* Introduction (Lisa og Helene)
* Background
  + Hva er gjort (Helene og Lisa)
  + Hva kan gjøres: splitte artikler (Helene/Lisa og Severin)
  + Teori/beskrivelse av metoder
    - Helene og Lisa fyller inn hva de har
    - Severin og Lars skriver om de metodene teoriene de har benyttet
* Splitting og utforskning
  + Metode
    - Severin skriver om algorithmic for splitting og embedding. (Kort)
      * Legge inn feilen med embeddings
    - Helene og Lisa kan fylle inn om plotting og clustering
  + Resultat
* Klassifisering
* Heuristic
  + - Metode
    - resultater
      * Severin sier noe kort her
* GPT
  + - Metode
    - resultater
      * Lisa og Helene snakker om optimalisering av GPT
      * Severin skriver om de to metodene
* BERT (Lars)
  + Metode
  + Resultater
* Projection scoring (Lars)
  + Metode
  + Resultater
* Analyse
  + Metoder
  + Resultater
  + Analyse vha projection scoring
* Felles Diskusjon
* Konklusjon/Further work/Ouverture

Article splitting (eller en kulere tittel)

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

# Contents

[Contents 2](#_Toc1488046033)

[Abstract 2](#_Toc1604031490)

[Introduction 2](#_Toc1518352041)

[Background 3](#_Toc297346079)

[Previous work 3](#_Toc195627481)

[Goals 3](#_Toc464253810)

[Dimensionality reduction 4](#_Toc1553754109)

[Clustering 5](#_Toc258492392)

[Article splitting, embedding and exploration 5](#_Toc1648710502)

[Initial exploration of the embedded data 6](#_Toc718174363)

[Classification 8](#_Toc497417416)

[Benchmarking 9](#_Toc1732681176)

[Classification with heuristics 9](#_Toc87162329)

[Classification with ChatGPT 10](#_Toc607267401)

[Classification with BERT 11](#_Toc1703945575)

[Analysis 11](#_Toc1601537352)

[Inter-article vector arithmetic 12](#_Toc2005170698)

[Theory identification with ChatGPT 12](#_Toc185616924)

[Discussion 14](#_Toc4759095)

[Conclusion 15](#_Toc1445649928)

[References 15](#_Toc552614247)

# Abstract

# Introduction

Text embeddings are an NLP method that contains the ability to transform textual data into numerical vectors. These vectors exist in high dimensional spaces, where their position in the embedding space carries semantic meaning understood by machine learning models. Their relative distance can reveal how related different words are to each other and can help highlight patterns. Semantically similar words usually exist close to one another. Representing text in such an environment by numerical vectors allows researchers to mathematically manipulate the data using classic vector-handling techniques, as well as study patterns in the data over time (Odden et al., 2024).

As such, this project’s aim is to explore a dataset consisting of 1222 articles gathered from the Physical Review Physics Education Research journal, (henceforth PRPER). The exploration involves splitting the articles and looking for patterns or interesting variations. This includes exploring different machine learning methods to attempt to classify sections of the articles, using classic vector-handling techniques (subtraction and addition) to estimate the position of sections in the embedding space as well as (...)

# Background

## 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.

## Goals

A limitation of Lane’s work is that it is based on the article-level: embeddings are generated by passing in the full text of each article. Thus, each article is reduced down to a single point in the embedding space. Scientific articles tend to have their meaning quite strictly separated into sections such as “Methods”, “Theory”, “Methods”, etc. Thus, a more fine-grained splitting of the articles would open the possibility of not only investigating the general theme of each article but also looking at more fine-grained aspects of each article. For instance, one could extract the theoretical frameworks and methods used or shifts in how results are discussed.

A strength of the PRPER dataset is that it includes the full documents as XML. Thus, by parsing the XML tree, we can easily and accurately extract each section for further analysis. For such an analysis to be fruitful, we would also need to classify each section according to its role in the article. Pursuing this line of inquiry, our project has gained three main parts: (1) splitting, embedding and initial exploration; (2) section classification; and finally, (3) analysis.

The main goal of our own project is to further explore and develop the methods mentioned above in order to analyze scientific literature. We will be using the same dataset as Lane: 1222 articles from PRPER. We will split the articles into sections such as “Introduction”, “Literature Review”, “Theoretical Framework”, “Methods”, “Results”, “Discussion” and “Conclusion”, and then embed them into vectors in a meaning space. We will then investigate how well different models are able to classify text from the articles according to the mentioned categories, og analysere et eller annet.

This project revolves around the key concepts of dimensionality reduction and clustering, which warrant further explanation.

In machine learning (henceforth, ML) embeddings exist in high-dimensional spaces. These high-dimensional spaces can be computationally expensive to run and difficult to interpret. To address this issue, dimensional reduction techniques can decrease the number of features in the space, while keeping the most essential parts of the structure and meaning of the data intact (Mazraeh, 2025). In this particular project we used t-SNE, UMAP and PCA. All these methods share the common goal of simplifying high-dimensional data, but they do this in different ways. For this reason, we applied multiple of the techniques on the different codes throughout the project. The use of these techniques enabled us to project the embedding space in as little as 2 or 3 dimensions, making it possible to visualize the data.

## Dimensionality reduction

Principal Component Analysis, or PCA, is a linear technique of dimensionality reduction meant to preserve as much variance as possible. By summarizing the information of the input, PCA creates a reduced dataset with new and fewer variables (principal components). In effect, this approach aims to preserve the most important features of the data, and thus the global structure of the dataset. When used in 2D-visualization, the two principal components that capture the most variance are used as the axes (IBM, n.d.).

T-distributed Stochastic Neighbour Embedding, or t-SNE, is a non-linear approach. It preserves neighbouring data points when transferred into lower dimensional spaces, and in contrast to PCA, therefore preserves the local structure of the data. The technique is therefore suitable for clustering and for visualizing data in lower dimensions (Mazraeh, 2025).

Lastly, Uniform Manifold Approximation and Projection, UMAP, is another non-linear approach that preserves both local and global structure to some extent. It functions much like t-SNE, but preserves more of the global structure and is faster to run. However, PCA is the computationally cheapest option, as well as the most intuitive to interpret, “because UMAP and t-SNE both necessarily warp the high-dimensional shape of the data when projecting to lower dimensions” (Coenen & Pearce, n.d.).

## Clustering

Clustering involves grouping similar data points, vectors in our case, around their position in the embedding space. This is done through an unsupervised learning technique, meaning that the algorithm is given unlabelled data and finds groups and patterns on its own. We used both K-means and HDBSCAN in our project. K-means partitions the data into clusters centred around a calculated centroid and assigns points to their nearest centre. The centroid being the average position for the shape in question. This is not the case for HDBSCAN, which creates clusters based on data density. More specifically how tightly the points in a scatter plot are placed. This makes HDBSCAN particularly effective for handling noisy datasets (Stewart & Al-Khassaweneh, 2022).

# Article splitting, embedding and exploration

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-3-large" 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 Jira’s embedding models.

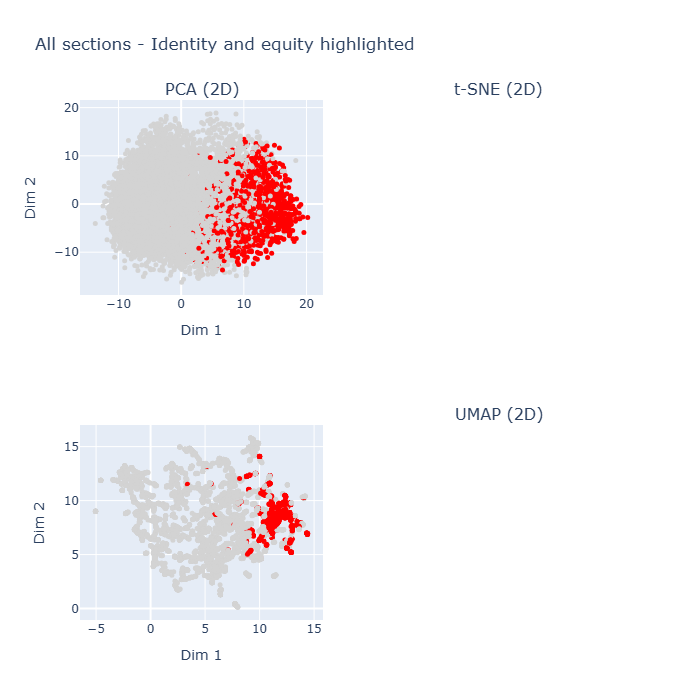
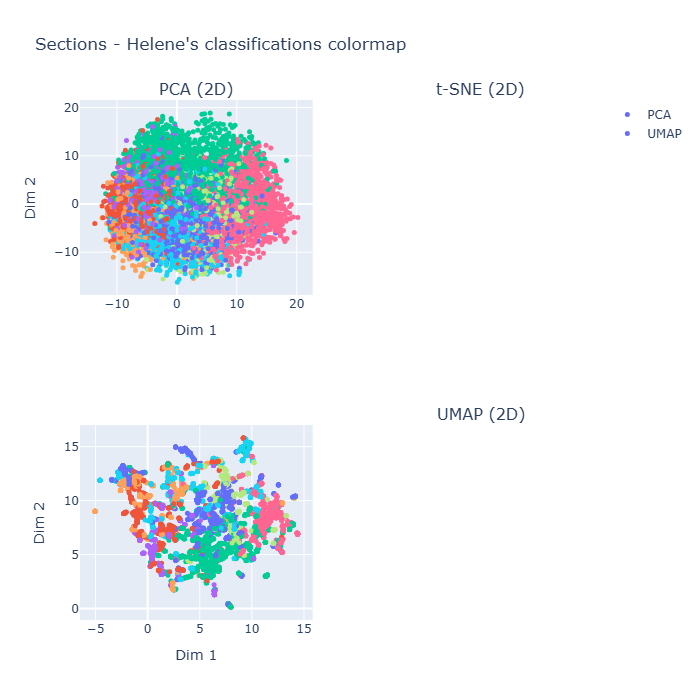
We chose to implement the extracted text in two ways. First, we embedded each whole section into one vector. It was this embedding that we used for our initial analysis. Second, we chucked the text into chunks of up to 300 characters, splitting the sentence level.

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

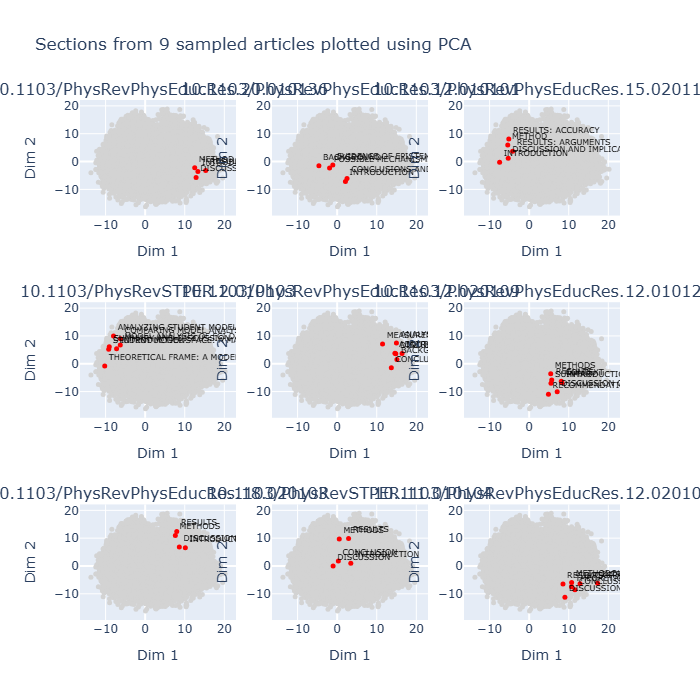
## Initial exploration of the embedded data

To get a general impression of the embeddings space and see potential avenues for classification we did some initial clustering, dimensional reduction, and plotting based on the per section embeddings. Our main focus here was on finding properties that could be exploited for classifying the sections into their respective function in the article – e.g. “Methods”, “Theory”, “Discussion” etc. A focus that our plots reflect. However, we also looked at the distribution of sections per article, and the distribution of sections based on thematical categorization from Helene Lane’s analysis. Let us begin with these latter two.

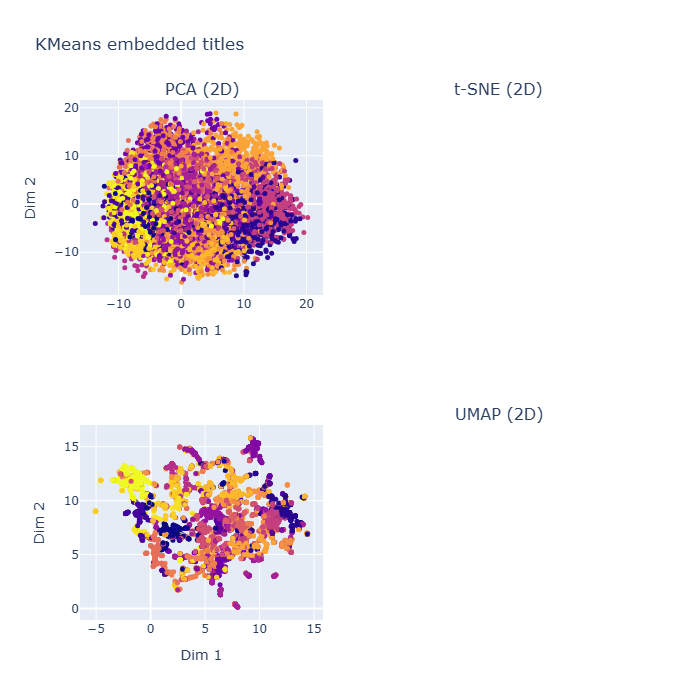
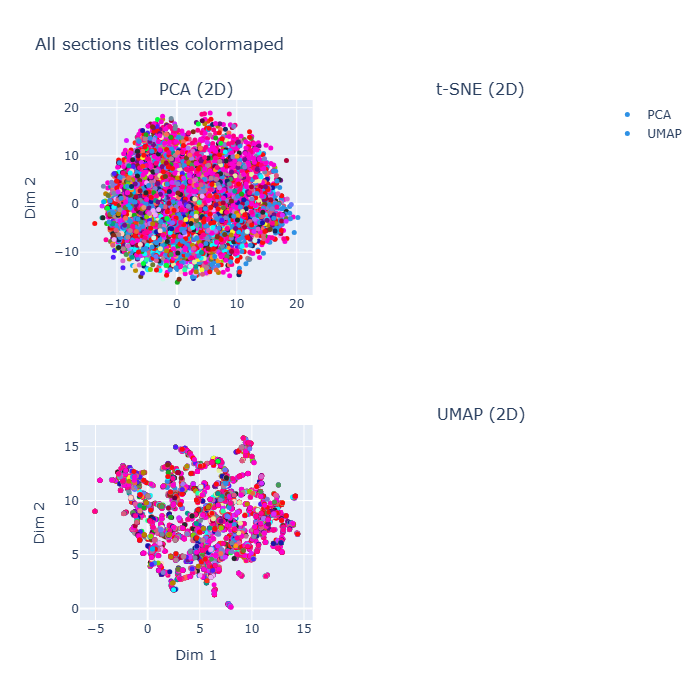
In [fig 1 and 2], we have plotted sections using PCA and UMAP reductions to 2 dimensions. In [fig 2] we have highlighted all sections that belong to an article that was classed as dealing with the theme of “identity and equity” by Helene Lane’s method. In [fig 1], we have plotted all sections and colored them based on the theme classification from Helene Lane’s project. As is evident from the clustering in both plots, each section's location in the embedding space seems to be dominated by the overall theme of the article. As will become evident later, this will make the classification of theoretical framework and methods used problematic.



We are also interested in the internal structure of each article. To get an impression of the distance between sections within an article, we plotted all sections using a PCA reduction, then highlighted the sections belonging to 9 randomly sampled articles. The results can be found in [fig 3]. Judging from the figure, there appears to be a great deal of clustering on the article level, but with meaningful variation on the intra-section level. This is not unsurprising given the importance of article theme for section position in the embedding space, but it also gives hope that there could be some internal structure to each article that can be exploited. We look at the possibility of exploiting this later on in this report.



To explore the overall distribution of different section types in the embedding space, we attempted to run clustering algorithms on the per section embeddings. For the sake of exploration, we tried both volume and density-based clustering algorithms (K-Means and HDBSCAN). Since a large proportion of the section types were inferable by titles such as “Methods”, “Methodology”, “Introduction”, etc. we also compiled some basic categories based on section headings and color coded them. See [fig ???]. We could see some patterns in the plots based on clusters. However, they were not based on section type, nor could we make them intelligible in any other way. Similarly, the color-coded heading-based plot did not yield any recognizable distribution based on the inferred section type.



**(Should we add the title embedding plots here too?)**

# Classification

Before we could commence any thorough analysis of the data, we saw a need to first classify each section by the function it played for its article; to do any analysis of theoretical developments, methodological shifts and so forth, we would first need to identify the theory, methods, etc. sections of each article.

As mentioned, many sections could be easily identifiable by headings such as “Methods”, “Methodology”, “Introduction”, and “Discussion”. Yet, many sections had headings too specific to be so easily classified. We therefore came up with a series of different approaches to classifying sections by type. These include: a weighted heuristical approach, a fine-tuned BERT model, a geometrical approach, and an LLM (ChatGPT) based approach. Before diving into these, let us briefly review our data and method for benchmarking these differing approaches.

## Benchmarking

To make meaningful judgments about the accuracy of each approach, we generated two labeled benchmarks for estimating accuracy. The main dataset was generated by sampling 22 articles with a total of approx. 170 sections and hand labeling them by their type. By randomly selecting and hand labeling these sections we got a representative and accurate set of sections that covered a broad set of section types. The drawback of this method is its resource intensive nature, limiting us to approx. 170 datapoints. To make up for this drawback, we also generated a secondary dataset for benchmarking. This was generated by taking a selection of sections with obvious section headings such as “Methods” and “Theory”. To avoid an uneven distribution of types, we filtered the dataset down to contain an equal number of each – yielding approx. 500 sections. This gives us a more robust size to run our benchmarks but has the drawbacks of not covering all possible types and probably containing fairly *un*ambiguous sections that are easy to classify.

To get a comprehensive way of diagnosing and tuning our classification methods, we created an evaluation function that gave us an overall accuracy estimate, and per category accuracy estimates, mis-categorization counts, and a confusion matrix. To allow for iteration of categories, we also included the option to map the hand labeled categories to new categories.

## Classification with heuristics

Building on the easily inferable type of many sections based on their heading, we attempted to find other similar recognizable features that could help our inference. We could expect each type to typically contain certain keywords in their heading and text body, for some sections to be longer than others – e.g. introductions and conclusions being shorter than discussion sections – and the relative position of a section in the article depending on the type of section – e.g. introductions first, conclusions last, and discussions somewhere in the middle. Thus, we ended up with the following heuristical features: title and content keywords, section length, and relative position. We could then combine these into a heuristical function that weighs the predictions based on these patterns to give a final estimate of the section type.

After tuning the weights, we found the optimal distribution was 0.8 for title keyword, 0.1 for length, and 0.1 for relative position. In other words, the best predictor of section type was title keyword matching. See computational essay for full implementation.

## Classification with ChatGPT

Perhaps a naïve approach to classification, we attempted to simply pass sections to ChatGPT using their API and asking it to classify them for us. To find the best approach we experimented with different prompts, output formats, tuning the categories, and ways of passing the sections and their context clues to the LLM.

We tried out two ways of passing the sections to the LLM. First, we attempted to pass one section at a time, and adding context clues such as section heading and its relative position in the article. Second, we passed the whole article at once with the sections clearly separated, then asking the LLM to classify all the sections at once. This latter approach proved superior in accuracy (and had the added benefit of speed as it made approx. 1/7th the number of requests to the OpenAI API). Therefore, we worked on further improving this approach.

We did not find much improvement with fiddling with the main prompt. Either we got lucky with our initial formulations, or the LLM is flexible in how it receives its instructions. Where we found significant improvements in adjusting the return format from the LLM. We used the json return option in the OpenAI API to ensure valid programmatically readable returns, then requesting it to return both a probability distribution of its classification and its final classification as an array. We requested the final classification as an array so that the LLM could classify ambiguous sections – e.g. sections titled “Discussion and Results” – as multiple types. However, we did not use this “multiple type” classification result in our benchmarking, rather sticking with the highest probability returned for the sake of simplicity.

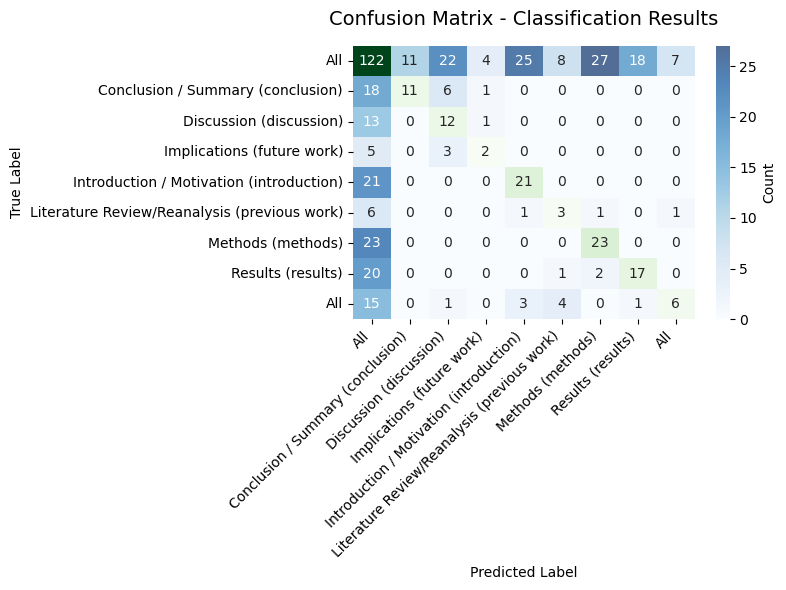
Finally, we used the evaluation output described above to tune the categories and their descriptions. These were added to the prompt as a list of named categories with descriptions for added context.

With the final version of this classifier, we got an accuracy of *0.8* using the model “gpt-4.1-mini” and *0.9* with “gpt-4.1” when run on the hand labeled benchmark dataset. On the heading-based benchmark set we got an accuracy of *1.0* using the “gpt-4.1xxx”. Since we suspect some mislabeling in the hand-labeled dataset, we appear to get near perfect accuracy using this method. We can also see some clear patterns in the misclassifications that this method makes.

Overall, the category the model has the most difficulty classifying is “Theoretical Framework”. Other categories that it often struggles with are "Implications”, “Conclusion/Summary” and “Literature Review/Reanalysis”. In fact, the most common mistakes for the model are to misclassify “Theoretical Framework” as “Literature Review/Reanalysis”, and to confuse “Implications” and “Conclusion/Summary” with each other as well as with the “Discussion” category. To some extent, these mistakes make sense. In effect, the content of these sections might often overlap or be similar to each other in research articles.

In effect, the most common mistakes for the model are ones concerning sections that even humans would struggle to classify. When hand-coding and manually sorting the articles, multiple sections were found to be ambiguous. Sections named “Results and Discussion” for instance, would conceptually belong to both the “Results” and “Discussion” categories. However, we had to pick only one category each time. The manually sorted categories could therefore spark some subjective disagreement as decisions had to be made to reduce complexity. We encourage future research to critically examine our manually sorted categories in order to ensure validity.

See [fig ???] for a full evaluation printout for this method.



## Classification with BERT

# Analysis

## Inter-article vector arithmetic

A well-known example to demonstrate how embedding encodes information in high-dimensional space is the use of basic vector arithmetic on related terms. Examples are the dyads man–woman and ki ng–queen. By subtracting *man* from *woman*, then adding the resulting difference to *king*, we will get a vector approximating the embedding vector of *queen*.

Transferring this logic to our section embeddings, we wondered if it would be possible to perform similar arithmetic to estimate the movement from e.g. a *theory* section to a *methods* section.

Based on our classifications of sections obtained from the ChatGPT method above, we started by sampling all articles that had a *theory* *section* directly followed by a *methods* *section*, as well as a *discussion* *section* at some point later in the article. We could then try to subtract the *theory section* from the *methods section* in an article, then use this difference to estimate the *methods section* location in another article by adding this difference to that article's *theory section.* We could then calculate the accuracy of our estimate by means of a cosine similarity measure. To control our result, we could also generate a cosine similarity between the estimate and the *discussion section* of the same article. To avoid an averaging of results hiding interesting results, we did this comparison pairwise for all possible combinations in our filtered dataset. [fig ??? And ??] shows our results as histograms.

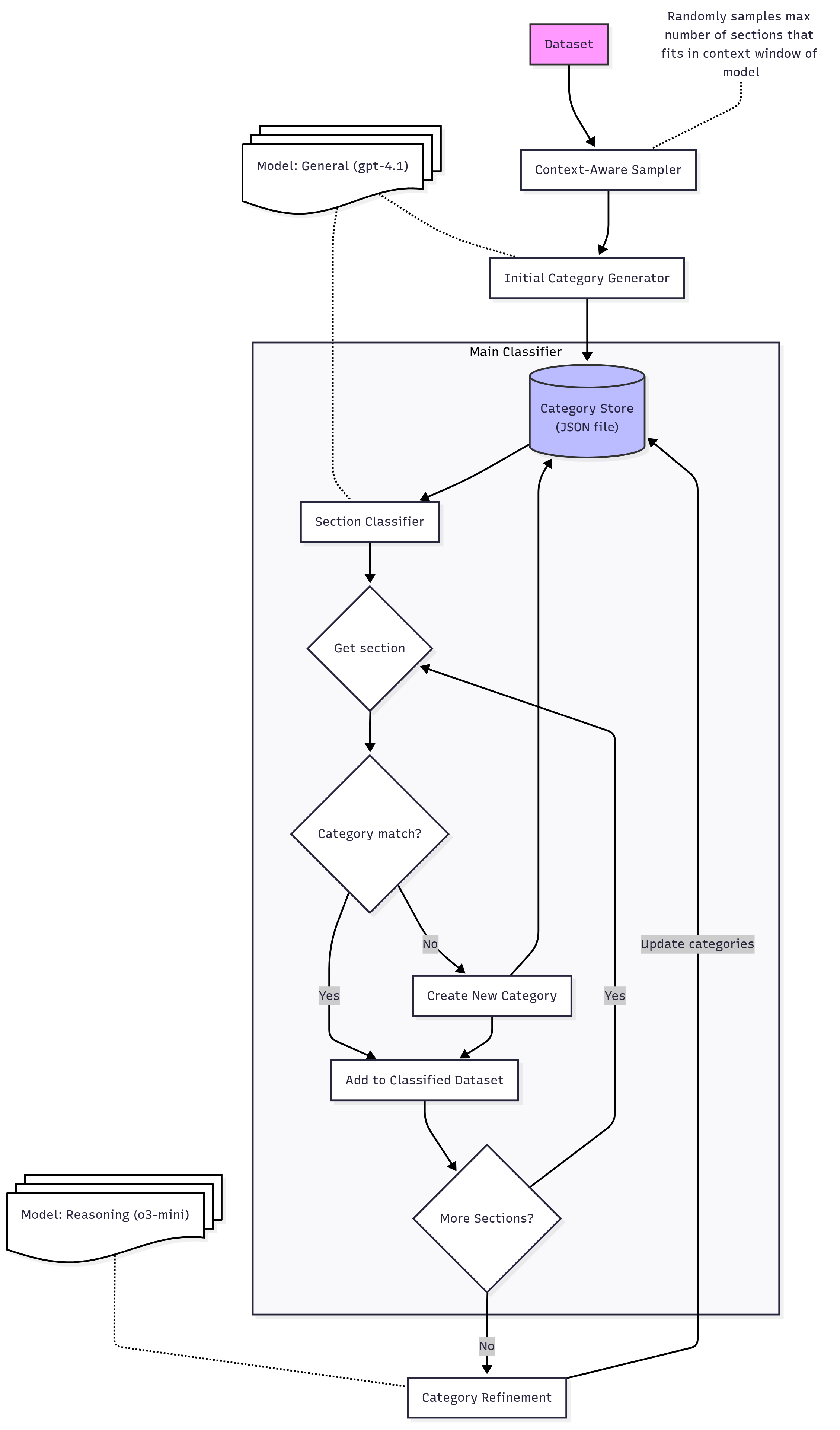
If such a naïve arithmetic approach worked, we should see a difference between our estimate scores and our control scores. Except for a small difference in variance, we can see no difference between the two plots. Given the dominance of article thematics in positioning the sections in our embedding space, this does not come as a surprise.

We tried to filter away this thematic influence from our comparison. For this filtering, we based ourselves on the chunked embeddings. For each article we generated a centroid by averaging all of its sections. We then filtered out the chunks in each section that exceeded a cosine similarity threshold. Our hope was that this would remove sentences that dealt more with the overall theme and leave behind sentences which dealt more with the isolated method and theory. After some explorative runs, we landed on 0.5 similarity as a threshold that ensured dissimilarity while retaining enough sample sections to run our comparison.

## Theory identification with ChatGPT

Using our classified section dataset, we are well poised to investigate theoretical developments within PRPER: by filtering for only *theory sections,* we will have a focused dataset dealing exclusively with theoretical aspects of the discourse. We have attempted to identify which theoretical frameworks are used in each article to see the theoretical development over time.

To extract the theoretical frameworks used, we have developed a staged approach that iteratively feeds data into LLM models to generate initial categories, extends this initial list and categorizes sections, then reviews the category list and repeats. By using suitable general purpose and reasoning models depending on the task and updating its own prompt, we appear to have been able to generate a comprehensive and fine-grained classification scheme for theory sections in our dataset. As the saying goes, a picture says a thousand words, so rather than try to trace the logic of the method here, we rather present this flow chart. For the full implementation, see the computational notebook and connected codebase.



This logic is implemented in an *AbstractClass* in python, which can be instantiated to implement it for different purposes. Included in the code is an implementation of the theory classificator, but with simple modifications to the prompts and data selection, it can also be used in a similar manner to identify methodologies.

Since this approach generates a fine-grained list of theories (approx. 100 theories for 500 samples), we have also looked at implementing a similar logic for aggregating these theories into larger categories. See the dev branch in the git repo for a work-in-progress implementation of this logic.

# Discussion

|  |  |
| --- | --- |
| **Classification method** | **Accuracy results** |
| Heuristics |  |
| LLM (ChatGPT) | Introduction: 1.0  Theoretical Framework: 0.357  Methods: 0.958  Results: 0.947  Discussion: 0.923  Conclusion: 0.722 |
| BERT |  |
| ? |  |

# Conclusion

# References

Caballar, R., & Stryker, C. (2024, December 13). *LLM APIs: Tips for bridging the gap*. IBM. <https://www.ibm.com/think/insights/llm-apis>

Mazraeh, A. (2025, February 23). *A comprehensive guide to dimensionality reduction: From basic to super-advanced techniques 1*. Medium. <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*, Article 020151. <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://www.sciencedirect.com/science/article/pii/S0306457303000955>

Stewart, G., & Al-Khassaweneh, M. (2022). An implementation of the HDBSCAN\* clustering algorithm. *Applied Sciences, 12*(5), 2405. <https://doi.org/10.3390/app12052405>

IBM. (n.d.). *Principal component analysis*. IBM. <https://www.ibm.com/think/topics/principal-component-analysis>

Coenen, A., & Pearce, A. (n.d.). *Understanding UMAP*. People + AI Research (PAIR). <https://pair-code.github.io/understanding-umap/>

‌

‌