From Section Type Classification⟶ to⟶Fine-Grained Literature Review

A Computational Workflow for Analyzing Scientific Literature

Helene L. Bygnes
Severin Gartland
Lars Campsteijn-Høiby
Lisa Sørdal

While text embeddings provide powerful tools for analyzing scientific literature, treating articles as single data points obscures the distinct functions of their sections. This report presents a complete workflow for performing a more fine-grained, section-level analysis on a corpus of 1222 articles from the *PRPER* journal.

This work first develops and compares multiple methods for **classifying article sections by type**. We show that while an API-based Large Language Model achieves the highest accuracy (∼90%), a novel, computationally efficient projection-based classifier also performs strongly (∼80%), surpassing a fine-tuned SciBERT model. Building on these classifications, we then introduce an iterative LLM-based workflow to **automatically discover and track fine-grained trends in theory and methodology** over two decades, enabling a new form of automated literature review.

INTED – UiO 18. august 2025

Contents

1 Introduction

The ever-expanding volume of scientific literature presents a significant challenge for researchers seeking to understand the trajectory of their field. While traditional review articles provide invaluable summaries, computational methods using text embeddings have emerged as a powerful tool for conducting large-scale, data-driven analyses of entire research corpora. This approach has been successfully used to map broad thematic trends by representing each scientific article as a single vector in a high-dimensional "meaning space" (Odden et al., 2024).

However, analyzing articles as monolithic units has a key limitation: it conflates the distinct functions of their constituent parts. An article is not a single block of text, but a structured argument composed of an *Introduction*, *Methods*, *Results*, and *Discussion*. Reducing an entire article to a single data point risks obscuring these distinct components, losing valuable information about a field's theoretical foundations or the evolution of its research methods. A section-level analysis, therefore, promises a more nuanced and powerful understanding of how a discipline develops.

This report details a project that develops and evaluates a complete workflow for such a section-level analysis, using a corpus of 1222 articles from the *Physical Review Physics Education Research* (PRPER) journal. Our primary contributions are threefold. First, we develop and compare a suite of models for classifying article sections by their section type. We demonstrate that while a Large Language Model (LLM) using an external API achieves the highest accuracy (∼90%), our novel, projection-based MLP classifier provides a highly competitive and computationally efficient local alternative (∼80%), significantly outperforming a standard fine-tuned SciBERT model (∼77%). Second, using these classifications, we introduce an iterative LLM-based method to automatically discover and track the prevalence of specific theoretical frameworks and research methods. Finally, we developed a use of projection-based visualizations as a tool for exploring the semantic space of academic theories.

The report is structured as follows. We begin by preparing the data and performing an initial exploration of the section embedding space. We then present our detailed comparative analysis of the different section-type classifiers. The report then demonstrates the utility of these classifications in two applications: the fine-grained trend analysis using an iterative LLM and the exploratory analysis using vector projections, before concluding with the implications of this work.

2 Background

2.1 Previous work

This project is inspired by the ongoing work of Helene Lane, where she employs a centroids based natural language processing (NLP) method developed by Odden et al. (2024). Centroids for text summarization were first introduced by (Radev et al., 2004) and consist of averaging the positions of a set of samples in the embeddings space. These averages – the centroids – should then represent some semantic meaning shared by its constitutive samples.

Lane has embedded a dataset of 1222 whole articles from PRPER. From these, centroids were computed based on handpicked sets representing common topics in PRPER. The topic categories are: "Mechanics", "Electricity and Magnetism", "Sound and Waves", "Relativity", "Thermal Physics", "Optics", "Fluid Dynamics", "Quantum Physics", "Astrophysics", "Identity", "Lab" and "Attitudes". By calculating the distance from each of the articles to these centroids, she can visualize the distribution of topics within physics education in an embedding (or "meaning") space. She intends to use the distribution to investigate the evolution of topics in PRPER over time. In other words, like (Odden et al., 2024), she employs text embeddings and centroids to conduct a qualitative analysis.

2.2 Project Goals and Structure

A limitation of Lane's work is its reliance on *article-level embeddings*, which reduces each complex document to a single point in the embedding space. Since scientific articles have a clear sectioned structure (e.g., theory, methods, results), a more fine-grained, *section-level analysis* could reveal new features.

To address this, our project comprises four primary components that build upon one another:

- **Data Preparation:** To process the raw XML data by splitting each of the 1222 articles into its constituent sections and converting the text of each section into a high-dimensional vector representation (embedding).
- **Exploratory Analysis:** To investigate the structure of the resulting embedding

space using dimensionality reduction and clustering techniques, identifying the primary factors that influence a section's position (e.g., article theme vs. section type).

- **Section Type Classification:** To develop and evaluate multiple models—including heuristics and large language models for classifying each section according to its rhetorical function (e.g., 'Introduction', 'Methods', 'Results').
- **Fine-Grained Trend Analysis:** To leverage the classified sections to identify the specific theoretical frameworks and research methods used within the articles and to analyze their prevalence over the 20-year history of the journal.

3 Article Splitting and Embedding

The initial splitting and embedding were in many ways straightforward. Despite being spread over 20 years of publishing, the data had a predictable XML structure. The extracted data could then be passed to an embedding function to yield a vector representation. The article splitting algorithm resulted in 7313 sections from the 1222 articles.

We chose to use the closed source "voyage-large-2" model from Voyage AI for our initial embedding, choosing a 1024 dimensional output. At the time of writing, Voyage AI's models are considered to provide the best embeddings for general purposes. We therefore chose one of their models for our initial embeddings that were to be used for general data exploration. A viable open-source alternative would be Jina AI's embedding models.

To see the implementation of the data extraction and embedding, see the "pre-processing.ipynb" computational notebook.

3.1 Transformer Fine-tuning

In recent years, transformer-based language models have become the industry standard for downstream classification tasks, typically used in combination with a final set of classification layers, fine-tuned on domain-specific datasets. In our analysis we employ BERT (Devlin et al., 2019), a sophisticated LLM utilizing attention mechanisms introduced by Vaswani et al. (2023) to capture context-specific representations of textual data. More precisely we use SciBERT, a BERT-based language model trained on a large corpus of scientific publication, offering state-of-art performance on downstream scientific NLP tasks (Beltagy et al., 2019). Fine-tuning is done by adding task-specific layers of classification heads, iteratively adjusted via supervised learning. This is implemented using Hugging Face's Transformer library (Wolf et al., 2020).

3.2 Multi-layer Perceptrons

Multi-Layer Perceptrons (MLPs) are a class of feed-forward neural networks generally consisting of an input and output layer sandwiching one or several hidden layers (Hornik et al., 1989). Layers consist of a finite number of nodes where each node is fully connected to the previous and subsequent layer, with individual neuron outputs dictated by non-linear activation functions. Despite their simple structure, MLPs are capable of modeling complex non-linear relationships (Hornik et al., 1989) and are widely used for classification tasks. In this paper, MLPs are used as components in custom hybrid classification models.

4 Initial Exploration of the Embedding Space

4.1 Methods for Visualization: Dimensionality Reduction

To visualize and interpret the high-dimensional embedding space, we use *dimensionality reduction* techniques. These methods project the data into lower dimensional spaces (e.g. 2-dim), aiming to preserve the data's most essential structural features. We employed three common techniques.

Principal Component Analysis (PCA) is a linear technique meant to preserve as much global variance as possible. It creates a reduced dataset with new, fewer variables (principal components). When used for 2D visualization, the two principal components that capture the most variance are used as the axes (IBM, n.d.).

T-distributed Stochastic Neighbour Embedding (t-SNE) is a non-linear approach that excels at preserving the local structure of the data. It focuses on keeping neighboring data points close to each other in the lower-dimensional projection, making it highly suitable for identifying clusters (Mazraeh, 2025).

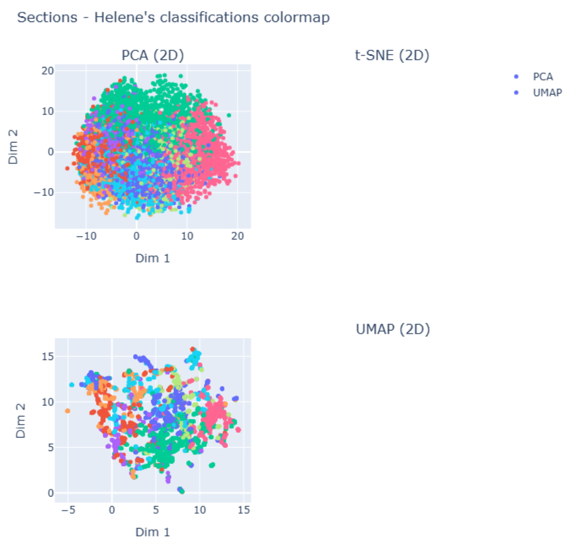
Uniform Manifold Approximation and Projection (UMAP) is another non-linear approach that balances the preservation of both local and global structure. It is often faster than t-SNE and can be more effective at representing the overarching shape of the data alongside local clusters (Coenen & Pearce, n.d.).

4.2 Methods for Discovery: Clustering

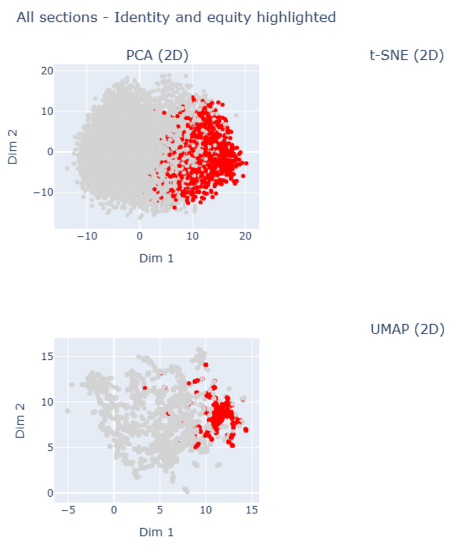
To programmatically identify groups in our unlabeled data, we use *clustering* algorithms. This unsupervised learning technique groups similar vectors based on their position in the embedding space.

K-Means partitions the data into a pre-defined number of clusters (k), assigning each data point to the cluster with the nearest mean or 'centroid'.

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) creates clusters based on data density. It identifies regions where points are tightly packed,



(a) Sections colored by article theme (from Lane's analysis).



(b) Sections from "identity and equity" articles highlighted.

Figure 1: PCA and UMAP plots of embedded sections, showing strong clustering by article theme.

making it particularly effective for oddly shaped clusters and for identifying noise points that don't belong to any cluster (Stewart & Al-Khassaweneh, 2022).

4.3 Initial Findings from Data Exploration

To get a general impression of the embedding space, we performed some initial clustering and dimensionality reduction. Our primary focus was to see if sections could be distinguished by their function (e.g., "Methods", "Theory"), which would inform our subsequent classification task.

First, we explored how section embeddings relate to the overall theme of their parent article, using the thematic categories from Lane's analysis. In Figure 1, we use PCA and UMAP to visualize the sections, coloring them by theme. As is evident from the distinct groupings, a section's location is *heavily dominated by its article's overarching theme.* This confirms that a section-level approach is needed to analyze intra-article structure.

Next, we investigated the internal structure within articles. Figure 2 shows a PCA plot of all sections, with sections from 9 randomly sampled articles highlighted. While sections from the same article cluster together, there is meaningful variation between them. This

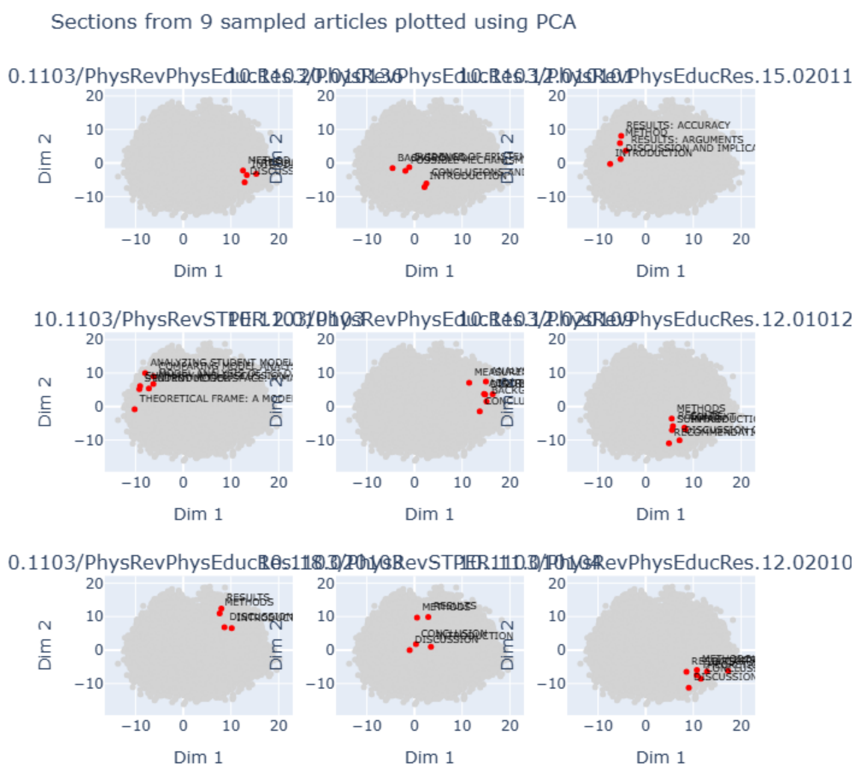


Figure 2: PCA reduction of all sections with 9 randomly sampled articles highlighted.

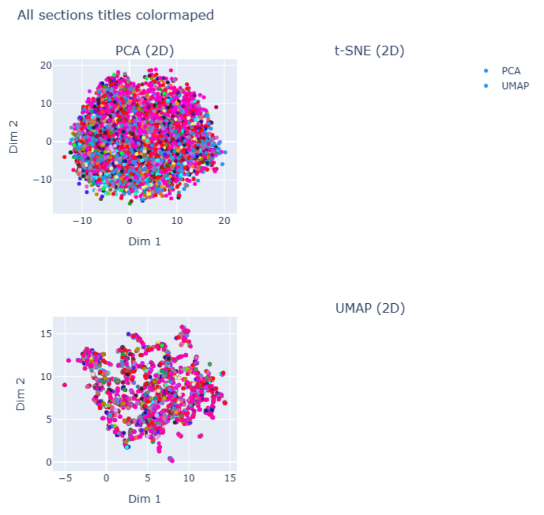
intra-article spread gives us hope that we can exploit this structure for classification.

Finally, we attempted to find natural groupings based on section type using clustering algorithms (K-Means and HDBSCAN), shown in Figure 3. While the algorithms produced clusters, these groupings did not correspond to discernible section types. This finding underscores the *need for more sophisticated, supervised classification methods*, which we explore in the following section.

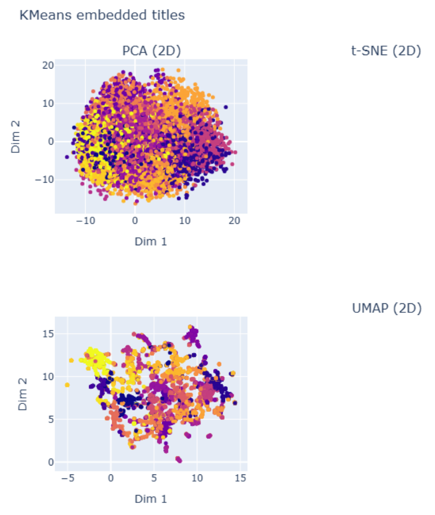
4.4 Summary of Exploratory Findings

Our initial exploration of the section embeddings yielded three crucial insights that shaped the subsequent direction of this project.

First, and most importantly, we found that the *overarching theme of an article is the dominant factor* determining the location of its sections in the embedding space. Sections from articles about "quantum physics" cluster together, far from sections from articles about "identity and equity". This presents a significant challenge, as the thematic signal



(a) K-Means clustering.



(b) HDBSCAN clustering.

Figure 3: A comparison of clustering algorithms applied to section embeddings. The resulting clusters did not align with section types.

could easily overwhelm the more subtle signal of a section's function when attempting to classify section types.

Second, despite this strong thematic clustering, we observed a *meaningful intra-article structural variance*. Sections within the same article are not identical points; they occupy a distinct region in the space, which suggests that their different functions can potentially be distinguished. This finding provides the basis for our project's feasibility.

Finally, our attempts to use unsupervised clustering algorithms to automatically group sections by type (e.g., all 'Methods' sections) were unsuccessful. This demonstrates that while structural differences exist, they are not easily separable.

This conclusion motivates the work of the following chapter, where we develop and evaluate supervised models specifically designed for this classification task.

5 Section Type Classification

To analyze theoretical and methodological trends in the PRPER corpus, we first had to classify each of the 7313 sections by its section type. Since many sections have ambiguous or non-standard headings, a simple keyword search is insufficient. We therefore developed and evaluated several classification methods.

This section details that process. First, we describe the benchmark dataset used to evaluate performance. Second, we present the different classification models we tested: a simple weighted heuristic method, a fine-tuned BERT model, and a large language model (LLM) approach using the OpenAI API. Finally, we discuss the trade-offs between these methods and justify our choice of model for the subsequent analysis.

5.1 Benchmarking

To make meaningful judgments about the accuracy of each approach, we generated two labeled benchmarks. The main dataset was generated by sampling 22 articles with a total of 127 sections and hand-labeling them by their type. This provided a representative, albeit small, set of sections covering a broad range of types.

To supplement this, we generated a larger secondary dataset of approximately 500 sections by selecting those with unambiguous headings (e.g., ”Methods,” ”Introduction”). While this set is easier to classify, it allowed us to test our models on a more robust number of samples. For evaluation, we developed a function to compute overall accuracy, per-category metrics, and a confusion matrix to diagnose errors.

5.2 Classification with Heuristics

As a baseline, we developed a simple heuristic classifier. We expected that a section’s type could be inferred from features like keywords in its title, keywords in its content, its relative length, and its position within the article. We combined these features into a weighted function to predict the section type. After tuning, we found that the section title was by far the most powerful predictor, with an optimal weight of 0.8. The heavy reliance on this single feature indicates that other signals like section content or length were not, on their own, strong predictors. Consequently, while this heuristic model performs reasonably on sections with standard headings (e.g., ”Introduction,” ”Methods”), it is inherently brittle and fails when encountering the many sections with more creative or non-standard titles.

5.3 Fine-tuned SciBERT LLM

5.3.1 Implementation

To evaluate the effectiveness of transformer-based language models for scientific section classification, we fine-tune SciBERT on a subset of our dataset. To construct a high-confidence training/testing dataset, sections were grouped based on title (see table 1), resulting in subset of 2046 samples spread equally over 6 separate classes.

Label ID	**Section Titles (Canonical Variants)**
0	INTRODUCTION
1	METHODS, METHODOLOGY, METHOD
2	DATA AND RESULTS, RESULTS, FINDINGS
3	DISCUSSION, DISCUSSION AND CONCLUSIONS
4	CONCLUDING REMARKS, CONCLUSION, CONCLUSIONS
5	THEORY, BACKGROUND, THEORETICAL FRAMEWORK, THEORETICAL BACKGROUND, LITERATURE REVIEW

Table 1: Canonical section title groups used for high-confidence label assignment.

Fine-tuning was implemented with the Huggingface Transformers library, initializing the model using the ’allenai/scibert_scivocab_uncased’ checkpoint and ’AutoModelForSequenceClassification’ class. Tokenization was handled using the corresponding ’AutoTokenizer’, truncated to a maximum of 512 tokens and padded to ensure homogeneous data length. Since we are only doing transfer learning, the embedding and encoder model layers were frozen, relying on iterative adjustment of the final pooler layers to aggregate model output into an interpretable classification output vector. The final model was optimized using a 30-35-35 training-testing-validation-split to minimize overfitting, with hyperparameters

$$l_r = 2 \cdot 10^{-4}, \quad \text{Batch Size} = 8, \quad \text{Epochs} = 10.$$

(see notebook ’bert_section_classifier.ipynb’ for full implementation)

5.3.2 Results and Discussion

The model was trained locally on an NVIDIA GPU over the span of ∼7 hours. Key training and evaluation metrics (summarized in table 2) show a steady decrease in training and evaluation loss from 1.3488 and 1.0107 after the first epoch to 0.5358 and 0.5754 after epoch 10, respectively. Model accuracy on evaluation data showed a corresponding increase from 0.638 to 0.769 during the same period, suggesting that the model is learning generalizable patterns rather than overfitting to the data. Notably, the metrics also show a clear plateau around epoch 7-8, suggesting that further mode convergence does not result in any noticable performance improvements. Reducing training to 8 epochs may therefore save time and computational resources without adversely affecting model classification capabilities.

Epoch	Train Loss	Eval Loss	Eval Accuracy	Learning Rate	Grad Norm
1	1.3488	1.0107	0.638	1.80e-4	5.63
2	0.9062	0.8415	0.668	1.60e-4	5.11
3	0.7612	0.7020	0.717	1.40e-4	4.19
4	0.6701	0.6754	0.746	1.20e-4	6.02
5	0.6338	0.6465	0.749	1.00e-4	3.16
6	0.6026	0.6249	0.752	8.01e-5	3.58
7	0.5862	0.6030	0.772	6.01e-5	3.16
8	0.5716	0.5855	0.772	4.01e-5	3.72
9	0.5560	0.5835	0.765	2.01e-5	3.34
10	0.5358	0.5754	0.769	1.12e-7	5.54

Table 2: Training and evaluation metrics for SciBERT across 10 epochs.

Table 3 presents model evaluation on validation data. Overall accuracy was found to be 0.772. The precision and recall of 0.791 and 0.772, respectively, and the resulting F1-score of 0.776, indicate balanced false-positive and false-negative across the classes. The log loss of 0.646 suggests relatively high model confidence when making correct choices and large uncertainty when making incorrect ones. This is further reinforced by an average confidence of 0.750 and average entropy of 0.645, implying (generally) sharply focused model prediction with low uncertainty and diffuseness. Examining Table 4 and Figure 4 reveals average model performance on sections ”Introduction”, ”Conclusion” and ”Discussion”, with higher predictive ability on sections ”Methods” and ”Results” and a rather lackluster F1-score of 0.629 on ”Theory” sections. Figure 4 indicates that this performance drop is a concequence of frequent misclassification of ”Theory” sections as ”Introduction”, and vice versa. Interestingly, the matrix shows elevated numbers for diagonal-adjacent tiles, suggesting that false label assignments primarily occur between ”bordering” article sections. This hints at the fact that articles trace a continuous ”semantic path” from beginning to end, resulting in textually adjacent sections being more closely related (and therefore harder to distinguish). We will explore this characteristic more thoroughly in later analysis.

Table 3: Overall Evaluation Metrics on Validation Data

Metric	**Value**
Accuracy	0.772
Precision	0.791
Recall	0.772
F1-score	0.776
Log Loss	0.646
Average Confidence	0.750
Average Entropy	0.645

Table 4: Per-Class Evaluation Metrics

Class	**Precision**	**Recall**	**F1-score**	**Support**
Introduction	0.759	0.721	0.739	61
Theory	0.550	0.733	0.629	45
Methods	0.933	0.808	0.866	52
Results	0.785	0.911	0.843	56
Discussion	0.809	0.731	0.768	52
Conclusion	0.906	0.707	0.795	41

5.4 Projection Based Classification

While fine-tuning a language model is effective, it is often computationally demanding, time-consumimg, and unecessary complex for simple og low-resource applications. Leveraging the geometric properties of textual embeddings, we iteratively develop and implement a lightweight classification approach that offers a fast, interpretable and data-efficient alternative to full fine-tuning. To maintain alignment with prior benchmarks, we rely on the same six-category classification dataset as used in SciBERT fine-tuning. All data are encoded using the text-embedding-3-large model from OpenAI.

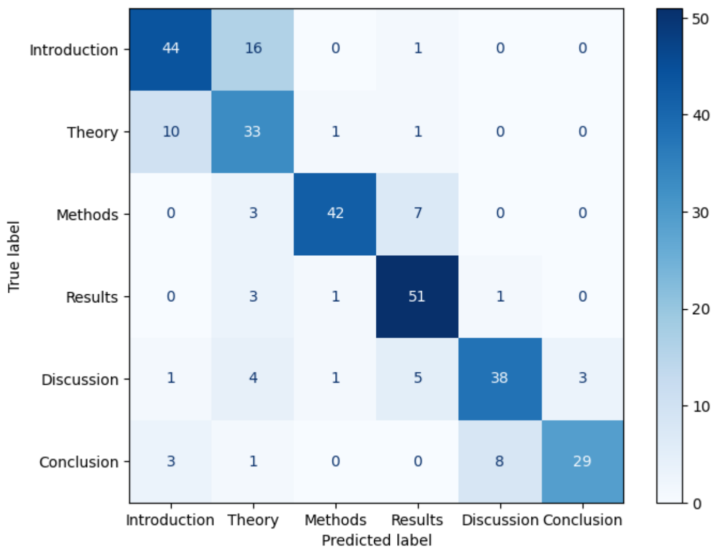


Figure 4: Confusion Matrix outlining model performance on validation split.

5.4.1 Implementation

Mean Embedding Similarity Given a small number of labeled samples per class, we calculate class wise means represent the prototypical embedding of each class in the semantic space. These mean vectors serve as centroids, or ”semantic signatures”, and summarize the contextualized representations of each class. The similarity of a given sample x to a given class is calculated using the inner product

$$\text{sim}(x, \mu_c) = \langle x, \mu_c \rangle \tag{1}$$

where μ_c is the respective, normalized, class mean. The resulting score can be interpreted as a measure of sample-class alignment. Returning to the data-subset used for fine-tuning previously, we create a new dataframe containing only instances of ”Introduction” and ”Results” (arbitrarily chosen). Calculating an ”Introduction” mean, we associate a score with each sample using (1). Sorting the data by this score, we see a clear class seperation (see figure 5 (2)).

Class-centroid Difference Scoring Given last paragraphs inerpretation of class centroids as being prototypical representation of each class, it seems intuitive that the difference vector $v = \mu_2 - \mu_1$ between two class centroids μ_1 and μ_2 should somehow ”encode” the semantic difference between the classes. In other words, v captures how the meaning of class 1 diverges from that of class 2 in the embedding space. It might therefore also

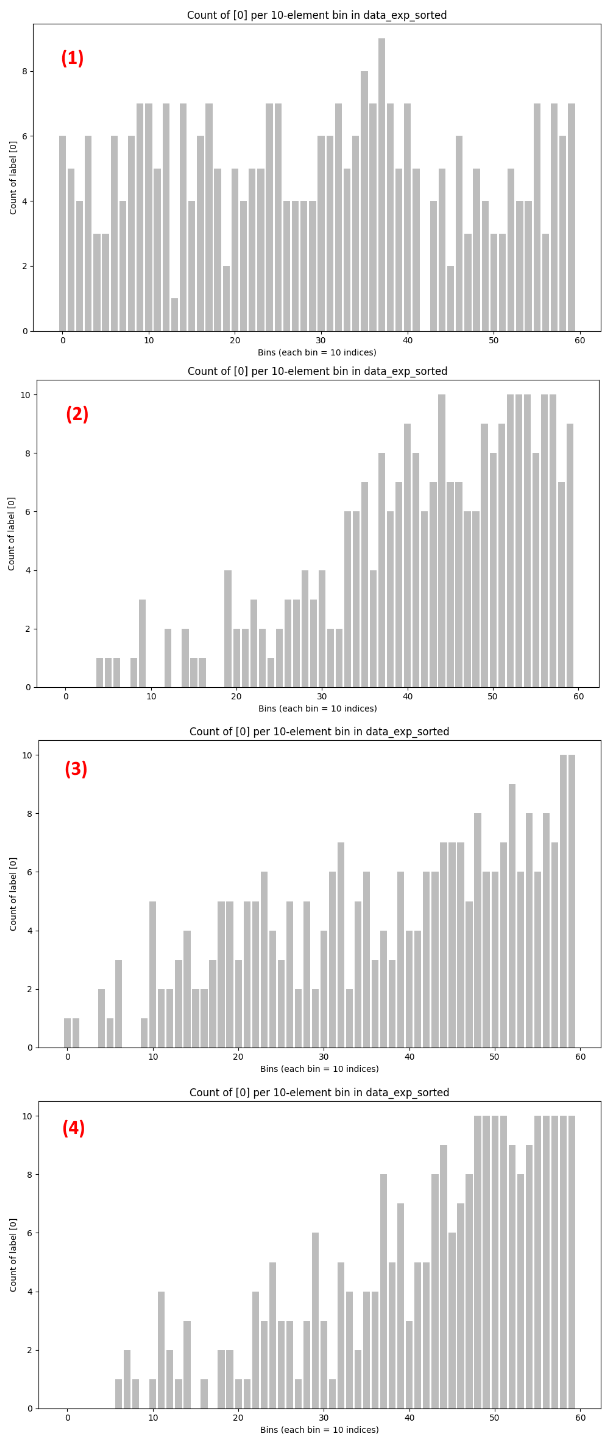


Figure 5: Distribution of labels in dataset. A bin along the x-direction represent a 10 consecutive data entries, with the number along the y-axis indicating percentage of label 0 among the entries.

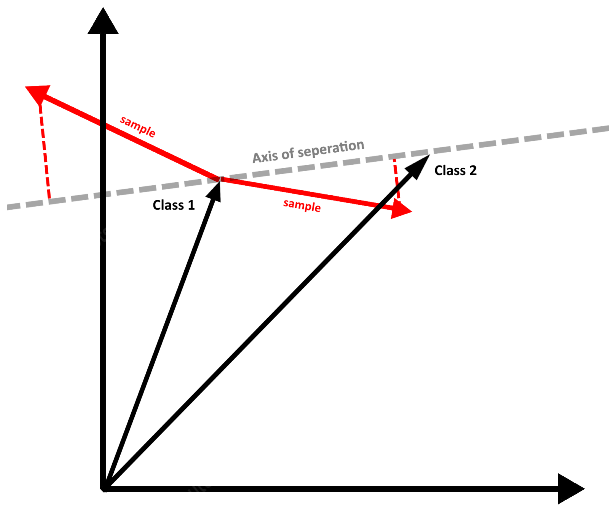


Figure 6: Geometric visualization of the centroid difference vector and the projection upon it.

be interesting to get a measure of where a sample embedding falls along this seperation axis. For any sample x, we take its projection onto the centroid difference vector (see figure 6)

$$\mathrm{proj}_v(x) = \frac{\langle x, v\rangle}{\langle v, v\rangle} v = \langle x, v\rangle\, v = k \cdot v, \quad k = \langle x, v\rangle \tag{2}$$

where we assume that the difference vector v has already been normalized $|v| = 1$. In the end result v is independent of x, and we are left with the inner product k being the metric of interest, indicating the position of a sample projection along the seperation axis, ranging from $k = -\infty$ (really far in direction of Class 1) to $k = \infty$ (really far in direction of Class 2). In the end we are left with a "seperation" score

$$\mathrm{sep}(x, v = \mu_{c2} - \mu_{c1}) = \langle x, v\rangle \tag{3}$$

Testing again on the aforementioned "Introduction/Results" dataset, we calculate a centroid difference vector and associate a "seperation" score with each sample. Sorting the data by this score, we again see a clear class seperation (see figure 5 (3)).

Combined Scores and Binary Model Implementation These scores are in a sense complementary, with "similarity" scores measuring sample alignment with centroids and "seperation" scores measuring sample-centroid displacement. Combining the two into a single, combined, score

$$\text{score} = \lambda \cdot \text{sim}(x, \mu_{c1}) + (1 - \lambda) \cdot \text{sep}(x, \mu_{c2} - \mu_{c1}) \tag{4}$$

where score weighting is adjusted by varying λ. Arbitrarily choosing $\lambda = 0.5$ and sorting the same data as before by this new score, we seemingly obtain the best class separation yet (Figure 5 (4)). Using this combined score, we implement a binary classification model and test it on all $\binom{6}{2}$ binary 6-class combinations (still testing on the same 2046-sample dataset used previously) using a 0.7/0.3 train-test-split. The decision threshold for assigning binary labels is defined as the median of the score distribution. We also evaluate model performance on the training split for a range of $\lambda \in [0, 1]$ to find optimal score-weighting.

Table 5: Summary of Binary Classification Metrics Across All Class Pairs

Statistic	**Accuracy**	**Precision**	**Recall**	**F1-score**
Mean	0.902	0.904	0.900	0.901
Maximum	0.976	0.989	0.990	0.974
Minimum	0.732	0.755	0.705	0.729

Table 5 shows a generally strong and balanced model performance above 0.900 across the board, with maximum performance in the high 0.900s. Introduction/Theory classification is however an outlier, with values in the low 0.700s. This comes as no suprise given our previous findings with the fine-tuned SciBERT model.

(see notebook "nonMLP_classifier.ipynb" for full implementation)

5.4.2 Full Multi-Class Classification Model

Non-MLP Implementation Multi-class classification ability is handled using a One-vs-One (OvO) approach, which involves fitting a distinct binary classifier to all $\binom{n_{\text{classes}}}{2}$ class combinations. Each binary classifier in the ensemble operates using the lightweight scoring approach outlined above, calculating class means, combined scores according to (4), with individually optimized λ-weighting. When applied to a sample, each constituent classifier casts a binary vote, and final model prediction is aggregated using a majority

Table 6: Overall Evaluation Metrics for Non-MLP Classifier

Metric	**Value**
Accuracy	0.713
Precision	0.715
Recall	0.713
F1-score	0.713
Log Loss	1.275
Average Confidence	0.331
Average Entropy	1.494

Table 7: Per-Class Evaluation Metrics

Class	**Precision**	**Recall**	**F1-score**	**Support**
Introduction	0.568	0.636	0.600	118
Methods	0.867	0.875	0.871	104
Results	0.787	0.842	0.813	101
Discussion	0.646	0.634	0.640	101
Conclusion	0.767	0.702	0.733	94
Theory	0.679	0.594	0.633	96

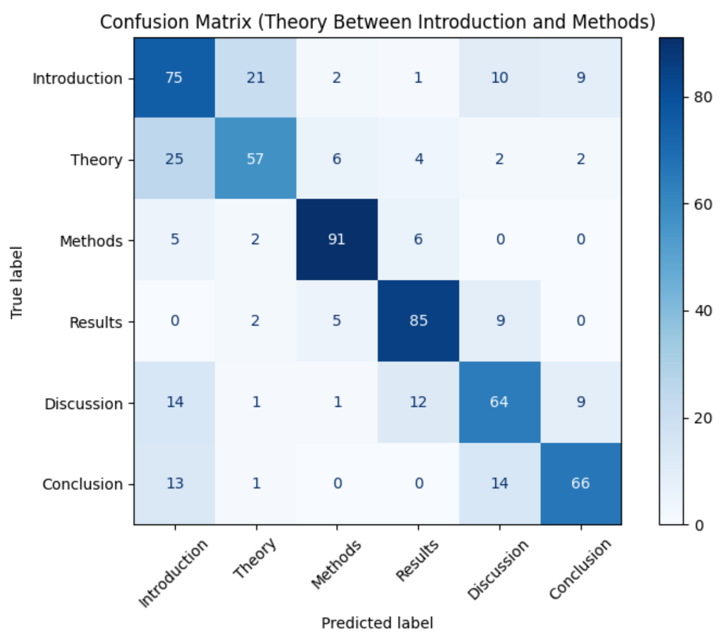
vote. The model was fitted to the data, taking sub-5 seconds. It was then evaluated on the test split using key metrics, presented in Table 6, Table 7 and Figure 7.

(see notebook 'nonMLP_classifier.ipynb' for full implementation)

MLP Implementation To further enhance model performance, we replace the majority voting system with a multi-layer perceptron (MLP), trained to map the combined projection scores to class logits. Using the same OvO-approach, the resulting scores are fed into an MLP with input dimensions $2 \cdot \binom{n_{\text{classes}}}{2}$. That is, weighting using the aforementioned λ is discarded in favour of directly inputting all scores into the neural network. The input layer is succeeded by three ReLU-activated hidden layers, gradually consolidating the input into fewer dimensions, resulting finally in an output layer producing a single logit for each class.

for a range of different hyperparameters (see notebook 'mlp_classifier.ipynb' for more details), optimal performance on the data was found for the following parameters

$$l_r = 5 \cdot 10^{-4}, \quad \text{Hidden Layer Dimensions} = [32, 16, 8], \quad \text{Epochs} = 200.$$



True \ Predicted	Introduction	Theory	Methods	Results	Discussion	Conclusion
Introduction	75	21	2	1	10	9
Theory	25	57	6	4	2	2
Methods	5	2	91	6	0	0
Results	0	2	5	85	9	0
Discussion	14	1	1	12	64	9
Conclusion	13	1	0	0	14	66

Figure 7: Confusion Matrix outlining non-MLP model performance on test split.

Model fitting completed in under 10 seconds. Resulting model performance is outlined in Table 8, Table 9, Table 10 and Figure 8.

Table 8: Summary of Binary Classification Metrics

Statistic	**Accuracy**	**Precision**	**Recall**	**F1-score**
Average	0.902	0.903	0.905	0.903
Maximum	0.985	0.991	1.000	0.986
Minimum	0.751	0.737	0.771	0.767

5.4.3 Model Comparison and Discussion

The projection classification model offer a lightweight alternative to full fine-tuning. From Table 5, Table 6 and Table 7 we see that the non-MLP variant performs well on binary classification tasks (both accuracy and F1 score above 0.9), but multi-class classification performance is significantly worse (accuracy and F1-score of 0.713) class wise performance even lower (F1-score of 0.600) on ”Introduction” sections. Looking at Figure 7 it is evident that the model suffers from the same diagonal-adjacent class mislabeling, but in addition it also frequently misclassifies ”Introductions” as ”Discussion/Conclusion” and vice versa. The implementation also suffers from a large flaw; the decision threshold from label assignment is based on the median of the score distribution. This results in a model that is extreme sensitive to class imbalances in the training split. To examine this

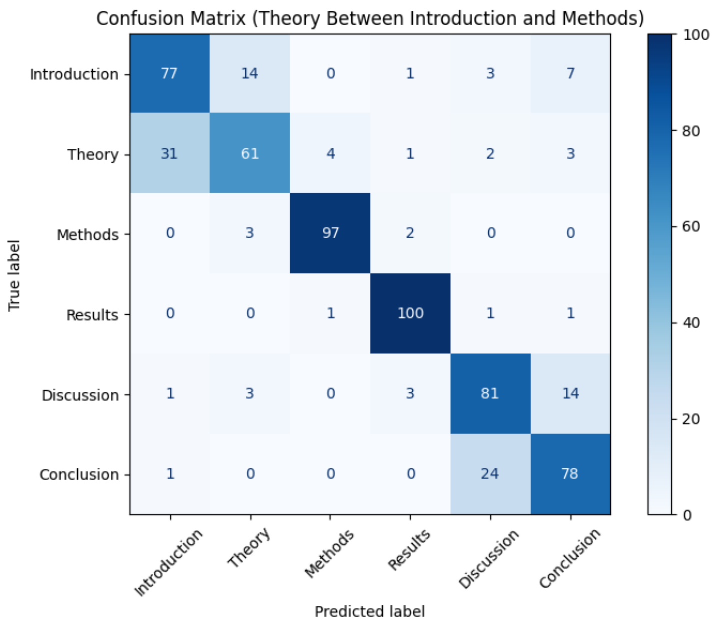


Figure 8: Confusion Matrix outlining MLP model performance on test split.

Table 9: Overall Evaluation Metrics for MLP Classifier

Metric	**Value**
Accuracy	0.805
Precision	0.804
Recall	0.805
F1-score	0.803
Log Loss	0.529
Average Confidence	0.786
Average Entropy	0.552

we test binary classification performance on ”Introduction” and ”Methods” sections for differing label imbalances. A 1:1 label relation results in a classification performance of ∼0.95. This number decreases to ∼0.84 for a label imbalance of 2:1, and increasing the discrepancy to 4:1 results in a performance drop to ∼0.73.

The MLP-approach replaces this fixed threshold strategy with a trainable component capable of learning non-linear relationships between class score-pairs. Table 8 shows that this does not seem to increase binary classification performance, with very little improvement in mean (though minimum performance does increase from an F1-score 0.729 to 0.767). Comparing Table 6 and Table 9 shows significant improvements in multi-class performance, with accuracy and F1-scores increasing from 0.713 and 0.713 to 0.805 and 0.803, respectively. Average model confidence also increase by almost 2.5x, while average

Table 10: Per-Class Evaluation Metrics for MLP Classifier

Class	**Precision**	**Recall**	**F1-score**	**Support**
Introduction	0.700	0.755	0.726	102
Theory	0.753	0.598	0.667	102
Methods	0.951	0.951	0.951	102
Results	0.935	0.971	0.952	103
Discussion	0.730	0.794	0.761	102
Conclusion	0.757	0.757	0.757	103

entropy is cut by 2/3 the original value (though these gains do come at the cost of intuitiveness/interpretability). Looking at Figure 8 we see that misclassifications are now limited almost completely to ”Introduction/Theory” and ”Discussion/Conclusion”. We also tried to train several MLPs in parallel, mimicking a sort of weak learning approach, with final output being the mean of all estimators. Testing for $n_{\text{estimators}} \in [1, 2, 4, 8, 16]$ over a range of hyperparameters, we saw very-slight to nonexistent performance improvements ($\sim$ 1%) on our data, indicating that this does not yield any significant advantage.

Comparing these projection models with the fine-tuned SciBERT model we see that the MLP-variant outperforms the fine-tuned LLM, with higher accuracy (0.805 v. 0.772), F1-score (0.803 v. 0.776), higher confidence (0.786 v.0.750) and lower entropy (0.552 v. 0.645). While not as fast as the non-MLP model (sub 5-second training time), the MLP model delivers performance rivaling the fine-tuned LLM at a fraction of the computational cost and time (sub 10-second v. $\sim$7hrs). It is also more data-efficient (though the MLP-variant is less so), primarily utilizing few shot sampling of 8 examples per class. Figure 9 shows that the means calculated using n samples converges to the global mean using all samples very quickly, suggesting that sampling only a few examples does not significantly impact performance. Testing on the models reflects this fact; increasing the number pof samples n results only in very slight performance gains ($\sim$1%) on our data. This indicates that these projection models are a lightweight alternative to full-on model fine-tuning, offering competitive performance with greatly improved computational- and data efficiency.

While these projection models do not utilize computationally intensive transformers, they depend on the existence of high quality embedding models. In fact, the models

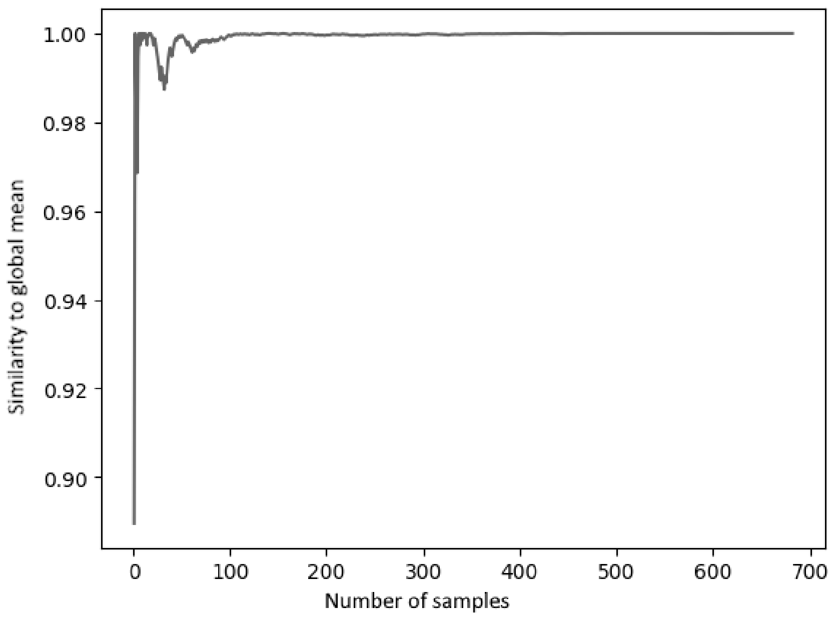


Figure 9: Inner product of normalized average of n embeddings with global average. We see that it quickly approaches 1.

are very sensitive to the quality of the embeddings. The data, instead embedded using the 'Voyager-3-large' embedding model, immediately causes a sharp performance drop of $\sim 7\%$ (relative). This approach is lightweight and quick only if embeddings are precomputed; in other words, it is still implicitly dependent on large, compute- and data hungry models, leading the real computational cost to be much higher than these results give the impression of. This juxtaposition risks completely invalidating the model's original motivation and conceptual justification. It also raises another important point; while the projection models utilize cutting-edge, modern embedding models, the SciBERT implementation uses its own aging tokenizer, perhaps resulting in lower quality input and thereby rendering the comparison between the two unfair and biased.

5.4.4 2D visualization of score-pairs

Visualizing these score pairs seems an interesting prospect, perhaps yielding some new insight or intuition. Returning to the same dataset as earlier, we calculate the global mean embedding and the difference vector between the "Introduction"- and "Conclusion" mean centroids. Finding the aforementioned "similarity" and "seperation" scores of all section samples and plotting them as 2-dimensional points (Figure 10), we see clear clustering on a section-type basis. What is perhaps most interesting is how this contrasts with

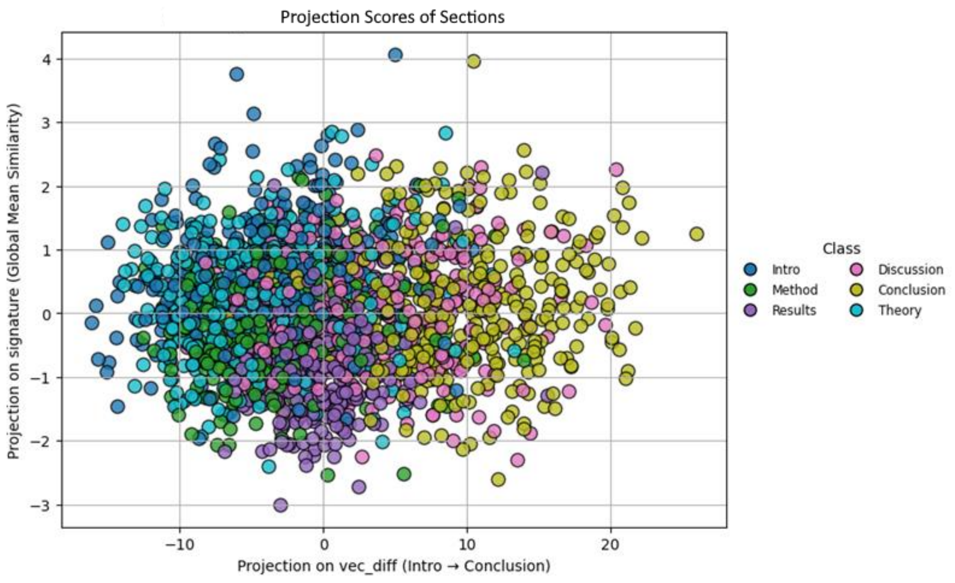


Figure 10: 2D projection scores of sections onto Introduction/Conclusion difference vector (x-axis) and global average embedding (y-axis).

earlier findings; using other dimensional-reduction techniques, sections seemed to display thematically based clustering. This plot seems to indicate the opposite (perhaps this is caused by the fact that the specially selected ”Introduction/Conclusion” difference vector specifically ”picks out” the sample-embedding characteristics most similar to its own?). Instead plotting the means of ∼20 embeddings on a class by class basis (attempting to reduce semantic noice), we see a clear seperation of classes (see Figure 11).

Interestingly, the clusters trace out the semantically logical path from ”Introduction” to ”Conclusion” along the x-axis, indicating that the ”Intro/Conclusion” difference vector seemingly captures the entire semantic path of the article from beginning to end, with ”Introduction” and ”Conclusion” at each side, ”Theory” and ”Methods” very similar to ”Introduction”, and ”Results” and ”Discussion” progressively moving towards the right. The y-axis also shows ”Introductions” as being the most similar to the global mean and ”Results” as being the most divergent. Furthermore, the proximity of ”Introduction” and ”Theory” reinforces the earlier result that the sections-classes are closely related, given that they were often mistakenly interchanged by the classification models.

This visualization tool provides a different way of considering the projection scores, and will be used extensively during further analysis.

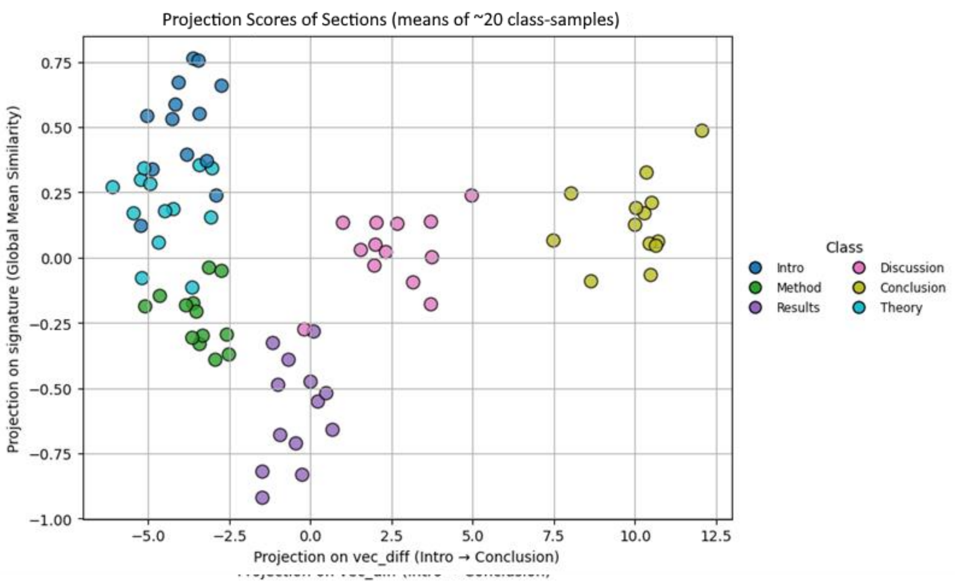


Figure 11: 2D projection scores of mean embeddings of 20 sections of same type onto Introduction/Conclusion difference vector (x-axis) and global average embedding (y-axis).

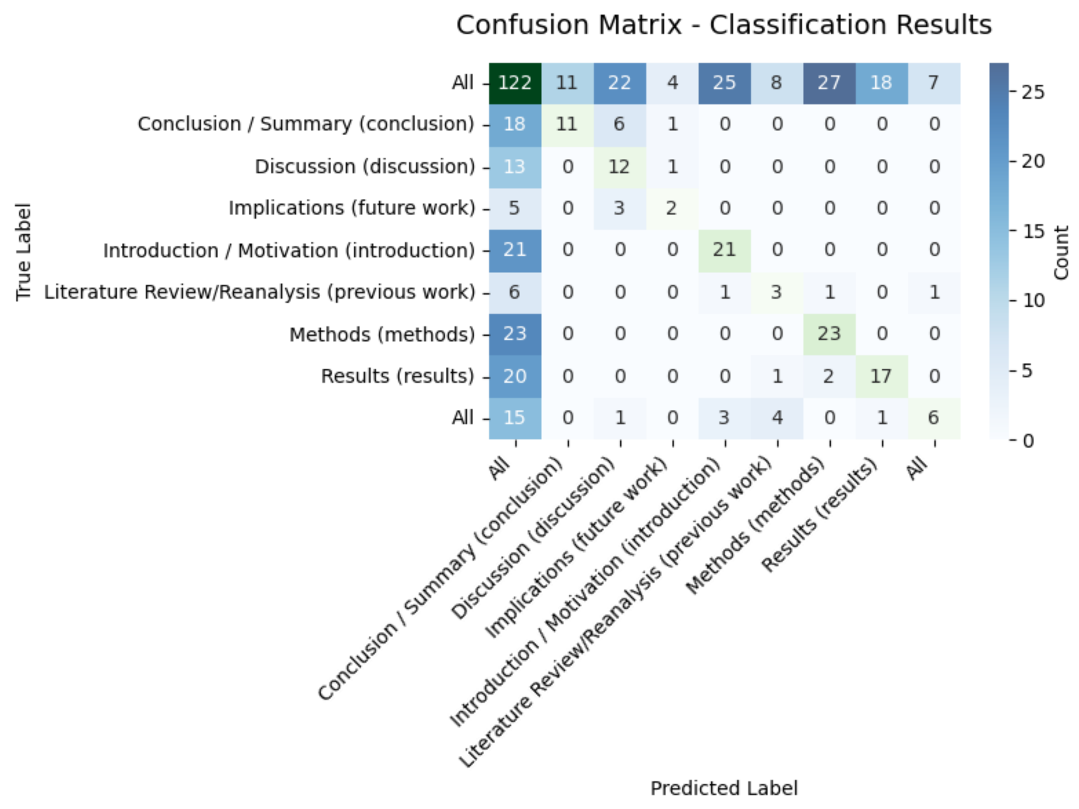
5.5 Classification with Large Language Models (LLMs)

We next tested a classification approach using OpenAI's API. After experimenting with prompts and context, we found the most effective and efficient method was to pass the entire text of an article at once, asking the model to return a JSON object classifying all of its sections in a single API call. This approach was approximately seven times faster and, in addition, more accurate than classifying sections individually.

Using this method with the 'gpt-4o' model, we achieved an accuracy of *90%* on our hand-labeled benchmark dataset. The model achieved near-perfect accuracy on the simpler heading-based dataset. The confusion matrix (Figure 12) reveals that most misclassifications occur between categories that are often conceptually similar, such as mistaking a 'Theoretical Framework' for a 'Literature Review', or confusing 'Implications' with 'Conclusion'.

5.6 Discussion of Classification Methods

Our experiments evaluated four distinct classification methods, revealing a clear trade-off between performance, computational cost, and implementation complexity. The final accuracy of each model on a comparable benchmark dataset is summarized in Table **??**.



	All	Conclusion / Summary (conclusion)	Discussion (discussion)	Implications (future work)	Introduction / Motivation (introduction)	Literature Review/Reanalysis (previous work)	Methods (methods)	Results (results)	All
All	122	11	22	4	25	8	27	18	7
Conclusion / Summary (conclusion)	18	11	6	1	0	0	0	0	0
Discussion (discussion)	13	0	12	1	0	0	0	0	0
Implications (future work)	5	0	3	2	0	0	0	0	0
Introduction / Motivation (introduction)	21	0	0	0	21	0	0	0	0
Literature Review/Reanalysis (previous work)	6	0	0	0	1	3	1	0	1
Methods (methods)	23	0	0	0	0	0	23	0	0
Results (results)	20	0	0	0	0	1	2	17	0
All	15	0	1	0	3	4	0	1	6

Figure 12: Confusion matrix for the LLM-based section classification on our hand-labeled benchmark dataset.

Table 11: Classification accuracy across different methods on the hand-labeled benchmark.

Classification method	**Accuracy**
Heuristics	∼0.70
Fine-tuned SciBERT	0.77
Projection-Based MLP	0.80
LLM (gpt-4.1)	∼0.90
LLM (gpt-4.1-mini)	∼0.80

The simple *heuristic model* served as a useful baseline but proved too brittle for this task. Among the locally run models, the fine-tuned *SciBERT* represents a standard deep learning approach, achieving a respectable accuracy of *0.77*, but at the cost of significant training time (∼7 hours). In contrast, our novel *projection-based MLP classifier* emerged as a powerful and lightweight alternative. It not only surpassed the SciBERT model with an accuracy of 0.80 but did so with a fraction of the computational cost (training in under 10 seconds) and greater data efficiency. Its primary dependency is the availability of high-quality, pre-computed text embeddings.

Finally, the *LLM-based approach* using an external API delivered the highest performance, with an accuracy of approximately ∼*0.90*. Its main drawbacks are the monetary cost of API calls and the reliance on a closed-source service, which is unsuitable for datasets with sensitive data.

Crucially, the errors made by the top-performing models often highlight genuine ambiguity in the source material. Sections titled ”Discussion and Conclusion” are conceptually overlapping, and the line between a ”Literature Review” and a ”Theoretical Framework” can be blurry even for a human reader. This suggests that the ∼*0.90* accuracy achieved by the LLM is approaching the practical ceiling for this classification task with this set of discrete categories.

Given its state-of-the-art performance and our goal of obtaining the highest quality classifications for our subsequent trend analysis, we selected the *LLM-based classifier* for processing the entire dataset. However, our results show that the projection-based MLP is a highly compelling local alternative, offering competitive performance with minimal computational overhead.

5.7 Exploring Inter-Section Vector Arithmetic

Inspired by classic word embedding analogies like ‘king - man + woman ≈queen‘, we briefly explored whether similar vector arithmetic could model the transitions between section types (e.g., from ‘Theory‘ to ‘Methods‘). Our initial, naïve attempt involved calculating a ’transition vector’ by subtracting a theory section’s embedding from its corresponding methods section’s embedding.

This simple implementation did not yield clearly predictive results. We suspect this is because the powerful 'thematic signal' of an entire article can easily obscure the more subtle 'section type signal' that defines a section's function. While a more sophisticated approach might yet prove fruitful, a deeper investigation of this specific technique fell outside the scope of the current project.

6 Content Analysis with an Iterative LLM: Identifying Theories and Methods

Once the articles were classified into section types, we were well-poised to perform a more fine-grained analysis. Our goal was to move beyond broad topics and identify the specific theoretical frameworks and research methods used within the PRPER corpus, and to track how their usage has evolved over time. To achieve this, we developed a novel, iterative classification workflow that uses an LLM to discover and assign fine-grained categories directly from the text.

6.1 Methodology: An Iterative Classification Workflow

To extract specific information like theoretical frameworks, we developed a staged, iterative approach that uses an LLM to build a classification scheme from the ground up. As illustrated in Figures 13 and 14, the process involves:

1. Feeding a sample of text sections to an LLM to generate an initial list of categories.
2. Using this list to classify all sections.
3. Reviewing the classifications and having the LLM refine or merge the categories.
4. Repeating the process until the classification scheme is stable and comprehensive.

This logic is implemented in a flexible Python ‘AbstractClass‘. A key feature of this implementation is the format of the LLM’s response. For each section, the model returns both a direct array of classifications (e.g., the specific frameworks it identifies) and a corresponding probability distribution. This provides analytical flexibility, allowing us to either accept the model’s direct classifications, filter results by a confidence threshold, or select only the single highest-probability result.

6.1.1 Identifying and Classifying Theoretical Frameworks

We first applied this workflow to the 589 sections classified as ‘Theoretical Framework‘. The process generated a fine-grained list of 135 distinct frameworks. To make this data more interpretable for trend analysis, we ran the classifier on the framework list itself to group them into six meta-categories (e.g., *Cognitive, Sociocultural, Social Justice*).

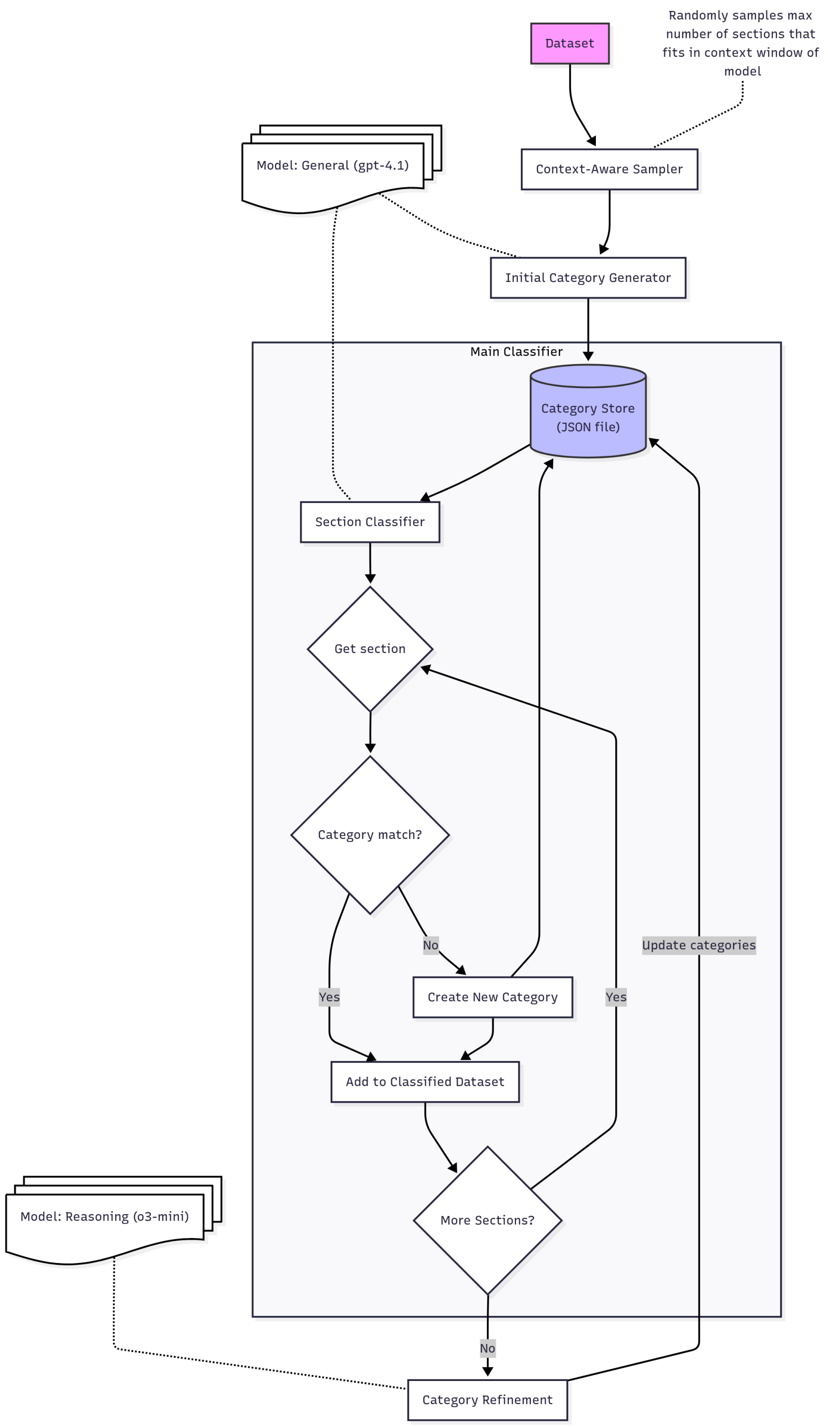


Figure 13: Abstract logic of the iterative LLM classifier.

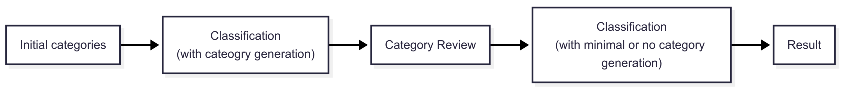

Initial categories
Classification (with cateogry generation)
Category Review
Classification (with minimal or no category generation)
Result

Figure 14: General workflow of how we used the classifier to generate results.

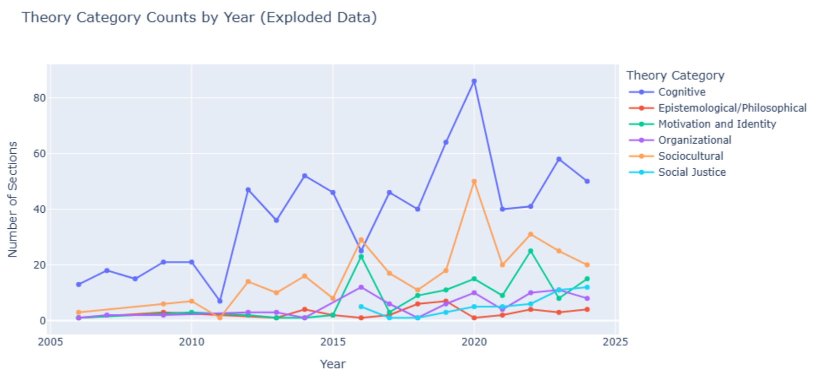


Figure 15: Theoretical developments in PRPER over time, grouped by meta-category.

Using these meta-categories, we were able to plot the evolution of theoretical interests in PRPER over time, as shown in Figure 15.

6.1.2 Identifying and Classifying Methods

We replicated the same process for all sections classified as ‘Methods‘. As we had no predefined meta-categories for research methods, we used the classifier’s automatic category generation feature to produce an initial list for review. This allowed us to meta-categorize the identified methods and plot their development over time.

6.2 Discussion of the Iterative Classifier

This iterative, LLM-based approach has several significant advantages. Rather than relying on a predefined list, the classifier discovers emergent theoretical and methodological categories directly from the corpus, making it a powerful tool for large-scale, qualitative discovery. Furthermore, its implemented as a flexible software framework. The classifier is built upon a set of abstract Python classes that define the core iterative workflow. This design allows for easy extension to new tasks; one can create a novel classifier, for example to identify research questions, simply by inheriting from a base class and providing new, task-specific prompts. The framework also uses Pydantic models to ensure that all data exchanged with the LLM is structured and valid. For more technical details, see the module’s documentation.

However, the approach involves practical considerations and trade-offs. The iterative process, which involves multiple chained calls to an LLM, can incur monetary costs with APIs. As we noted earlier, running these tasks on powerful, local open-source models would solve the cost and data privacy issues but demands notable local computing resources. Furthermore, the ”black box” nature of LLMs means that the reasoning behind category creation is not always transparent, and the process requires a human-in-the-loop to review and validate the generated categories for quality and coherence.

A key challenge we noticed, particularly when classifying research methods, was that the model sometimes struggled to distinguish between the *use* of a method versus the *discussion* of one. For instance, some articles were classified as employing both qualitative and quantitative methods. While time constraints prevented a systematic manual review of these cases, we suspect that an article can *use* one approach while contrasting it with another; i.e. the LLM can struggle to distunguish between *use* and *discussion* of a method. This suggests the ‘gpt-4o-mini‘ model, while efficient, can lack the nuance to grasp this contextual distinction. We hypothesize that using a more powerful model could mitigate this issue, as it may be better at discerning an author’s primary methodology from the surrounding methodological discussion.

Despite these considerations, this method enables a novel form of automated literature review, allowing for a fine-grained analysis of conceptual trends at a scale that would be prohibitive to achieve manually.

7 Analysis using projections

The following sections outlines the use of the previously described 2D projection scoring to analyze, classify and interpret patterns among theoretical frameworks published in physics education research, utilzing the ChatGPT-sorted dataset containing ∼600 ”Theory” sections, divided into ∼ 130 seperate categories.

7.1 Category Grouping using Centroid Difference Vectors

Attempting to divide these ∼ 130 seperate categories into two superordinate classes, we identify two rough overarching categories

1. Cognitive, Conceptual and Mathematical Theories.
2. Critical Theories concerning Social Justice, Identity and related constructs.

Mean embeddings for each were calculated using the frameworks ”Philosophy of Science”, ”Mathematical Modeling”, ”Conceptual Change Theory”, ”Cognitive Load Theory” and ”Communities of Practice”, ”Critical Race Theory” and ”Sociocultural Identity Frameworks”, respectively. Taking the vector difference of these centroids and projecting all samples against it, we see a rough clustering (see Figure 16). The right hand side of the plot contains theories such as ”Culturally Relevant Pedagogy”, ”Gender Performativity” and ”Quantitative Critical Race Theory”, while the left side is mostly inhabited by different conceptual theories such as ”Conceptual Blending Theory” and ”Conceptual Metaphor Theory”, as well as ”Measurement Theory”, ”Epistemic Games” and ”Math-Physics Algorithmic Theory”. The middle ground is occupied by ”Activity Theory”, ”Argumentation Theory” and ”Intellectual Humility Framework” which perhaps do not fit very well into either category.

In an attempt to make this sorting more rigorous we trained the previously developed projection models on this ”Cogntive/Critical” theories difference vector and their respective class means. Applying the resulting classification model to the data, we saw a seemingly logical correlation between the theoretical framework and model classification of it as belonging more to either class 1 or 2 (see Table 12). In future work, it might be interesting to further explore whether a more sophisticated derivation of this method could be used

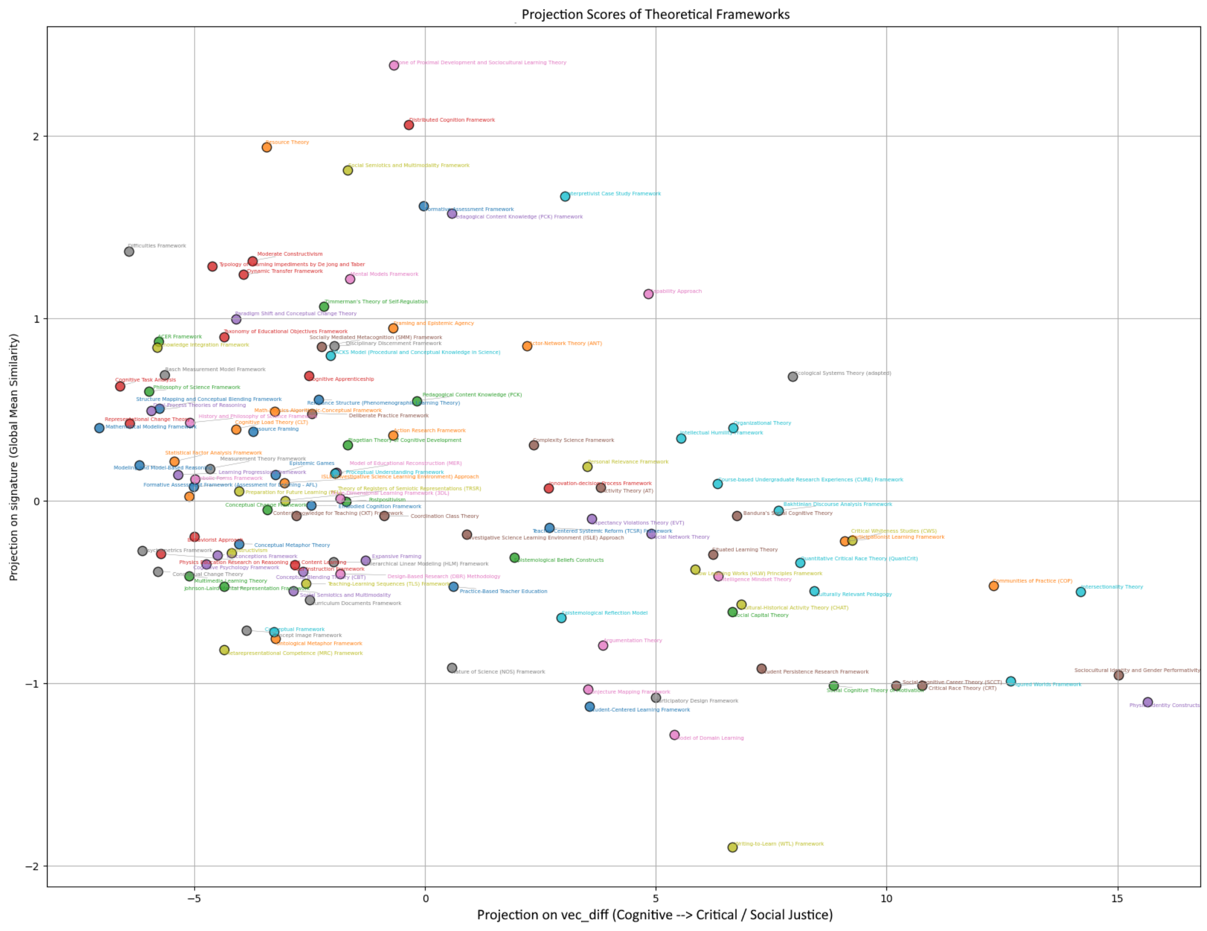


Figure 16: 2D projection scores of different theoretical frameworks, projection onto a ”Cognitive/Critical” theories difference vector (x-axis) and the global average (y-axis).

to systematically group data based on certain semantic traits.

Table 12: Categorization of Theoretical Frameworks

Class 1 (Cogntive/Mathematical)	**Class 2 (Critical/Social)**
Basch Measurement Model Framework	Physics Identity Constructs
Cognitive Task Analysis	Writing to Learn Framework
History and Philosophy of Science Education	Intersectionality Theory
Statistical Factor Analysis Framework	Sociocultural Identity and Gender Performativity
Constructivism	Social Cognitive Career Theory
Resource Framing	Student Persistence Research Framework
Cognitive Apprenticeships	Bandura's Social Cognitive Theory
Framing and Epistemic Agency	Social Capital Theory
Mathmatical Modeling Framework	Cultural-Historical Activity Theory
Multimedia Learning Theory	Quantitative Critical Race Theory

(see notebook '2D_projection_space.ipynb' for implementation and further details.)

7.2 Identifying Temporal Patterns

Focusing in on specific theoretical frameworks, we attempt to identify temporal patterns by taking the mean embedding of all theories in the first and last year of publishing, and projecting all samples on the difference of the means. Looking at Figure 17 and Figure 18, this approach does not seem to capture any clear linear development of the frameworks over time. While theories published in the same year seem to cluster together, the clusters jump sporadically around the x-axis from year to year rather than follow a clear incremental path from left to right over time (compared to Figure 11). This could perhaps indicate that theories are developed in an irregular, unpredictable fashion, rather than a continuous incremental evolution. This being said, a lot of information is lost while projecting the embeddings, and it is entirely possible that a better suited difference vector could reveal a clearer pattern or trajectory.

7.3 Exploring semantic correlations with article citation performance

Using SemanticScholars API to retrieve citation counts for all the theoretical frameworks, we also attempted to look for correlations between semantic properties and average ci-

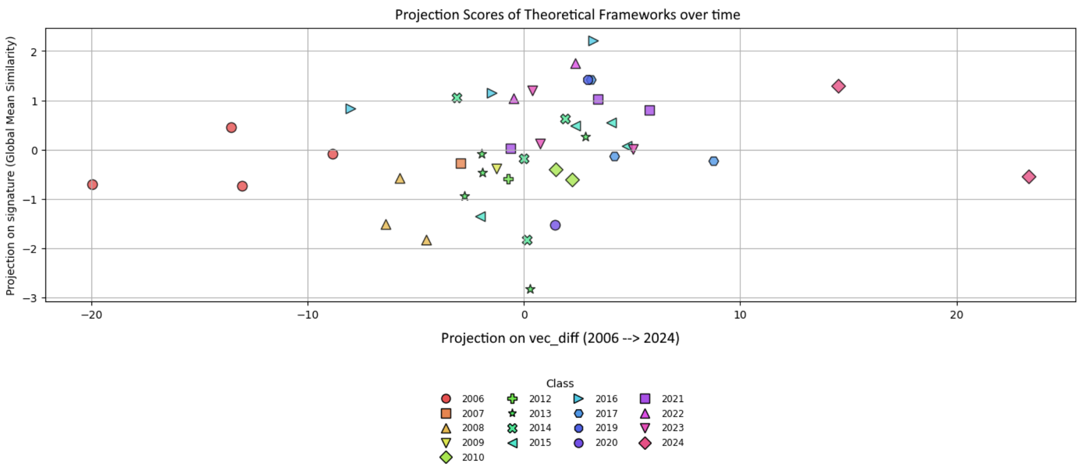


Figure 17: Projection scores of all theories classified as "Resource Framing", with 'vec_diff' being vector difference between mean of years 2024 and 2006.

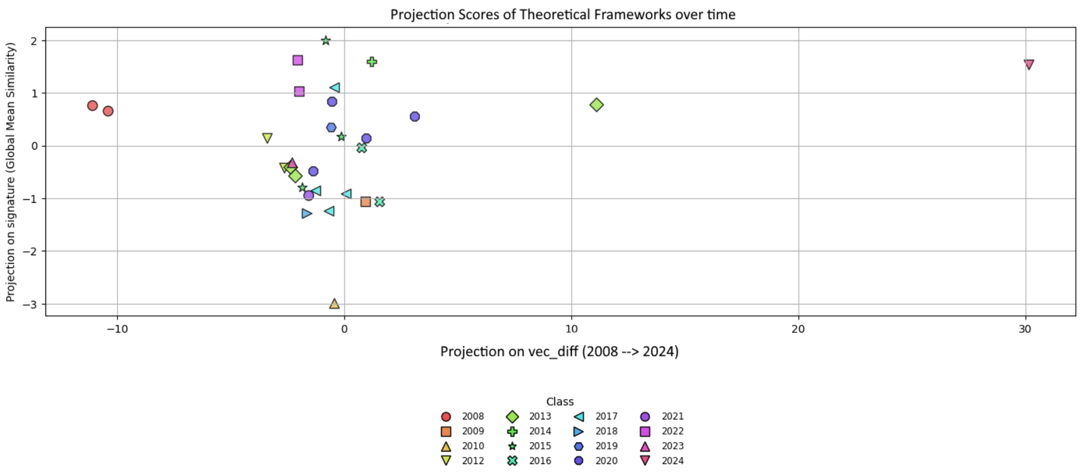


Figure 18: Projection scores of all theories classified as "Modeling and Model-based reasoning", with 'vec_diff' being vector difference between mean of years 2024 and 2008.

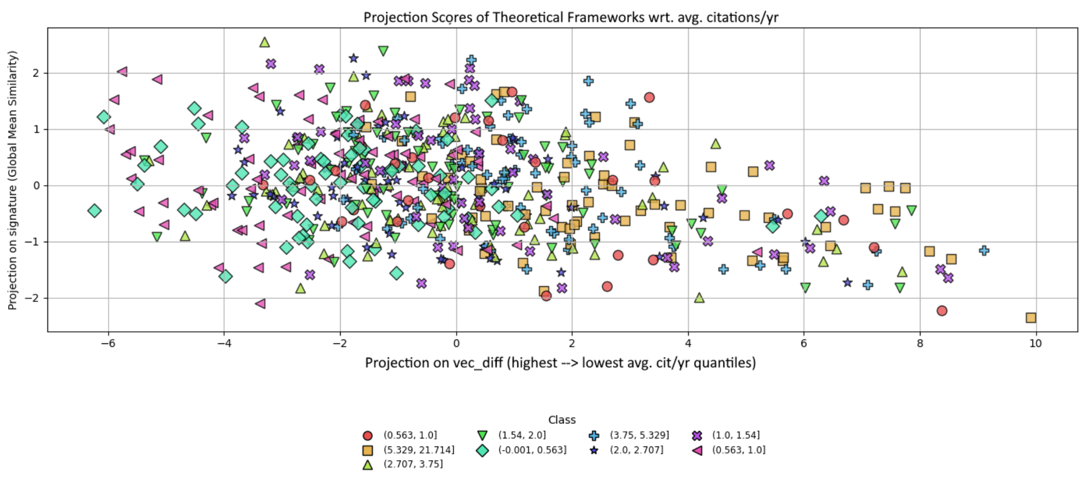


Figure 19: Projection scores of all theories onto difference vector between top and bottom 1/6 quantile of citations per year (x-axis) and global average embedding (y-axis). Points are labeled by their respective citations/year interval.

tations/yr. Dividing the theoretical frameworks into eight quantiles ranging from least to most average citations per year and projecting onto a difference vector based on the means of the highest and lowest quantiles (Figure 19), we were unable to find any patterns or general seperation.

We did however see a clear seperation between the lowest and highest citation quantiles (Figure 20). Training our MLP projection model on this binary problem using a 0.7/0.3 train-test-split, it performs with an accuracy of ∼0.76. This indicates that there exists some semantic characteristic differentiating high- and low-traction academic papers. Attempting to determine whether this semantic difference is related to the underlying nature of the theoretical frameworks, we again project the samples onto the aforementioned ”Cognitive/Critical” difference vector, labeling each point by their citation performance. Looking at Figure 21, there does not seem to be a particularly noticable crowding of either low- or high-citation papers on either the ”social/critical” or ”cognitive/mathematical” side, perhaps indicating that this differentiating semantic property is not dependent on theoretical orientation. Of course, this type of ”visual” analysis is very ”hand-wavey” and lacking in rigour, so these results should be taken with many a grain of salt.

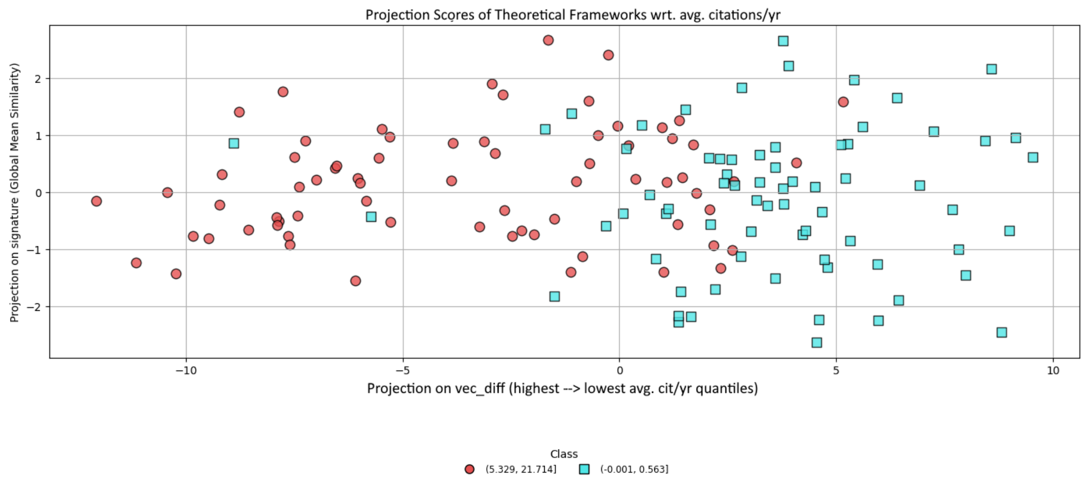


Figure 20: Projection scores of all theories onto difference vector between top and bottom 1/6 quantile of citations per year (x-axis) and global average embedding (y-axis). Points are labeled by their respective citations/year interval. Here we only include points belonging to the bottom and top quantile.

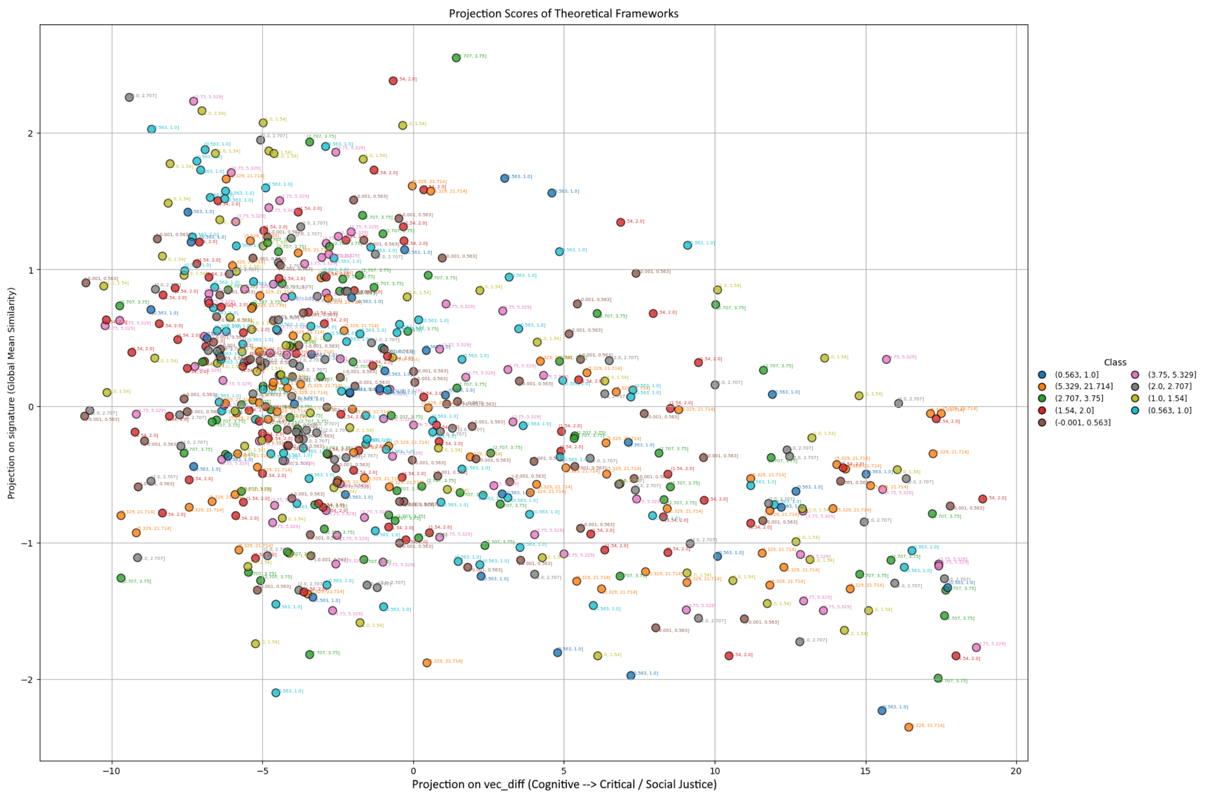


Figure 21: 2D projection scores of theoretical framework embeddings onto "Cogntive/Critical" difference vector (x-axis) and global average (y-axis). The points are labeled according to their citations/year quantile.

7.4 Discussion and Future Work

This type of projection-based analysis, primarily centered on vector differences between mean embeddings, is conceptually intuitive and easy to implement. The general idea of substracting mean centroids to produce customized interpretable axes seems to have some merit, yielding results that align with plausable groupings. The low computational cost and high data-efficiency allows for rapid iteration and testing of different grouping hypothesis. The method is also very generalizable, and swapping the constituent centroids of the difference vector immediately changed the result sample layout in understandable ways. This flexibility could make it a useful exploratory tool.

However, the success of this method hinges almost entirely on the usefulness of the chosen difference vector. Selecting representative anchors for each conceptual pole is also a subjective process and could easily bias the outcome. Furthermore, semantic centroids flatten all internal variation within a category, and projecting all embeddings down into a 2D space may be conceptually easy to interpret, but this comes at the cost of signficant information loss that is difficult to account for. And while the results may appear conceptually straightforward, this can be misleading; these geometric operations occur in highly abstract, high-dimensional spaces, and their outcomes are not always as semantically meaningful as they might initially seem. Lastly, unlike clustering or supervised classification where evaluation metrics can guide interpretation, projection maps rely heavily on visual inspection. This makes it difficult to rigorously assess what a pattern means, or whether it's meaningful at all.

While this analysis demontrates the potential of the projection-based semantic approach, it raises many questions and directions for future study. Further work is required to identify what the projection space captures in general. While the method is efficient and visually intuitive, interpretability of the axes remains a challenge. Although the citation difference vector successfully seperates highly and sparsely cited papers, its semantic signficant remains unclear; does it reflect alignment with theoretical paradigms, or is it more sensitive to shallow lexical and grammatical properties? Can any insights be derived from the relationship between x- and y-axes? Shedding light on these areas may also aid in the identification of temporal difference vectors that more clearly capture patterns over time. Investigating how different pre-trained models, embedding granularities, or normalization techniques affect projection behavior may also help interpret the semantic dimensions being encoded.

Developing or utilizing more rigorous inspection tools is also essential. Currently the analysis relies solely on visual inspection of plots, which is prone to inaccuracy and subjective bias. It would also be interesting to explore whether similar tools, in conjunction with a more sophisticated derivation of this method, could be used to systematically group data based on certain semantic traits. This would perhaps allow for other interesting prospects, such as the identification of emerging theoretical frameworks or under-cited but semantically novel work.

8 Conclusion

This report set out to address a key limitation in the computational analysis of scientific literature: the loss of nuance when treating entire articles as single data points. We developed and evaluated a complete workflow for performing a fine-grained, section-level analysis of the PRPER corpus, demonstrating the feasibility and value of this approach.

Our initial exploration confirmed that while a section-level approach is necessary, it is also challenging, as the thematic signal of an article often dominates the more subtle rhetorical signal of its sections. To overcome this, we compared several supervised classification models. We found that a Large Language Model using an external API achieved the highest accuracy at ∼90%, while a novel, projection-based MLP classifier offered a strong and computationally efficient local alternative (∼80%), outperforming a fine-tuned SciBERT model (∼77%).

Using the most accurate classifications, we pioneered a novel, iterative LLM-based method capable of discovering and categorizing specific theoretical frameworks and research methods directly from the text. This technique successfully mapped the conceptual evolution of the PRPER journal over two decades, showcasing its potential as a powerful tool for automated, data-driven literature reviews.

This work is not without limitations. The classification task is complicated by the inherent ambiguity of section boundaries, and the iterative analysis method requires significant computational resources and human-in-the-loop validation. Future work should focus on applying this workflow to other research corpora and refining the iterative classifier with more powerful models to better distinguish between the ”use” and ”discussion” of a concept. Further investigation into the semantic axes discovered through projection-based analysis could also yield more rigorous tools for interpreting the conceptual landscape of a scientific field.

Bibliography

Beltagy, I., Lo, K., & Cohan, A. (2019). Scibert: A pretrained language model for scientific text. *Conference on Empirical Methods in Natural Language Processing*. https://api.semanticscholar.org/CorpusID:202558505

Coenen, A., & Pearce, A. (n.d.). Understanding UMAP [Accessed: 2025-08-09]. https://pair-code.github.io/understanding-umap/

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/https://doi.org/10.1016/0893-6080(89)90020-8

IBM. (n.d.). Principal component analysis [Accessed: 2025-08-09]. https://www.ibm.com/think/topics/principal-component-analysis

Mazraeh, A. (2025, February). A comprehensive guide to dimensionality reduction: From basic to super-advanced techniques 1. https://medium.com/@adnan.mazraeh1993/a-comprehensive-guide-to-dimensionality-reduction-from-basic-to-super-advanced-techniques-1-d17ce8e734d8

Odden, T. O. B., Tyseng, H., Mjaaland, J. T., Kreutzer, M. F., & Malthe-Sørenssen, A. (2024). Using text embeddings for deductive qualitative research at scale in physics education. *Physical Review Physics Education Research*, *20*(2). https://doi.org/10.1103/PhysRevPhysEducRes.20.020151

Radev, D. R., Jing, H., Sty, M., & Tam, D. (2004). Centroid-based summarization of multiple documents. *Information Processing & Management*, *40*(6), 919–938. https://doi.org/10.1016/j.ipm.2003.10.006

Stewart, G., & Al-Khassaweneh, M. (2022). An implementation of the HDBSCAN* clustering algorithm. *Applied Sciences*, *12*(5). https://doi.org/10.3390/app12052405

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention is all you need. https://arxiv.org/abs/1706.03762

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., … Rush, A. (2020, October). Transformers: State-of-the-art natural language processing. In Q. Liu & D. Schlangen (Eds.), *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-demos.6