training split and evaluated them on our test split. We did not end up doing this for several reasons:

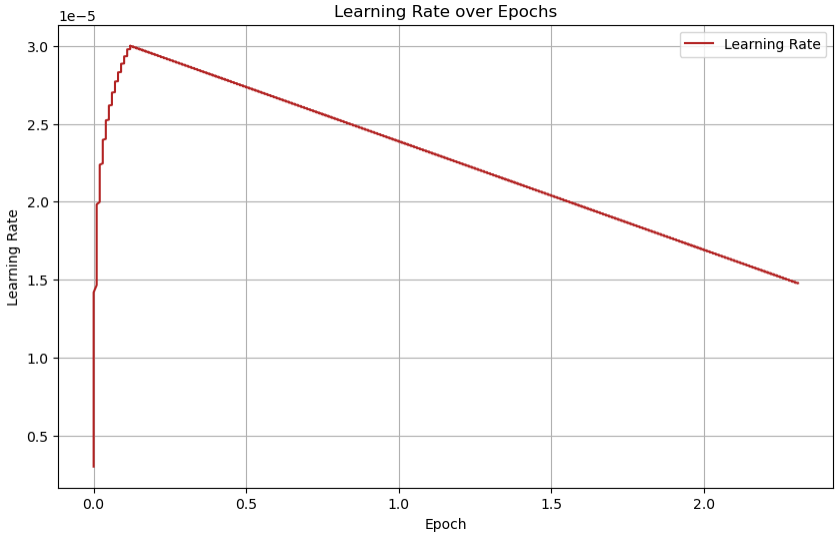

Figure 4: learning rate for fine-tuning Phi-1.5 LLM on training split over 2.4 epochs

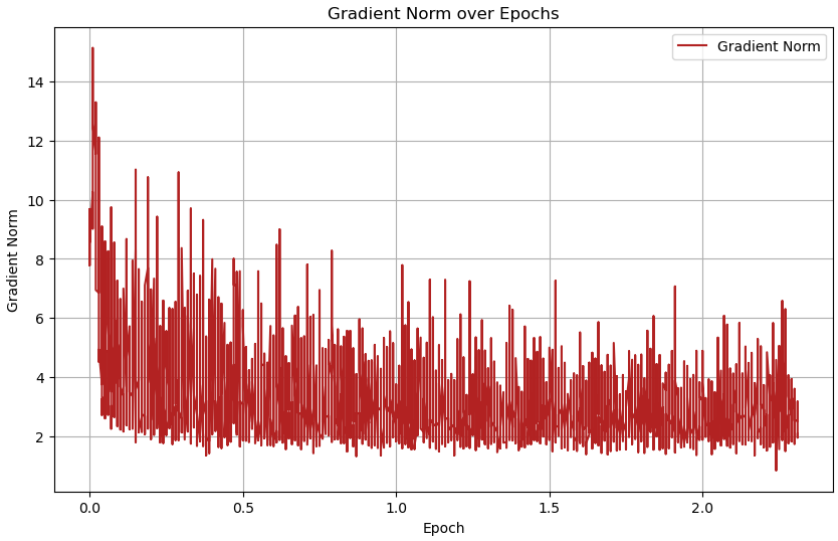

Figure 5: norm of gradients for fine-tuning Phi-1.5 LLM on training split over 2.4 epochs

1. The graph-based models (GCN and GCCAD) require compute-intensive data preprocessing in order to transform the dataset into a graph-based structure, which we felt we did not have the expertise and compute to do.

2. The ChatGLM model was simply too large (the documentation recommends using 8 A100 GPUs to simply do inference, one can only imagine how much time and compute is needed for fine-tuning) for us to try fine-tuning it. Not only did we not have enough computational power and time, we did not even have enough VRAM to fit the model in its entirety in our GPU.

But why didn't we just use the provided model checkpoint for ChatGLM to perform inference on our test split? This is because our test split is a subset of the full training dataset, which was used to train the provided checkpoint of ChatGLM in the first place. This would definitely make for an unfair comparison because we are essentially evaluating ChatGLM on the data is was trained on.

Weighted AUC Scores			
Method	Parameters	Evaluation Dataset	Weighted AUC
GCN	-	Evaluation Set	0.58625
GCCAD	-	Evaluation Set	0.63451
ChatGLM (best baseline)	6 Billion	Evaluation Set	0.71385
Phi-1.5 (this paper)	1.3 Billion	Test Split	**0.80181**

Figure 6: comparison of performance between Phi-1.5 and baselines

With all these considerations in mind, we decided that so long as the training dataset is similar enough to the evaluation dataset (which we believe to be a valid assumption), our test split from the training set must be an accurate reflection of the evaluation set and thus the weighted AUC scores obtained on the test split is reflective of the model's performance on the evaluation dataset. In the ideal case, our group would have finished fine-tuning our models much before the competition deadline, but we were faced with many challenges and difficulties which set back our progress, which we will go into depth in the next section.

In the end, we were able to show that the much more light-weight Phi-1.5 model is able to significantly improve the performance on the IND task as compared to the baseline models. This makes sense since Phi-1.5 was designed specifically with logical reasoning in mind, whereas ChatGLM was originally designed to perform well at language generation. We believe that language generation is admittedly a much more complex task than simple logical reasoning, thus requiring many more parameters, but in the context of incorrect name detection, simple logical reasoning is the attribute we are looking for.

5 Challenges and Difficulties

In the progress of our research, we encountered several technical challenges and adopted a series of solutions to ensure the smooth progress of the project.

5.1 Compute Limitations

Even with Google Cloud credits available, we were unable to get quota requests approved for access to GPUs more powerful than the Nvidia L4. This meant that we were unable to get enough VRAM to run our model on a Google Cloud virtual machine, which was our original plan. We decided to try using Google Colab instead. Initially, we connected Colab to the virtual machine instance and configured the Settings to 60GB of memory and 200GB of storage. Our GPU power was still insufficient, so we decided to buy Colab Pro+ (the $50 per month plan that offers 400 compute units) to get more computing resources. This still did not provide enough compute to complete the entire training. However, we noticed that the loss value remained stagnant for the last thousand iterations, so we decided to end the training after 2.4 epochs because we had reached the convergence state.

After we finished training, we had about 80 compute units left. With limited resources, we believe these computing units are more effective for reasoning and evaluation than continuing training. Therefore, we prioritized the allocation of compute units to inference and evaluation tasks to ensure that we could obtain reliable evaluation results of model performance. We had to be very conservative with our limited computational resources and optimized as much as we could, but we were likely unable to obtain the optimal results.

5.2 Model Limitations

The LLM we originally planned to use was Meta Llama3, which is powerful and state-of-the-art. Llama3 ended up being such a heavyweight model that we would be unable to finish running it in time with our limited compute power. We then switched to the lighter weight Microsoft phi-1.5 model. Phi-1.5 has 1.3 billion parameters, significantly less than Llama3's 8 billion. This means that phi-1.5 is unable to learn as well as Llama3, reducing the quality of our results.

5.3 Programming Challenges

While solving the computing resource problem, we also ran into code compatibility issues. During last Friday's attempt, we needed to clone the codebase and encountered many other issues that required code changes to ensure compatibility. Resolving these issues ensures that our code runs smoothly in different environments and is consistent with various dependencies.

In order to deal with the problem of insufficient computing resources, we designed a checkpoint preservation strategy. A checkpoint is saved every 250 iterations, and if computing resources run low, we can load the nearest checkpoint into the virtual machine to continue training. This approach ensures that we are able to interrupt and resume the training process even with limited resources.

5.4 Competition Timing

The hardware and technical challenges we experienced made it so that we were unable to submit our results to the OAG-Challenge in time. This meant that we had to rely solely on the results obtained on our own dataset. This limited our ability to compare and validate our results against external benchmarks or peer submissions, potentially affecting the robustness of our findings.

5.5 Areas for Improvement

To improve performance and the quality of results in this challenge, several difficulties must be addressed. The key way in which this project could be improved is with access to more computing resources. Improved optimization techniques could also be used to run powerful models using less compute power. Both of these improvements would allow a more powerful model to be used. Modern models that we expect to provide better results are Llama3, Gemini 1.5, and GPT 4. These LLMs all use far more parameters than phi-1.5, allowing them to learn from the data and detect paper assignment errors more effectively. Additionally, submitting the results in time for the competition would allow for more accurate validation of our methods.

6 Conclusion

In this study, we addressed the problem of incorrect paper assignments to authors by leveraging lightweight transformer models, specifically Microsoft's Phi-1.5. Our comprehensive exploratory data analysis laid the groundwork for understanding the dataset, although traditional statistical tests did not yield significant predictors due to class imbalances. By fine-tuning Phi-1.5, we achieved a notable improvement in weighted AUC scores compared to existing baselines, highlighting the model's capability in logical reasoning tasks pertinent to name disambiguation.

Our approach demonstrated that even with limited computational resources, significant advancements in detecting erroneous paper assignments are achievable. The challenges faced, particularly in compute limitations and competition timing, highlight the necessity for better resource optimization and access to more powerful computational environments. Future work should focus on employing more advanced models such as LLaMa3, Gemini 1.5, and GPT-4, alongside continued optimization of lightweight models for enhanced accuracy in academic attribution tasks. This research paves the way for more effective and efficient management of academic resources, contributing to the broader goal of improving the integrity and reliability of scholarly databases.

References

[1] Chen, B., Zhang, J., Zhang, F., Han, T., Cheng, Y., Li, X., … & Tang, J. (2023, August). Web-scale academic name disambiguation: the WhoIsWho benchmark, leaderboard, and toolkit. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 3817-3828).

[2] Li, Y., Sébastian, B., Eldan, R., Del Giorno, A., Gunasekar, S., Tat Lee, Y. (2023, September). Textbooks Are All You Need II: phi-1.5 technical report. arXiv preprint arXiv:2309.05463

[3] Zhang, F., Shi, S., Zhu, Y., Chen, B., Cen, Y., Yu, J., … & Tang, J. (2024). OAG-Bench: A Human-Curated Benchmark for Academic Graph Mining. arXiv preprint arXiv:2402.15810.

[4] Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In A. Gangemi, R. Navigli, M. E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, & M. Alam (Eds.), Semantic Web (ESWC 2018) (Lecture Notes in Computer Science, Vol. 10843, pp. 593-607). Springer. https://doi.org/10.1007/978-3-319-93417-4_38

[5] Xu, Y., Liu, X., Liu, X., Hou, Z., Li, Y., Zhang, X., Wang, Z., Zeng, A., Du, Z., Zhao, W., Tang, J., & Dong, Y. (2024). ChatGLM-Math: Improving math problem-solving in large language models with a self-critique pipeline. arXiv preprint arXiv:2404.02893. https://doi.org/10.48550/arXiv.2404.02893

Appendix

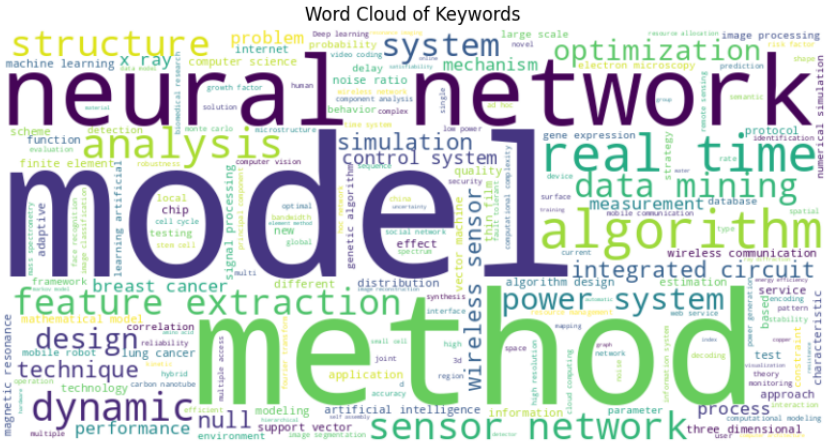

Figure 7: wordcloud visualizing the most common keywords among all papers

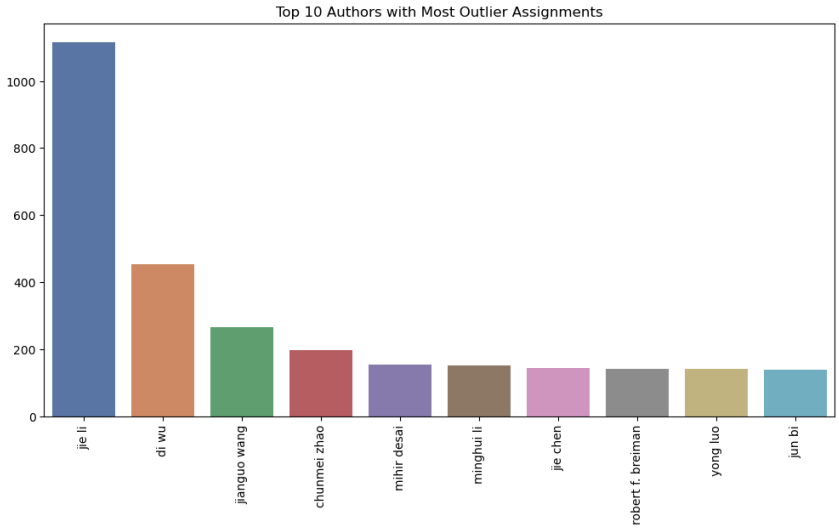

Figure 8: top 10 authors with most outlier assignments

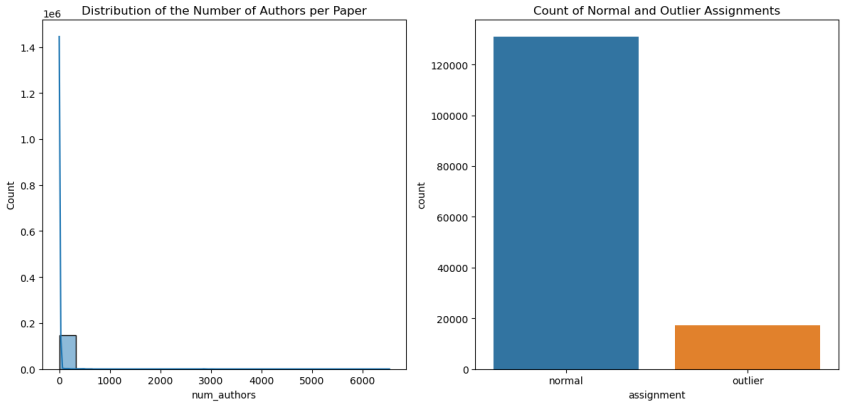

Figure 9: distribution of number of authors per paper (left), counts of data classes (right)

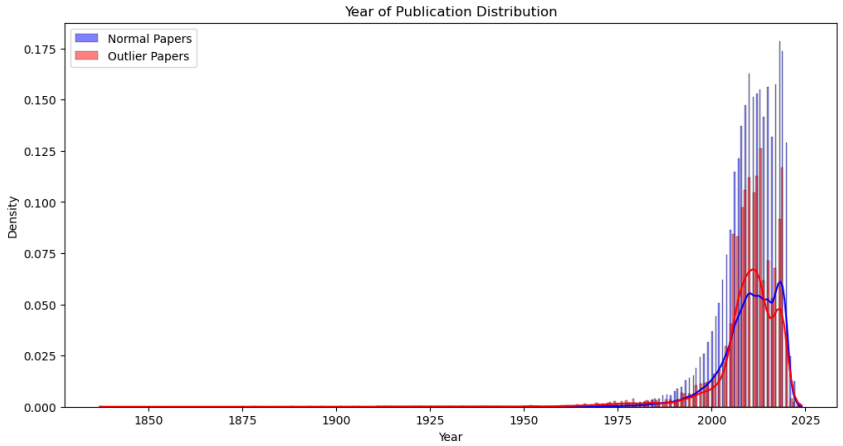

Figure 10: distribution of puplication year, grouped by data class

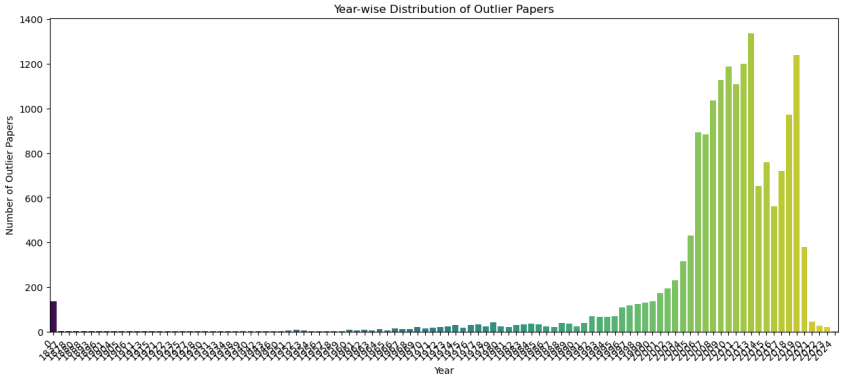

Figure 11: distribution of publication year of outlier papers

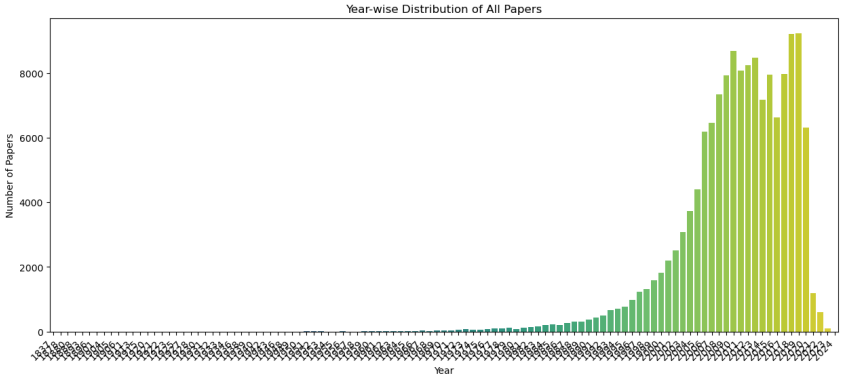

Figure 12: distribution of publication year for all papers

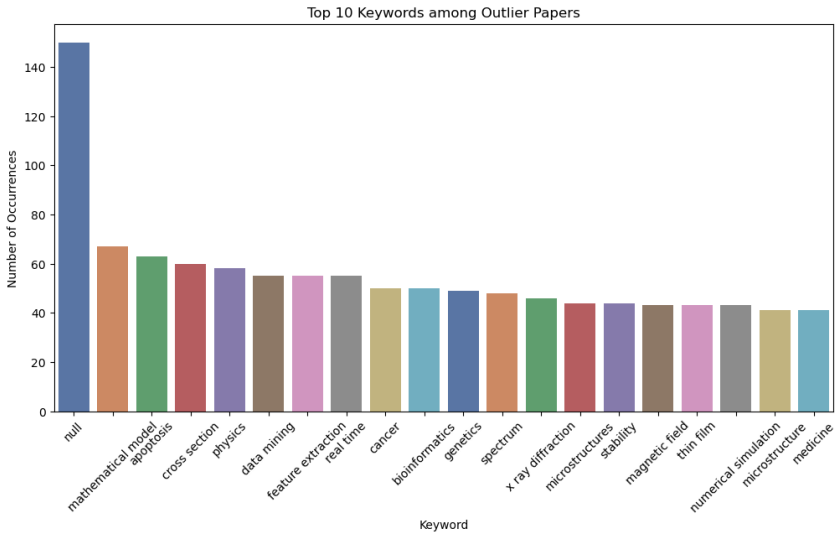

Figure 13: top 10 keywords among outlier papers

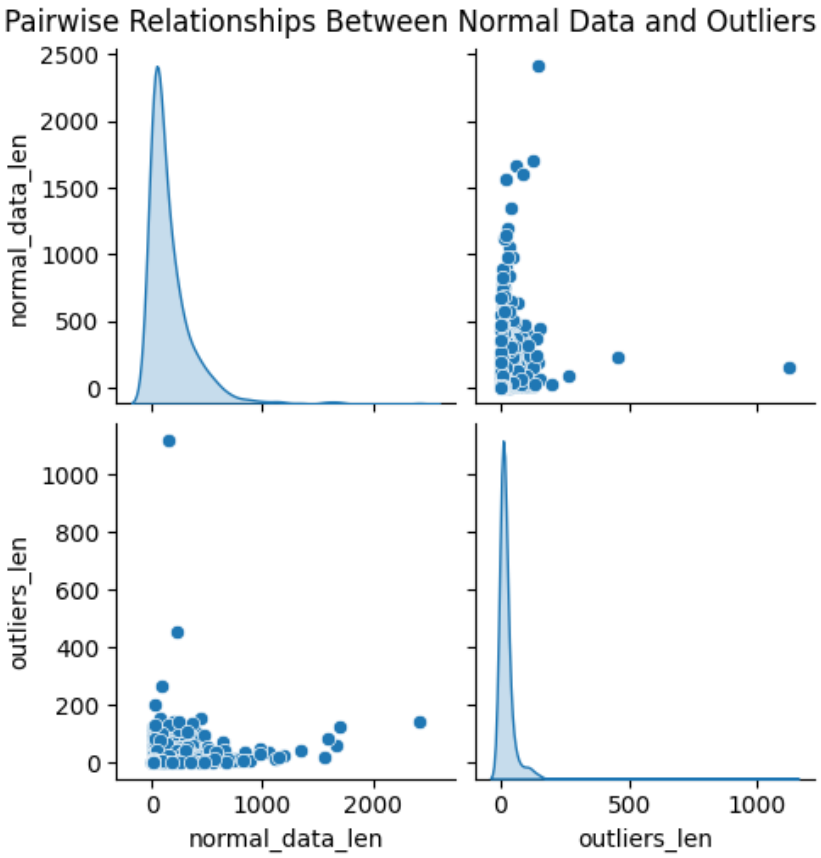

Figure 14: comparison between counts of normal and outlier papers

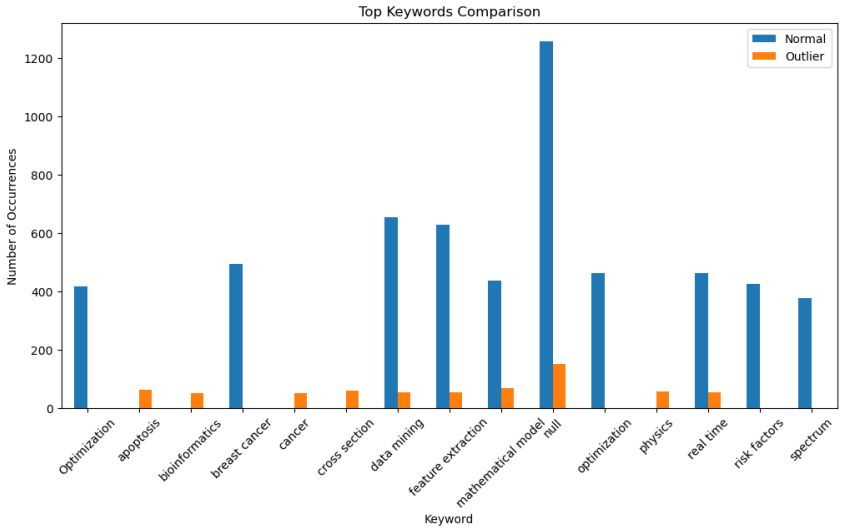

Figure 15: top keywords, grouped by data class

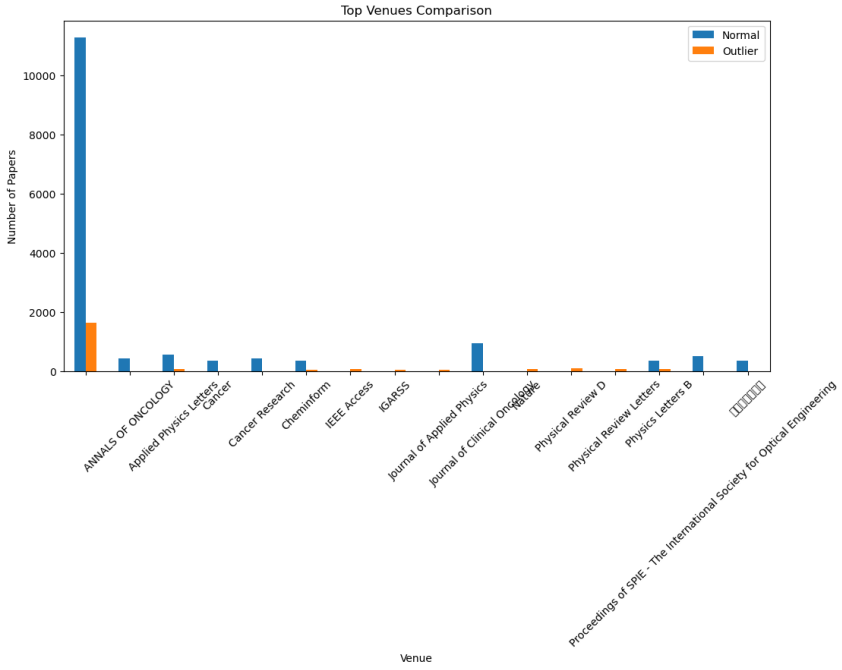

Figure 16: number of papers from top venues, grouped by data class

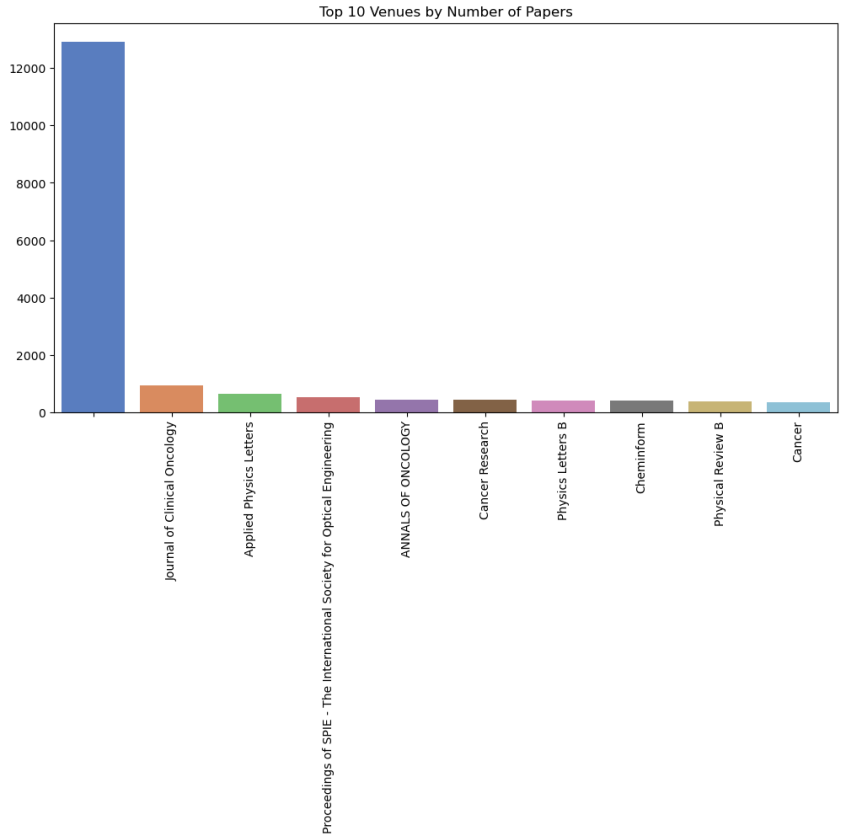

Figure 17: top 10 venues by number of papers