תמונה שמכילה טקסט, גופן, לוגו, גרפיקה

התיאור נוצר באופן אוטומטי

Software Engineering Department  
ORT Braude College

Capstone Project Phase B – 61999

**Metalanguage As an Interdisciplinary Classifier for Mathematics and Computer Science Fields**  
**24-2-R-10**

Ilya Vorotyntsev | Ilya.Vorotyntsev@e.braude.ac.il

Aleksandr Pitkin | Aleksandr.Pitkin@e.braude.ac.il

**Supervisors:**

Dr. Dan Lemberg | [lembergdan@braude.ac.il](mailto:lembergdan@braude.ac.il)

Mrs. Elena Kramer | [elenak@braude.ac.il](mailto:elenak@braude.ac.il)

**https://github.com/iliavrtn/final-project**

Table Of Contents

[Abstract 2](#_Toc189499249)

[1. Introduction 3](#_Toc189499250)

[2. Literature Review 4](#_Toc189499251)

[2.1 Traditional Approaches to Text Classification 5](#_Toc189499252)

[2.2 Evolution to Deep Learning Techniques 5](#_Toc189499253)

[2.3 Applications in Educational Text Classification 6](#_Toc189499254)

[2.4 Traditional Approaches to Text Clustering 6](#_Toc189499255)

[3. Background 7](#_Toc189499256)

[3.1 XLNet 7](#_Toc189499257)

[3.2 K-Means Clustering 11](#_Toc189499258)

[3.3 PAM (Partitioning Around Medoids) 11](#_Toc189499259)

[3.4 Density-Based Clustering 12](#_Toc189499260)

[3.5 Gaussian Mixture Models (GMMs) 12](#_Toc189499261)

[4. System Implementation 13](#_Toc189499262)

[4.1 Dataset Collection 13](#_Toc189499263)

[4.2 Text Preprocessing 14](#_Toc189499264)

[4.3 Text Classification 15](#_Toc189499265)

[4.4 Text Clustering 16](#_Toc189499266)

[5. Achieved Results 17](#_Toc189499267)

[5.1 Classification Performance 17](#_Toc189499268)

[5.2 Clustering Outcomes 20](#_Toc189499269)

[5.3 Overall Conclusion 24](#_Toc189499270)

[6. Challenges 25](#_Toc189499271)

[6.1 Preprocessing Difficulties 25](#_Toc189499272)

[7. Future Work 25](#_Toc189499273)

[8. Bibliography 27](#_Toc189499274)

# Abstract

Providing students with the most effective learning resources remains essential for improving educational outcomes, particularly in fields like mathematics and computer science, which involve complex and varied concepts. To better understand how these subjects are taught, we explored the linguistic structures that underlie academic texts across selected fields of mathematics and computer science.

Our methodology employed advanced natural language processing (NLP) techniques—lemmatization, and tokenization—to remove domain-specific terminology from the texts. What remains is the metalanguage—a more general form of language that consists of common words and structures used to communicate ideas across different fields. We then analyzed these preprocessed texts using XLNet, a deep learning model, to classify them according to their respective domains.

Alongside classification, we applied multiple clustering algorithms (K-means, PAM, Density Clustering, and Gaussian Mixture) to verify the classification results. This dual approach allowed us to examine correlations between classification accuracy and the natural grouping of texts, thereby offering insights into whether texts from related fields share similar linguistic patterns.

By successfully classifying and clustering educational texts without relying on subject-specific terminology, this research contributes to the development of more adaptable learning tools. Such tools could be better aligned with the diverse needs of students in mathematics and computer science by focusing on deeper linguistic structures rather than traditional keyword-based approaches.

**Keywords:**

Text Classification, Natural Language Processing, Clustering, Mathematics and Computer Science Education.

# Introduction

Students often face different challenges in various academic fields. Although mathematics and computer science share logical foundations, they require distinct skill sets, which may cause some students to excel in one domain while struggling in the other. This discrepancy can result in lower performance or even students dropping out of courses. Because a single, uniform teaching approach does not always address these differences, there is a growing need for learning strategies that accommodate individual student needs more effectively.

This project was motivated by an observation: students frequently encounter difficulties in specific subjects within mathematics or computer science, such as Linear Algebra, Calculus, Object-Oriented Programming, or Data Structures. We wanted to investigate whether these areas could still be distinguished by their general linguistic style once we stripped away the domain-specific terminology.

To explore this idea, we removed domain-specific words from a collection of educational PDF texts, then lemmatized and tokenized the resulting corpus. Our primary objective was to see if any “metalanguage” remains—i.e., if these texts can still be correctly classified or clustered based on more universal language features alone. For classification, we used XLNet, a transformer-based model that processes the preprocessed text chunks directly. Meanwhile, for clustering, we employed Doc2Vec [1] to generate numerical embeddings of each text segment, followed by several algorithms (K-Means, PAM, Density-Based Clustering, and Gaussian Mixture Models) to see whether texts group naturally by field.

By running both supervised (XLNet) and unsupervised (clustering) analyses, we aimed to determine whether the removal of domain-specific terminology obliterates all distinguishing markers between fields—or whether enough linguistic cues remain to enable meaningful categorization. If a consistent “metalanguage” does exist, it might offer valuable insights for designing more flexible educational tools or developing teaching methods that focus on writing style, structure, and common vocabulary instead of domain-specific jargon.

In the following sections, we discuss existing approaches to educational text classification, detail our preprocessing pipeline, describe how we applied both classification and clustering, and present our findings on the feasibility of classifying texts once their specialized terms have been removed.

# Literature Review

This section reviews the evolution of text classification methodologies, emphasizing the transition from traditional approaches to modern deep learning techniques. In addition to text classification, text clustering methods are essential for grouping similar texts without prior labels. Techniques such as K-means, Partitioning Around Medoids (PAM), Density-based clustering, and Gaussian Mixture models are widely used in natural language processing to uncover hidden patterns within large datasets [2].

## Traditional Approaches to Text Classification

Historically, text classification relied heavily on machine learning algorithms such as Naive Bayes, Support Vector Machines (SVM) [3], and decision trees. These models required extensive feature engineering, where text features such as word frequencies, TF-IDF scores, and n-grams were manually extracted and fed into the classifiers [4]​. Despite their effectiveness, these traditional approaches often struggled with the complexities of language, especially in handling nuances like context, polysemy, and semantic relationships.

**Naive Bayes** classifiers, one of the earliest models used for text classification [3], operate on the assumption that features are independent given the class label. While this assumption is rarely true in practice, Naive Bayes models perform surprisingly well, especially in scenarios where the independence assumption approximately holds, such as spam detection [5]​.

**Support Vector Machines (SVMs)**, another popular traditional method, utilize a hyperplane to separate classes in a high-dimensional feature space. SVMs are particularly effective for binary classification tasks and are known for their robustness in handling high-dimensional data. However, they require careful tuning of hyperparameters and kernel functions to perform optimally [4].

Despite their success, these traditional approaches have limitations, particularly in their ability to capture the deeper semantics of the text. They are heavily reliant on the quality of the input features and often require large amounts of labeled data to perform well. Additionally, they may not generalize effectively to unseen data or complex language patterns.

## Evolution to Deep Learning Techniques

The advent of deep learning has revolutionized text classification, enabling models to automatically learn features from raw text data. **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)** are among the earliest deep learning architectures applied to text classification tasks.

**CNN**s were initially developed for image processing but have proven effective in NLP tasks due to their ability to capture local dependencies in text. By applying convolutional filters over word embeddings, CNNs can identify significant phrases or n-grams, which are crucial for classification [5] [4].

**RNN**s, particularly Long Short-Term Memory (LSTM) networks, are designed to capture sequential dependencies in text. LSTMs address the vanishing gradient problem in traditional RNNs, allowing them to learn long-term dependencies, making them well-suited for tasks where the order of words is important [3] [5]​. However, RNNs can be computationally expensive and are often slower to train compared to CNNs [3].

The introduction of **Attention Mechanisms** and **Transformers** has further advanced the field. Transformers, such as the XLNet [6]. model, have become the state-of-the-art in text classification. Unlike RNNs, Transformers do not process text sequentially; instead, they rely on self-attention mechanisms to weigh the importance of each word in a sentence relative to all other words. This allows them to capture contextual information more effectively, leading to significant improvements in classification accuracy [4].

## Applications in Educational Text Classification

In educational contexts, text classification can be used to tailor learning materials to students' needs by automatically categorizing content based on difficulty level, topic, or other relevant criteria. For example, using semantic networks [5] can enhance the understanding of educational texts by representing relationships between concepts in a graph-like structure, making it easier for machine learning models to classify and generate educational content [5].

Various AI techniques such as Artificial Neural Networks (ANNs), Classification and Regression Trees (CARTs), and decision trees were evaluated for their ability to classify educational texts based on their knowledge content. The study [4] found that ANNs performed significantly better in distinguishing between texts designed to transfer knowledge and those that do not [4]. This highlights the potential of advanced AI techniques in improving educational outcomes through more precise text classification.

## Traditional Approaches to Text Clustering

Clustering techniques are widely used in data analysis to uncover hidden structures and patterns within large datasets, offering various methods to group similar data points. K-means, one of the most widely used clustering methods, partitions data into *k* clusters by minimizing the squared Euclidean distance between points and their respective cluster centroids. Although it is computationally efficient, K-means requires prior knowledge of the number of clusters and is sensitive to initial centroid placement [2]. Partitioning Around Medoids (PAM), a more robust alternative, uses actual data points (medoids) to represent clusters, making it less sensitive to outliers. PAM begins with an initial set of medoids and iteratively improves the clustering by swapping medoids with non-medoids to reduce the overall distance within clusters, though it can be computationally expensive [7]. Density-based clustering, such as DBSCAN, identifies clusters based on regions of high data density, making it particularly useful for discovering clusters of arbitrary shapes without needing to predefine the number of clusters [2]. Lastly, Gaussian Mixture Models (GMMs) take a probabilistic approach by assuming that the data is generated from a mixture of Gaussian distributions. GMMs allow for soft clustering, where each data point can belong to multiple clusters with varying probabilities, making them effective for handling overlapping clusters [2]. Each of these methods offers distinct advantages, with their application depending on the nature of the dataset and the clustering objectives.

# Background

Text classification is a crucial task in natural language processing (NLP), having evolved in tandem with advanced machine learning models that handle the complexities of language. In this project, we employed XLNet, a state-of-the-art model known for its strong performance across various text classification tasks [8] [9] [10] .We used XLNet to classify texts spanning different fields of mathematics and computer science. Additionally, we applied clustering techniques—namely K-means, PAM, Density Clustering, and Gaussian Mixture—to further analyze how the texts naturally group, thereby providing an unsupervised perspective for verifying the classification results.

## XLNet

XLNet is a powerful model developed for language understanding tasks, built upon the foundation of the Transformer-XL architecture. Unlike traditional models that process text sequentially, XLNet [6] analyzes all possible word arrangements (or permutations) in a sentence, allowing it to capture a broader context and develop a deeper understanding of language. This permutation-based approach enables XLNet to grasp complex language patterns that other models might miss, making it particularly effective in tasks such as text classification and document understanding [11].

* Permutation-based Training

A key innovation in XLNet is its permutation-based training strategy [Figure 1]. Traditional language models predict words in a fixed left-to-right or right-to-left order, which limits the model's ability to capture the full context of a sentence. XLNet, on the other hand, generates all possible permutations of the input sequence during training. This allows the model to predict words based on a richer context, where each word is influenced by all other words in different arrangements. By doing this, XLNet can better understand long-range dependencies and capture complex relationships between words, resulting in more accurate predictions [11].

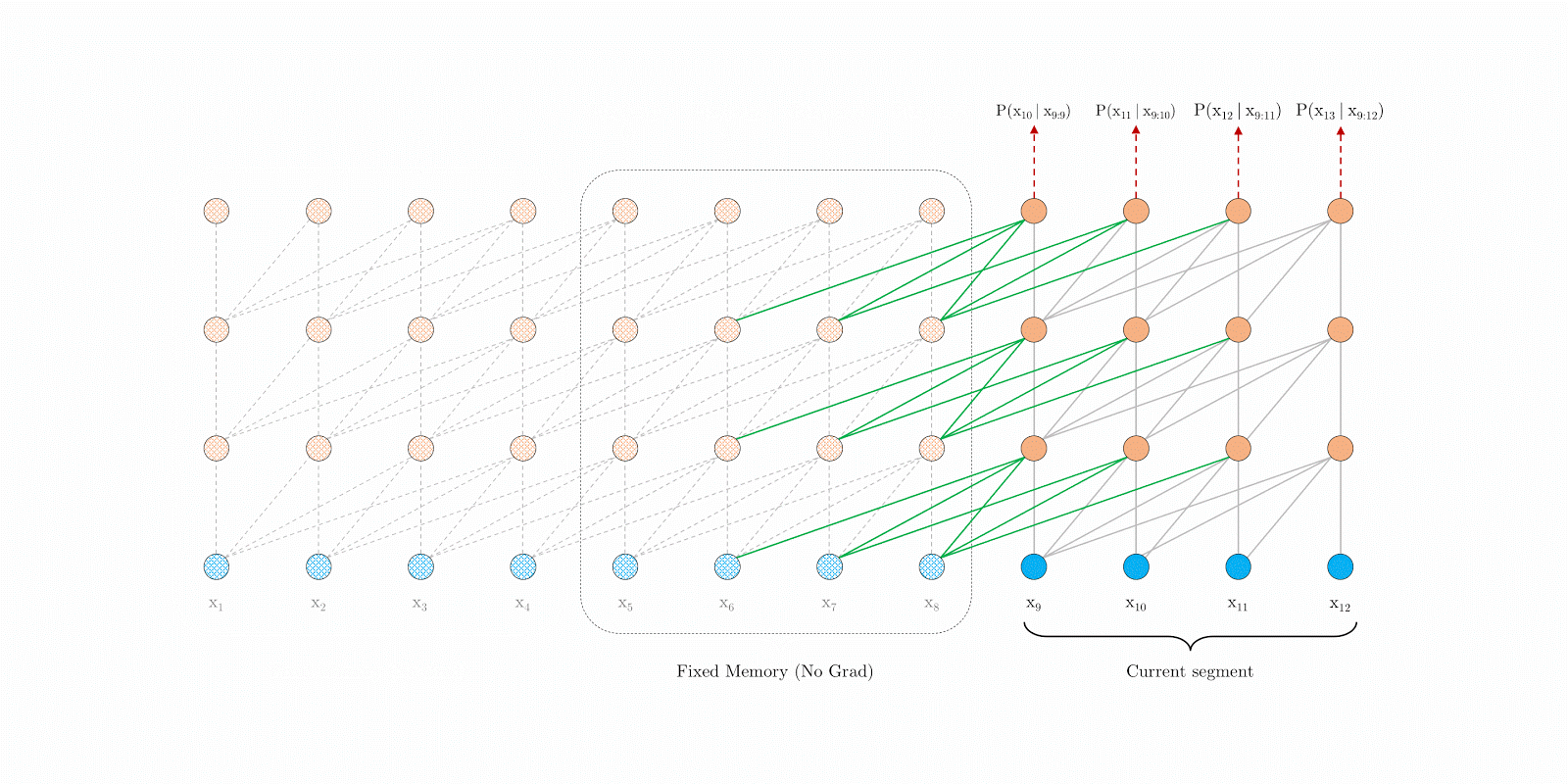


Figure 1: Permutation-based approach of XLNet, illustrating how the model processes different word arrangements to better capture contextual relationships within the text.

* Attention Masking

While processing permutations, XLNet uses attention masking to maintain the correct word order and ensure that each word is aware of its surrounding context. Attention masking allows the model to control what information is visible to each word during the learning process. For instance, if a word appears in the middle of a sentence, the attention mask will only allow XLNet to consider the relevant words in its context while ignoring others. This ensures that even though the model is processing different word permutations, it still retains the original sentence structure and meaning.

* Two-Stream Attention Mechanism

XLNet introduces a two-stream attention mechanism, which is designed to improve how the model predicts the next word in a sequence. In this setup, XLNet maintains two separate streams of information: one stream (the content stream) focuses on the actual content of the input tokens, while the other stream (the query stream) is used to predict the next word based on its position in the sentence. This architecture enables XLNet to handle bidirectional contexts more effectively, as the model is able to make position-aware predictions without knowing whether a word belongs to the sentence, improving its ability to generalize across different tasks. [Figure 3] presents an overview of the proposed permutation language modeling with two-stream attention [11].

* Segment Recurrence

One of the standout features of XLNet is its segment recurrence mechanism [Figure 2], borrowed from Transformer-XL, which allows the model to handle long sequences of text more efficiently. In traditional Transformer models, each segment of text is processed independently, which limits the model’s ability to retain information from previous segments. XLNet overcomes this limitation by reusing the hidden states from previous segments, allowing it to maintain a continuous flow of information across longer texts. This feature is particularly useful for processing documents that span multiple paragraphs, where understanding the entire context is critical.

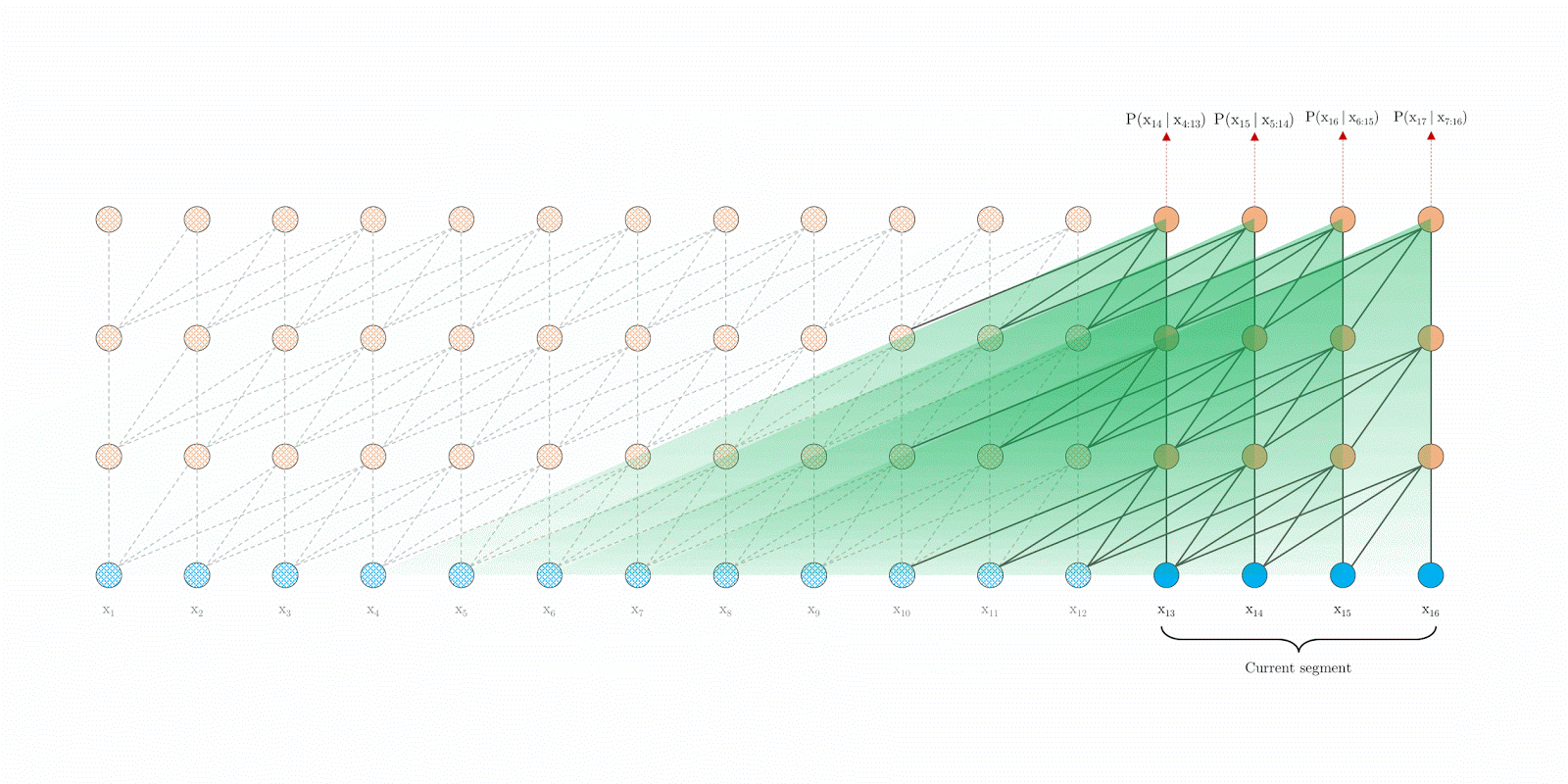


Figure 2: Segment recurrence mechanism in XLNet, showing how information from previous text segments is reused to enhance understanding of longer text sequences.

* Relative Positional Encoding

XLNet also improves upon traditional models with its use of relative positional encoding. Instead of relying on the absolute positions of words in a sentence, which can vary widely between different texts, XLNet focuses on the relative positions of words to one another. This allows the model to better understand how words relate within a given context, regardless of their specific order in the text. By focusing on the relative distance between words, XLNet can capture more flexible sentence structures, making it better suited for handling complex, non-linear sentence patterns.

* Autoregressive Objective

XLNet combines the best of autoregressive models and bidirectional models by using a generalized autoregressive training objective. This allows the model to generate word predictions based on both left-to-right and right-to-left contexts, providing a more complete understanding of each word’s role in the sentence. This autoregressive approach is a key factor behind XLNet’s ability to outperform many traditional models, as it can capture dependencies between distant words while still maintaining bidirectional context [11].

* Pretraining and Fine-tuning

Finally, like other state-of-the-art language models, XLNet undergoes two phases: pretraining and fine-tuning. During pretraining, the model is exposed to a large corpus of text data where it learns general language patterns using its permutation-based approach. After pretraining, XLNet can be fine-tuned for specific tasks such as text classification, question answering, or sentiment analysis. The flexibility of the model allows it to adapt to different tasks while maintaining high performance across various language-related benchmarks.

A diagram of a medical procedure

Description automatically generated with medium confidence

Figure 3: (a): Content stream attention, which is the same as the standard self-attention. (b): Query stream attention, which does not have access information about the content . (c): Overview of the permutation language modeling training with two-stream attention.

Overall, XLNet’s innovative architecture—built around permutation-based training, attention masking, and segment recurrence—makes it one of the most advanced models for natural language processing. Its ability to handle long-range dependencies, predict word order in both directions, and retain contextual information across lengthy texts gives it a significant advantage in tasks like text classification, question answering, and document understanding.

## K-Means Clustering

K-means is one of the most widely used clustering methods due to its simplicity and solid mathematical foundation. The core idea is to partition a dataset into *k* clusters, where each cluster is represented by its centroid, which is the arithmetic mean of the data points within that cluster. The goal is to minimize the total squared error, which represents the deviation of the data points from their respective cluster centroids. This is achieved by iteratively assigning data points to the nearest centroid and recalculating the centroids based on the updated clusters. The algorithm [Figure 4] continues until convergence, where the positions of the centroids and the cluster assignments no longer change. While easy to implement and computationally efficient, K-means has some limitations, such as sensitivity to the initial placement of centroids and the need to specify the number of clusters beforehand.

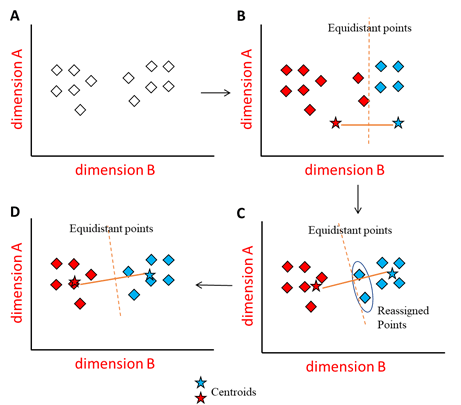


Figure 4: This illustration outlines the steps of the K-means clustering algorithm, including the initial random placement of centroids (Panel A), the assignment of points to centroids (Panel B), the adjustment of centroids based on their clusters (Panel C), and the final stabilization of clusters (Panel D).

## PAM (Partitioning Around Medoids)

PAM is a clustering technique that improves upon K-means by using actual data points, called "medoids," to represent clusters. A medoid is the most centrally located data point within a cluster, making it more representative and robust to outliers than centroids, which are average values in K-means. The process starts with the BUILD algorithm, which selects an initial set of medoids by iteratively choosing points that minimize the total distance between them and all other data points. After this initialization, the SWAP algorithm is used to refine the clusters by swapping medoids with non-medoids to further reduce the total distance within the clusters. This iterative process continues until no further improvements can be made. While PAM is more accurate and resistant to outliers compared to K-means, it is computationally expensive, requiring O(n²) time for distance calculations, which limits its scalability for large datasets.

## Density-Based Clustering

Density-based clustering identifies clusters based on regions of high object density, separated by areas of low density. This approach assumes that data points are sampled from an unknown underlying probability distribution, and clusters are defined as dense regions of the data space. The most well-known density-based clustering algorithm is DBSCAN, which groups points that are closely packed together and marks points in low-density regions as outliers. Unlike methods such as K-means [Figure 5], density-based clustering does not require specifying the number of clusters in advance and can handle clusters of arbitrary shapes and sizes. The flexibility of this approach makes it particularly useful for identifying complex structures in data. Density-based methods are also robust to noise and can effectively detect outliers.

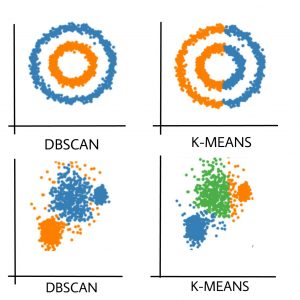


Figure 5: DBSCAN accurately captures the circular patterns and identifies distinct clusters in dense areas, whereas K-means segments the data into pie slices and arbitrary groups, failing to conform to the natural data shapes.

## Gaussian Mixture Models (GMMs)

Gaussian Mixture Models (GMMs) approach clustering by assuming that data points are generated from a mixture of several Gaussian distributions. A Gaussian distribution, or normal distribution, is a bell-shaped curve that represents how data points are distributed around a mean, with most points close to the mean and fewer points as you move further away. In GMMs, each Gaussian distribution represents a cluster, and data points are assigned to these clusters based on the probability of belonging to each distribution. These probabilities come from how closely a data point matches the characteristics (mean and variance) of each Gaussian distribution.

To fit the model, GMMs use the Expectation-Maximization (EM) algorithm, which finds the parameters (mean and variance) of each Gaussian distribution. EM iteratively adjusts these parameters to improve the match between the model and the data until it converges on the best possible fit. GMMs are particularly effective when the data follows a normal distribution, and they work well with overlapping clusters, which methods like K-means struggle to handle. GMMs also allow for soft clustering, meaning that each data point can belong to multiple clusters with varying probabilities, rather than being assigned to just one cluster. However, the method relies on knowing the number of distributions (or clusters) in advance, and its performance can degrade if the assumed distribution does not match the data well.

# System Implementation

This section outlines the processes we used for **dataset collection, text preprocessing, text classification, and text clustering**, including the models and techniques we applied at each stage. We begin by discussing how we gathered and filtered our textual data, proceed to describe our preprocessing pipeline for removing domain-specific terms, and then detail the steps for both classification (using XLNet) and clustering (using K-means, PAM, Density Clustering, and Gaussian Mixture).

## Dataset Collection

Our dataset was manually compiled from publicly accessible academic sources, focusing on mathematics and computer science textbooks that were free to access. Each book was individually reviewed to confirm its suitability for our research objectives. This process yielded a mix of **original PDF files** (which contain digitally embedded text) and **scanned PDFs** (in which text is rendered as images).

To convert the original PDFs into machine-readable text, we experimented with several text extraction libraries:

* **PyMuPDF (Fitz)**
* **pdfminer.six**
* **PyPDF2**

Each library varies in how it handles complex PDF elements such as embedded fonts, unusual encodings, and intricate metadata layers. We compared their outputs on a subset of PDFs to determine which approach yielded the most accurate and reliable text. In most cases, *PyMuPDF* emerged as a strong balance between speed and fidelity, while *pdfminer.six* served as a fallback in specific edge cases.

For **scanned PDFs**, where the document pages are stored purely as images, we employed **Tesseract** [12], an optical character recognition (OCR) engine, to convert images into searchable text. This process involved splitting each PDF into individual page images and then running the OCR tool on those images. The resulting text files—whether obtained via PDF parsing or OCR—were subsequently integrated into our dataset.

## Text Preprocessing

The text preprocessing phase was **crucial** for ensuring that our dataset was consistently structured before clustering and classification. We performed **manual review** across all books because the PDF-to-text conversion process may introduce artifacts into the extracted text, which required manual verification to prevent corrupted data from being included in the final dataset.

Additionally, **we eliminated sections not directly related** to the subject matter, such as bibliographies, tables of contents, and indices.

Once the dataset was pruned of these extraneous sections, we **normalized** the raw text by:

1. **Converting** all characters to lowercase.
2. **Removing** punctuation marks, numerical digits, and extraneous whitespace.
3. **Filtering out** non-alphabetic content using regular expressions.

Because PDF parsing and OCR can introduce noise (e.g., broken words or stray symbols), we **further cleaned** the text by removing any tokens not listed in a standard English dictionary file (oxford.txt). This step reduced obscure artifacts that might skew our analysis.

Next, we used the **Natural Language Toolkit (NLTK)** to **lemmatize** each token, mapping words to their base forms while respecting part-of-speech tags.

A key element of our preprocessing was **removing domain-specific terminology**. We compiled curated lists of specialized terms for both mathematics and computer science (including 1-gram, 2-gram, and 3-gram entries). Any match was replaced with a placeholder (<TERM>) or deleted from the text. For example, a multi-word phrase like “linked list” or “partial derivative” was treated as a single domain term. This ensured that none of these overt “giveaways” would remain, forcing our subsequent analysis to focus on the **metalanguage** rather than subject-specific keywords.

Finally, the **preprocessed** text was **segmented** into chunks. To balance meaningful context with computational constraints, we typically used **256 tokens** per chunk, though we experimented with smaller or larger sizes as needed. These segmented chunks served as the input for both our clustering pipeline (using Doc2Vec embeddings) and our classification model (XLNet).

## Text Classification

After removing domain-specific terminology from each text and dividing it into chunks, we classified the remaining content (what's left after text preprocessing step) using **XLNet**, a transformer-based model known for its ability to capture both short- and long-range linguistic dependencies [10]. Our primary objective was to determine whether these texts still contain enough distinctive features (metalanguage) to differentiate between various fields of mathematics and computer science.

**Model Setup**

* We used the **Hugging Face** transformers library to load and fine-tune the *xlnet-base-cased* model.
* The preprocessed text chunks were tokenized (up to **512 tokens** in length) and then fed directly into XLNet.

**Hyperparameter Choices**

* **Sequence Length:** We typically set a maximum input size of **512 tokens** in the tokenizer. In practice, our segments often hovered around **256 tokens** after preprocessing.
* **Batch Size:** We experimented with small batch sizes (e.g., **4** or **8**) due to memory constraints and to help the model generalize.
* **Learning Rate & Training Schedule:** We used a **2e-5** learning rate and fine-tuned for about **3–5 epochs**, monitoring validation loss to avoid overfitting.
* **Regularization:** By default, XLNet includes a modest dropout (often 0.1) within its architecture. We also applied *weight decay* (0.01) to the optimizer, encouraging the model to avoid overly large parameter values.

**Training & Evaluation**

* We randomly split the final dataset into **train**, **validation**, and **test** sets (e.g., 80%/10%/10%). Each text chunk was associated with a numeric *label\_id*, ensuring consistent ground truth.
* During training, we tracked **accuracy**, **precision**, **recall**, and **F1** at each epoch. Upon concluding training, we made predictions on the held-out test set to compute final performance metrics.
* We leveraged the Trainer API from Hugging Face, which automates data batching, gradient accumulation, and evaluation steps. This approach streamlined the process of fine-tuning XLNet and analyzing performance.

By fine-tuning XLNet in this manner, we could determine how well the model distinguishes academic texts across multiple fields—even after the removal of domain-specific terms.

* 1. Text Clustering

After removing domain-specific terminology and segmenting each document, we initially **experimented with both TF-IDF vectors and Doc2Vec embeddings** to encode each text chunk numerically. Through preliminary tests, we found that **Doc2Vec** [1] often yielded better clustering coherence and more stable results than TF-IDF, because Doc2Vec captures semantic relationships beyond simple word counts. Consequently, we relied on **Doc2Vec** vectors in our final clustering experiments, although some TF-IDF trials were performed for comparison.

We then tested several clustering algorithms—K-means, PAM (K-Medoids), DBSCAN, and Gaussian Mixture Models (GMMs)—to see whether texts naturally group together. We implemented each clustering algorithm through *scikit-learn* and *scikit-learn-extra* [13].

Below are the main steps and parameters we used for each approach:

1. **K-Means**
   * **Number of Clusters (k):** We examined different values of *k*.
   * **Initialization (k-means++):** Implemented by default in *scikit-learn* to ensure well-distributed initial centroids.
   * We computed **Adjusted Rand Index (ARI)**, **Adjusted Mutual Information (AMI)**, **Normalized Mutual Information (NMI)**, **Silhouette Score**, and other metrics to measure cluster quality in relation to known labels.
2. **PAM (Partitioning Around Medoids)**
   * We employed the KMedoids class from *scikit-learn-extra* and used **Euclidean distance**.
   * **Number of Clusters:** As with K-means, we tested a handful of *k* values and evaluated the clusters via Silhouette and Davies-Bouldin indices.
   * **Medoid Selection & Iterations:** The algorithm’s *BUILD* phase chose initial medoids, followed by *SWAP* iterations for refinement.
3. **Density-Based Clustering (DBSCAN)**
   * **Outcome:** DBSCAN clustering on 50,147 Doc2Vec embeddings with eps=0.593, min\_samples=5, and cosine similarity yielded 13 clusters with 2,647 noise points, aligning with the expected cluster structure. However, the evaluation metrics—Davies-Bouldin index of 2.9632, and Calinski-Harabasz index of 20.7728—indicate that the clusters are not very well separated.
4. **Gaussian Mixture Models (GMMs)**
   * We fit GMMs across a range (2–14 components) and relied on **BIC (Bayesian Information Criterion)** and **AIC (Akaike Information Criterion)** to pick the optimal number.

**Validation & Visualization**

* **Comparison with True Labels:** Although clustering is unsupervised, we mapped cluster assignments against each text’s field label to compute metrics like ARI, AMI, and NMI.
* **Dimensionality Reduction:** We used **PCA** (down to three components) to visualize cluster structures, offering a more interpretable view of the embedding space.

In summary, **Doc2Vec** provided richer embeddings for our domain-stripped texts, ultimately improving cluster cohesion across various algorithms. By examining the alignment of these clusters with known labels, we gained additional insight into the extent to which metalanguage alone can differentiate between the original academic fields.

# Achieved Results

In this section, we present what we found in our **classification** and **clustering** experiments. First, we show how **XLNet** performed on math and computer science subfields after we removed their domain-specific words. Next, we explain the results of using **K-Means** and **Gaussian Mixture Models** to group the same texts without labels. Finally, we give a short conclusion about how math and computer science texts differ in writing style, yet sometimes overlap within closely related subfields.

## Classification Performance

After training the **XLNet** model on our domain-agnostic text segments, we achieved an overall **accuracy of 83%** on the test set (5,015 samples). The **weighted F1-score** also stood at **0.83**, indicating generally robust performance across multiple classes. Figure 6 shows the detailed classification report, while Figure 7 presents the confusion matrix.

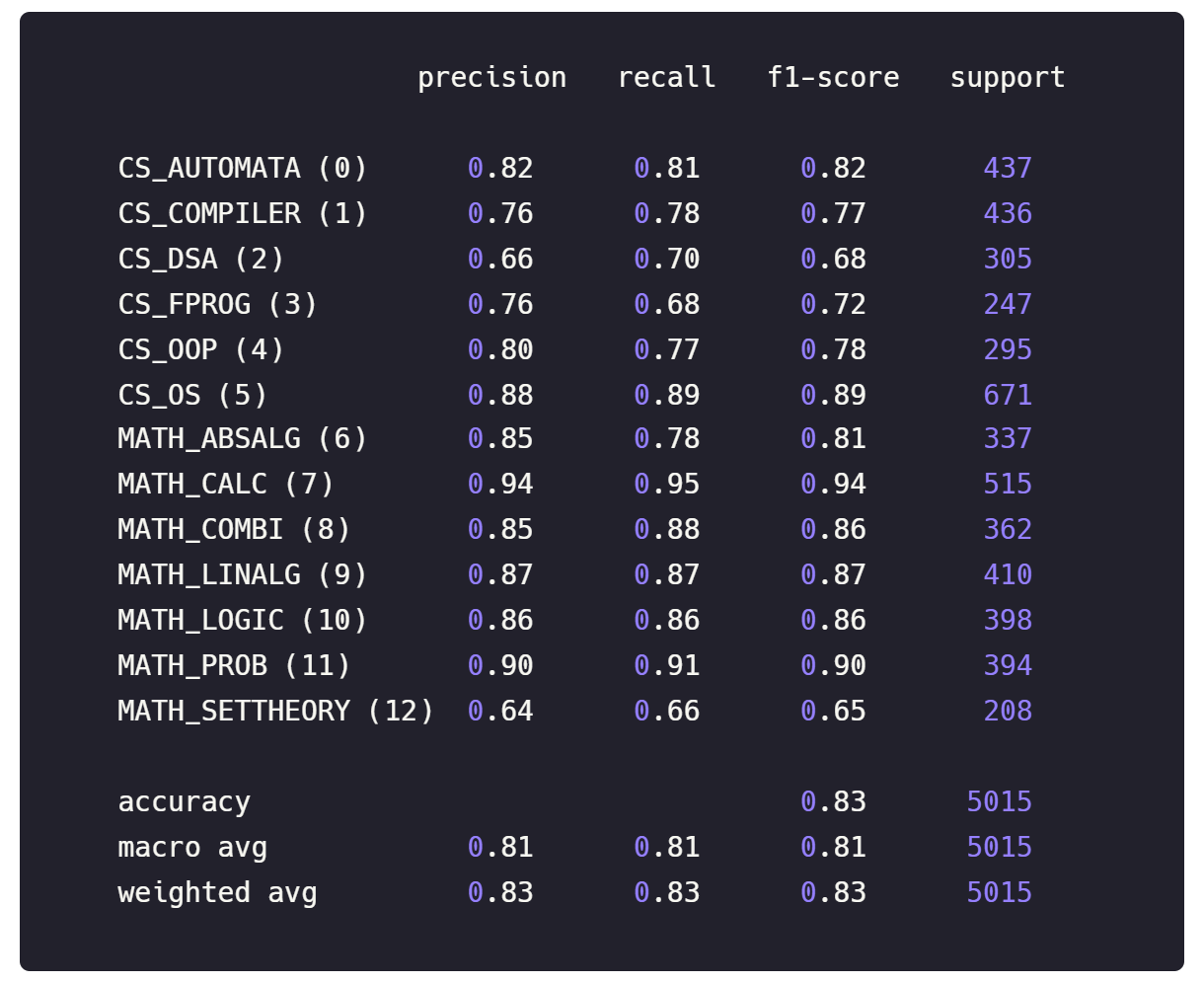


Figure 6: Classification Report.

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Figure 7: Confusion Matrix.

From the confusion matrix, several observations stand out:

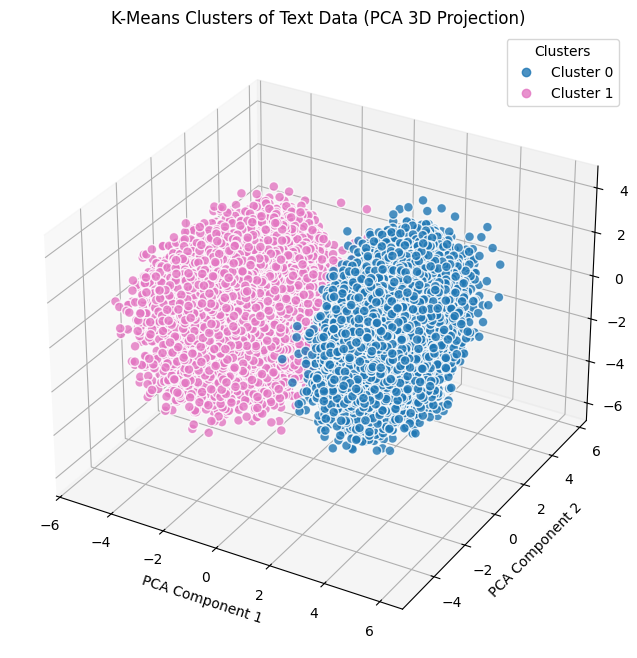
1. **Clear Separation of Major Fields**  
   Texts labeled as *CS\_OS* (label 5) and *MATH\_CALC* (label 7), for instance, achieve notably high recall, indicating the model readily distinguishes many Computer Science vs. Mathematics segments even without domain-specific keywords.
2. **Overlaps in Closely Related Subfields**  
   For instance, *MATH\_SETTHEORY* (label 12) exhibits lower recall (0.66) and is sometimes confused with other advanced mathematics fields (e.g., MATH\_ABSALG). These subfields share enough structural and linguistic similarities to challenge the model.
3. **Moderate Overall Accuracy (83%)**  
   This level of performance aligns well with our **Phase A** prediction that an “intermediate” accuracy would emerge if each subfield’s underlying *metalanguage* retained distinguishing signals but was not entirely unique.

## Clustering Outcomes

We evaluated **K-Means** and **Gaussian Mixture Models** on the same Doc2Vec-encoded text segments used for classification. Our aim was to see if an unsupervised approach would naturally form clusters corresponding to the fields that XLNet distinguished.

**K-Means**

* **K=2 Clusters**
  + Produced a clear **two-group division**, which strongly correlates with **Mathematics vs. Computer Science** when compared to the true labels.
  + We observed a **high Adjusted Rand Index (ARI)**, indicating that these two clusters closely match the broad Math/CS distinction already identified by XLNet.
  + However, this approach collapses all subfields within Math or CS into just one cluster each, providing no finer-grained separation.



Изображение выглядит как диаграмма, текст, снимок экрана, Красочность

Автоматически созданное описание

* **K=13 Clusters**
  + Attempted to match the **13 actual subfields** (e.g., *MATH\_CALC*, *CS\_AUTOMATA*, etc.).
  + Although some subfields dominated certain clusters, there was notable overlap among closely related areas (e.g., *Set Theory* and *Abstract Algebra*).
  + The resulting **ARI** and **NMI** scores dropped compared to *k=2*, indicating that these subfields’ linguistic similarities make a purely unsupervised grouping less distinct than the broad Math vs. CS split.

Изображение выглядит как текст, диаграмма, снимок экрана, дизайн

Автоматически созданное описание

Изображение выглядит как текст, диаграмма, снимок экрана, дизайн

Автоматически созданное описание

**Gaussian Mixture Models (GMM)**

* **Optimal Number of Components**
  + By fitting models from **2** to **14** components and evaluating **BIC/AIC**, the algorithm consistently indicated **2 components** as the best fit.
  + Similar to K-Means with *k=2*, GMM separated the data into two major clusters that aligned strongly with Math vs. CS when we cross-referenced the known labels.

Изображение выглядит как текст, диаграмма, снимок экрана, дизайн

Автоматически созданное описание

* **Subfield Separation**
  + Like K-Means, GMM did **not** further subdivide the mathematics cluster or the CS cluster in a meaningful way when asked to find more than 2 components. The BIC/AIC curves tended to worsen with higher component counts, underscoring the difficulty of splitting subfields based on the remaining “metalanguage.”

Изображение выглядит как линия, График, текст, диаграмма

Автоматически созданное описание

**Alignment with XLNet Classification**

Overall, both K-Means and GMM **validate** the **broad** distinction that XLNet detected (Math vs. CS). Nevertheless, **purely unsupervised methods encountered difficulties** in recreating the 13-subfield division, suggesting that although domain-agnostic texts retain enough linguistic cues to broadly separate Math from CS, **the finer distinctions** among subfields remain too subtle for reliable detection without supervision.

## Overall Conclusion

Our initial motivation was to investigate whether mathematics and computer science share a common “metalanguage” after domain-specific terminology is removed, and how that might correlate with students’ proficiency in both disciplines. The results indicate that **Mathematics and Computer Science indeed use distinct underlying language structures**, as evidenced by the model’s ability to classify texts in each domain and by the clustering analyses that naturally split the data into broad Math vs. CS groups.

At the same time, we observed that **subfields within the same discipline** (e.g., Set Theory vs. Abstract Algebra) often exhibit overlapping linguistic patterns, making them difficult to distinguish purely from metalanguage features. This finding aligns with the educational reality that excelling in one area of mathematics (or computer science) frequently transfers at least some advantage to related subfields due to shared language constructs.

However, we also found **no guaranteed correlation** between proficiency in Math and proficiency in CS—each discipline appears to demand its own cognitive and linguistic framework. In other words, being good at mathematics does not necessarily imply strength in computer science, or vice versa.

# Challenges

Several challenges are anticipated throughout the course of this project, particularly given the novel approach of removing domain-specific terminology and relying solely on general linguistic patterns (metalanguage) for classification and clustering. These challenges span from dataset preparation to model optimization and clustering validation.

## Preprocessing Difficulties

Preprocessing is complex phase of the project, involving multiple challenges:

* Irrelevant Pages:

Identifying and removing irrelevant sections, such as bibliographies and indices, is complicated by the inconsistent structure of academic books. We **manual checked** each book to ensure that they have only relevant sections.

* Character Cleaning:

Text extraction from scanned PDFs often introduces unwanted characters/words/corruptions. Removing these data reduced our dataset size, forcing us to gather **additional books** to maintain variety.

* Removing Domain-Specific Terms:

Building an effective dictionary of domain-specific words from both mathematics and computer science was critical to this project. This dictionary must capture a wide range of terms, including multi-word phrases. Building dictionaries to capture specialized terms in multiple subfields was **time-consuming**. It was also challenging to decide which words are truly domain-specific versus general English (e.g., “for” can be both a normal preposition and a programming keyword).

# Future Work

1. **Context-Aware Term Identification**

While we relied on manually curated dictionaries to remove domain-specific terms, a next step could involve building an **automated model** that recognizes technical terms based on context. Such a system would differentiate between everyday usage (“for example”) and specialized meanings (e.g., “for loop”), reducing the risk of misclassifying common words as domain terminology.

1. **Automated Relevance Detection**

Another avenue for improvement is **automating the extraction of relevant sections** from PDF documents before or after text conversion. Rather than relying on manual checks, a specialized algorithm could detect (and potentially exclude) indices, bibliographies, or heavy formula/code segments. This automation would streamline preprocessing, minimize human intervention, and help maintain consistent text quality for classification and clustering.

By advancing these two areas—context-aware terminology filtering and automated PDF relevance detection—we can further refine the pipeline, potentially enhancing both the accuracy and scalability of our method.

# Bibliography

|  |  |
| --- | --- |
| [1] | Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *31st International Conference on Machine Learning (ICML)*, 2014. |
| [2] | J.-O. Kapp-Joswig and B. G. Keller, "Clustering—Basic concepts and methods," *Journal of Chemical Theory and Computation,* 2021. |
| [3] | S. Minaee et al., "Deep learning based text classification: A comprehensive review," *ACM Computing Surveys (CSUR),* 2021. |
| [4] | T. Horáková, M. Houška, and L. Dömeová, "Classification of the educational texts styles with the methods of artificial intelligence," *Journal of Baltic Science Education,* 2017. |
| [5] | B. Touis, S. Warraki, O. El, and J. Lahiassi, "Using semantic networks for text classification in education: Generating tailored questions for students," *Proc. Int. Conf. E-Learning Smart Eng. Syst. (ELSES),* 2023. |
| [6] | Z. Yang et al., "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. 33rd Conf. Neural Information Process. Syst. (NeurIPS)*, 2019. |
| [7] | E. Schubert and P. J. Rousseeuw, "Faster k-medoids clustering: Improving the PAM, CLARA, and CLARANS algorithms," *Journal of Statistical Software,* 2022. |
| [8] | S. Vijayarani and S. Janani, "A survey on text classification algorithms," *nternational Journal of Computer Science and Information Technologies,* 2016. |
| [9] | Md. I. Hossain, H. U. Khan, M. S. Irfan, and M. R. Basir, "A survey on text classification algorithms from text to predictions," *International Journal of Innovative Research in Computer and Communication Engineering,* 2019. |
| [10] | S. Roy, "A survey on text classification from traditional to deep learning," *Journal of Computing Technologies,* 2020. |
| [11] | Research Data Pod, "Paper Reading: XLNet Explained," 8 9 2023. [Online]. Available: https://researchdatapod.com/paper-reading-xlnet-explained/. [Accessed 21 9 2024]. |
| [12] | R. Smith, "An overview of the Tesseract OCR engine," in *9th International Conference on Document Analysis and Recognition (ICDAR)*, 2007. |
| [13] | F. P. e. al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research,* p. 2825–2830, 2011. |