ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

The utilization of LLM synthetic data in Machine Learning tasks

Μίγκος Ηλίας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπεύθυνος

Βασίλειος Πλαγιανάκος

Λαμία, Σεπτέμβριος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

The utilization of LLM synthetic data in Machine Learning tasks

Μίγκος Ηλίας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων

Βασίλειος Πλαγιανάκος

Λαμία, Σεπτέμβριος 2025

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις (1), που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. *Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή.*
2. *Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*
3. *Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια.*
4. *Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: August 27, 2025

Ο – Η Δηλ.

(Υπογραφή)

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

The utilization of LLM synthetic data in Machine Learning tasks

Μίγκος Ηλίας

Τριμελής Επιτροπή:

Contents

Abstract

The rapid advancement of Large Language Models (LLMs) has opened new avenues in data augmentation, particularly through the generation of synthetic text data. This thesis explores the utility of LLM-generated synthetic data in supervised machine learning tasks, focusing on multiclass text classification. A real-world dataset of book descriptions categorized into seven classes was used to construct three datasets: (1) real data only, (2) synthetic data generated by LLMs, and (3) a mixed dataset combining real and synthetic entries. Text data underwent standard preprocessing, tokenization, and word embedding using pre-trained GloVe vectors. Identical model architectures were used for all three to ensure fair comparison. The performance of models trained on each dataset was evaluated in terms of accuracy and generalization. Results indicate that while models trained on real data achieve strong baseline performance, synthetic data alone can offer comparable results. In particular, the model trained on a mixed dataset demonstrated improved generalization and accuracy. These findings highlight the potential of LLM-generated synthetic data as a viable tool for enhancing machine learning models, particularly in scenarios involving data imbalance or scarcity.

1 Introduction

1.1 Background and Motivation

The use of Large Language Model (LLM)-generated synthetic data is a growing approach in machine learning, aimed at addressing challenges such as data scarcity and privacy concerns. Traditional machine learning models rely on large, high-quality datasets for effective training, but collecting such data can be expensive, time-consuming, or even impossible in specialized domains or when sensitive information is involved [1]. Synthetic data, produced by Natural Language Processing (NLP) models to approximate the properties of real data, provides a practical alternative [2]. With their strong text generation abilities, LLMs have made it easier to create synthetic datasets that can be used for data augmentation and model training [3].

The value of synthetic data lies in its ability to solve several key issues. In situations where real data are scarce, synthetic samples can expand training sets and improve both performance and generalization [4]. Synthetic data can also support privacy by masking or removing sensitive information, allowing models to be developed without exposing confidential details. However, challenges remain: the usefulness of synthetic data depends heavily on its quality and realism, and poor or biased samples can reduce accuracy and fairness.

This thesis investigates the role of LLM-generated synthetic data in supervised machine learning, focusing on multiclass text classification. By comparing models trained on real, synthetic, and mixed datasets, the study aims to assess both the benefits and the limitations of using synthetic data to enhance machine learning performance.

1.2 Contribution

The main goal of this thesis is to test whether text generated by a Large Language Model (LLM) can be used as a reliable resource for training classification models, and to examine how this synthetic data interacts with real samples to affect overall performance. The results show that synthetic-only training can reach performance levels close to real data, while the mixed dataset achieved the strongest results across all seven categories.

This study compares outcomes under identical experimental conditions, giving clear evidence of the standalone value of synthetic data. It also shows that mixing synthetic and real examples reduces the weaknesses

of each source, improving accuracy, lowering overfitting, and supporting minority classes.

Future work could explore different LLMs, embedding methods, or architectures to see whether they further improve the usefulness of synthetic data. Another direction is adaptive prompt design to generate more domain-specific text, such as for medical or legal tasks. Additional research could focus on better metrics for evaluating the quality and diversity of synthetic data, as well as human-in-the-loop methods that combine automated generation with expert review.

2 Literature Review

2.1 Introduction to Text Classification

Text classification represents a cornerstone of modern natural language processing (NLP), enabling machines to automatically categorize unstructured text into predefined classes through learned patterns from labeled examples. This technique is utilized in critical applications across industries, from filtering spam emails (binary classification), to organizing literary genres (multiclass categorization) [5].At its core, the process involves training algorithms on annotated datasets where human experts have assigned category labels, allowing models to discern linguistic patterns that distinguish different classes.

The practical implementations of text classification span diverse domains, each requiring tailored approaches. Sentiment analysis systems deployed by companies analyze customer reviews to gauge satisfaction levels, enabling proactive service improvements [6].In healthcare, classification models process patient feedback to identify trends in care quality [7], while financial institutions employ sentiment-aware text classification models-such as the Polarity Pattern Approach (PPA) and LSTM-based frameworks-to systematically analyze news headlines and search trends, thereby enhancing their ability to forecast stock returns and trading volumes based on the polarity of financial news[8] [9] .

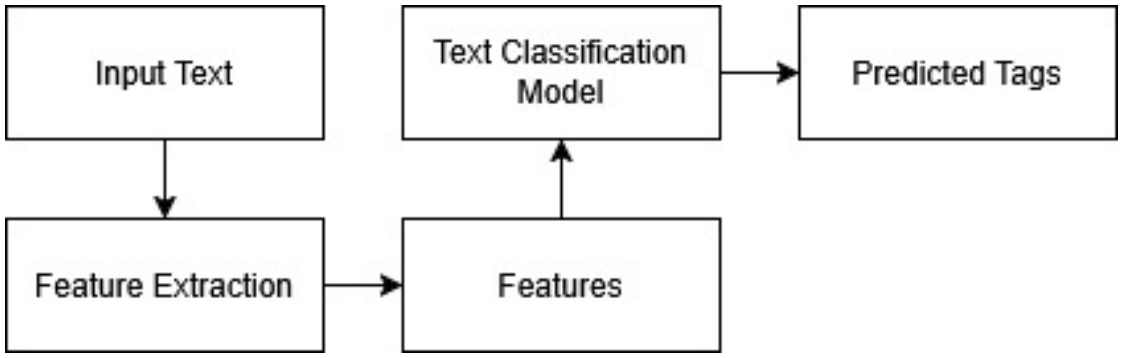

Figure 1: Simple text classification pipeline.

2.2 Evolution and Mechanisms of Text Classification

Text classification has evolved significantly over the past several decades, reflecting advances in computational power, algorithmic design, and linguistic understanding. Initially, text classification relied heavily on traditional statistical models such as Bayesian classifiers and logistic regression, which utilized handcrafted features like term frequency-inverse document frequency (TF-IDF) to represent text data numerically. These early approaches treated text as a bag of words, ignoring word order and contextual relationships, which limited their ability to capture the full semantic meaning of documents. [10] Despite these limitations, such models laid the foundation for automated text categorization and were widely applied in spam filtering, document organization, and sentiment analysis.

The advent of machine learning introduced more sophisticated algorithms, including Support Vector Machines (SVMs) and decision trees, which improved classification accuracy by better handling high-dimensional feature spaces and nonlinear decision boundaries [11] Subsequently, the integration of Natural Language Processing (NLP) techniques enabled models to incorporate syntactic and semantic information, enhancing their ability to interpret text beyond simple keyword matching. Deep learning architectures, notably Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), marked a major shift by automatically learning hierarchical and sequential features from raw text data. CNNs effectively captured local patterns such as n-grams, while RNNs modeled temporal dependencies in sentences, enabling improved performance in tasks requiring contextual understanding [10].

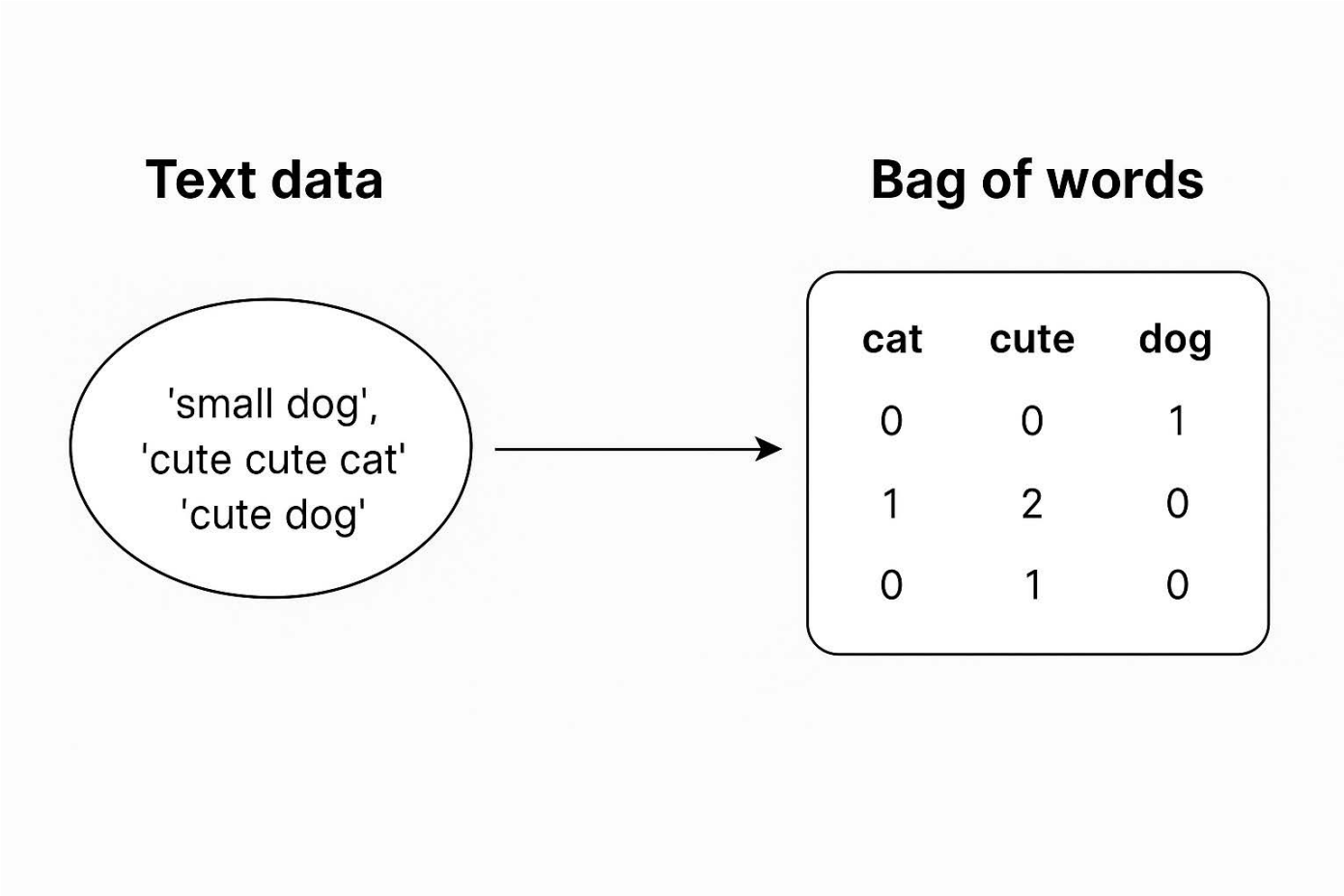

Figure 2: Bag-of-Words Representation of Text Data

2.3 The Critical Role of Data Quality in Machine Learning Systems

The foundation of robust machine learning systems depends on the quality and balance of training data. The need for high-quality, well-balanced datasets is clear, as organizations face performance degradation, extended development cycles, and operational failures stemming from inadequate data preparation. Recent studies show that data quality issues represent one of the most significant barriers to successful AI deployment, causing challenges that extend far beyond the initial training stage.

Contemporary research demonstrates that even state-of-the-art architectures remain vulnerable to degradation when exposed to low-quality inputs, with error propagation manifesting across the model lifecycle. A 2024 study in Nature Digital Medicine introduced the METRIC framework, which quantifies 15 dimensions

of medical data quality - including label accuracy, feature relevance, and temporal consistency - showing an accuracy improvement in deep learning models when optimizing these parameters [12]. This aligns with theoretical analyses from computational learning theory, where the signal-to-noise ratio (SNR) of training data directly bounds model performance through the deterministic-non-deterministic ratio (DDR) metric[13].

2.4 The Problem of Class Imbalance

Dataset imbalance is one of the most critical but often overlooked issues of data quality in machine learning applications. Unbalanced datasets occur when certain classes or categories are significantly underrepresented compared to others, leading to biased models that perform poorly on minority classes even if overall accuracy looks acceptable. [14][15]. The challenge becomes particularly acute in real-world applications where the minority classes often represent the most critical cases requiring accurate prediction.

The mathematical foundations of machine learning algorithms inherently favor majority classes when training on imbalanced data. Standard optimization objectives like accuracy maximization bias models toward predicting the most frequent class, as this strategy yields the highest overall accuracy even while completely failing to detect minority class instances [16]. This phenomenon explains why models trained on imbalanced datasets often achieve high accuracy scores while demonstrating poor practical utility in detecting rare but important events [14].

The implications of dataset imbalance extend throughout the entire machine learning pipeline, affecting not only model training but also evaluation, deployment, and maintenance phases. Models trained on imbalanced data typically exhibit poor generalization to new data distributions, particularly when the test environment contains different class proportions than the training set [15]. This distribution shift creates significant challenges for model deployment in production environments where data characteristics may evolve over time.

2.5 Synthetic Data Generation in Machine Learning

Synthetic data generation has emerged as a crucial technique in machine learning to address challenges related to data scarcity, privacy, and data quality. It involves creating artificial data that preserves the statistical properties and underlying patterns of real-world datasets, enabling model training without relying solely on

limited or sensitive real data [17].

Several machine learning approaches have been developed for synthetic data generation. Among the most prominent are generative models such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and large language models (LLMs). These models learn latent representations of the original data distribution and generate new samples that mimic real data characteristics [18][19].For example, GANs pit two neural networks against each other to produce realistic synthetic data, while VAEs encode data into a latent space and decode it back to generate new samples [17].

The applications of synthetic data span multiple domains including computer vision, natural language processing, healthcare, and business analytics. In text data specifically, synthetic data generation can augment datasets to improve model robustness, especially in low-resource settings or when facing class imbalance. Synthetic data can also facilitate privacy preservation by enabling data sharing without exposing sensitive information.

Despite its advantages, synthetic data generation faces challenges such as computational complexity, training instability, and ensuring synthetic data's fidelity and diversity. Moreover, ethical concerns arise regarding biases embedded in synthetic data and potential misuse. Addressing these issues is critical for the broader adoption and trustworthiness of synthetic data in machine learning workflows.

In summary, synthetic data generation using machine learning models, particularly generative models and LLMs, offers a promising avenue to overcome data limitations and enhance model performance across various tasks. Ongoing research continues to refine these techniques to balance data utility, privacy, and ethical considerations [17].

2.5.1 Traditional Text Data Augmentation Techniques

Traditional text data augmentation techniques have long helped improve the performance of NLP models, especially when labeled data is limited or imbalanced. Among the most widely used methods are synonym replacement, random insertion and deletion, back translation and paraphrasing

Synonym replacement involves substituting words in a sentence with their synonyms, thereby creating new versions of the text while keeping its meaning. This technique is straightforward and helps models become less sensitive to specific word choices. However, its effectiveness can be inconsistent, as inappropriate

synonym choices may change the intended meaning or make the sentence sound unnatural. Studies have found that while synonym replacement can improve performance, its impact varies depending on the language, task, and model architecture [20][21].

Random insertion and deletion refer to randomly adding or removing words within a sentence. These token level manipulations introduce noise and diversity into the training data, encouraging models to learn more robust representations. These methods are effective for morphologically rich languages and some sequence tagging tasks, though less useful for analytic languages [20].

Back-translation is a powerful augmentation technique where a sentence is translated into another language and then translated back to the original language. This process often results in paraphrased versions of the source text, adding linguistic diversity and helping to reduce overfitting. Empirical studies have shown that back-translation significantly enhances classification accuracy and model robustness, particularly in low-resource and fake news detection tasks [21][22].

Paraphrasing involves generating semantically similar sentences using rule-based or model-based approaches. While back translation is a form of paraphrasing, dedicated models or templates can also generate paraphrases. Paraphrasing expands the dataset with diverse expressions of the same underlying meaning, which is beneficial for generalization and handling varied real world inputs [23][22].

Comparative studies indicate that the effectiveness of these traditional augmentation techniques depends heavily on the specific NLP task, language characteristics, and the underlying model. For example, character level and token level augmentations tend to be more effective for certain architectures, while syntactic and semantic methods like back translation and paraphrasing usually give more consistent improvements for higher level tasks [20][23][22]. Overall, these methods form the foundation upon which more advanced, model driven augmentation strategies such as those using LLMs are built.

2.6 Large Language Models (LLMs) and Their Applications

Large Language Models (LLMs) have greatly advanced natural language processing (NLP) by enabling machines to understand and generate human like text with strong fluency and coherence. LLMs are typically based on deep learning architectures such as the Transformer, which use self attention mechanisms to capture long range dependencies in text data efficiently.

Prominent examples of LLMs include OpenAI's GPT series [24], Google's BERT [25], and more recent models like GPT-4. These models are pretrained on massive amounts of text using unsupervised or self supervised learning objectives, such as predicting masked tokens or the next word in a sequence. This extensive pretraining enables LLMs to learn rich contextual representations that can be finetuned for a wide range of downstream NLP tasks, including text classification, summarization, translation, and question answering.

One of the key applications of LLMs relevant to this thesis is synthetic text data generation. LLMs can generate high quality, contextually relevant text samples that augment existing datasets and help address issues such as data scarcity and class imbalance. For instance, GPT-based models have been successfully used to create synthetic training data for text classification tasks, improving model performance and generalization.

Recent studies have also explored the use of LLMs for data augmentation in lowresource settings, demonstrating that synthetic data generated by these models can rival or even surpass real data in effectiveness. However, challenges remain, including ensuring diversity in generated samples and mitigating potential biases inherited from the training data [26].

LLMs represent a powerful tool for both understanding and generating natural language, with synthetic data generation being a promising application for enhancing supervised learning tasks such as multiclass text classification.

2.6.1 How Large Language Models Work

The design of Large Language Models (LLMs) is based on the Transformer architecture, introduced in 2017, which has become the backbone of modern NLP systems [27]. The process begins with the ingestion of raw text, which is first tokenized, split into smaller units such as words or sub words, so they can be represented numerically and processed efficiently by the model. Each token is then mapped to a dense vector in a high dimensional space through an embedding layer, capturing semantic relationships between tokens and enabling the model to recognize linguistic patterns.

Once embedded, these token representations are passed into a series of stacked Transformer blocks, each composed of two primary sublayers: a multi head self-attention mechanism and a position-wise feed forward neural network [27]. The self-attention mechanism computes attention scores between all pairs of tokens in the sequence, capturing long-range dependencies and contextual relationships that help the model understand text.

This mechanism is mathematically formulated as scaled dot-product attention, where each token's representation is updated based on a weighted sum of all other tokens, with the weights determined by their contextual relevance. Multi-head attention extends this idea by allowing the model to attend to information from multiple representation subspaces simultaneously, providing a richer contextual representation.

To retain information about the order of tokens, since the Transformer's self-attention does not encode word order, positional encodings are added to the embeddings at the input of each block [27]. After the self-attention layer, the position-wise feed forward network applies nonlinear transformations independently to each token's representation, enhancing the model's ability to capture complex features. Both sublayers are wrapped with residual connections and followed by layer normalization, which stabilizes training and helps the flow of gradients in deep networks.

The outputs from the final Transformer block are then passed into a task specific output layer, such as a softmax classifier for language modeling or text classification. The entire model is trained end to end on large scale corpora using objectives like next token prediction or masked language modeling, with optimization performed via stochastic gradient descent and backpropagation. This training setup, combined with the architectural innovations of the Transformer, enables LLMs to learn context sensitive representations of language and achieve strong performance across many NLP tasks [27].

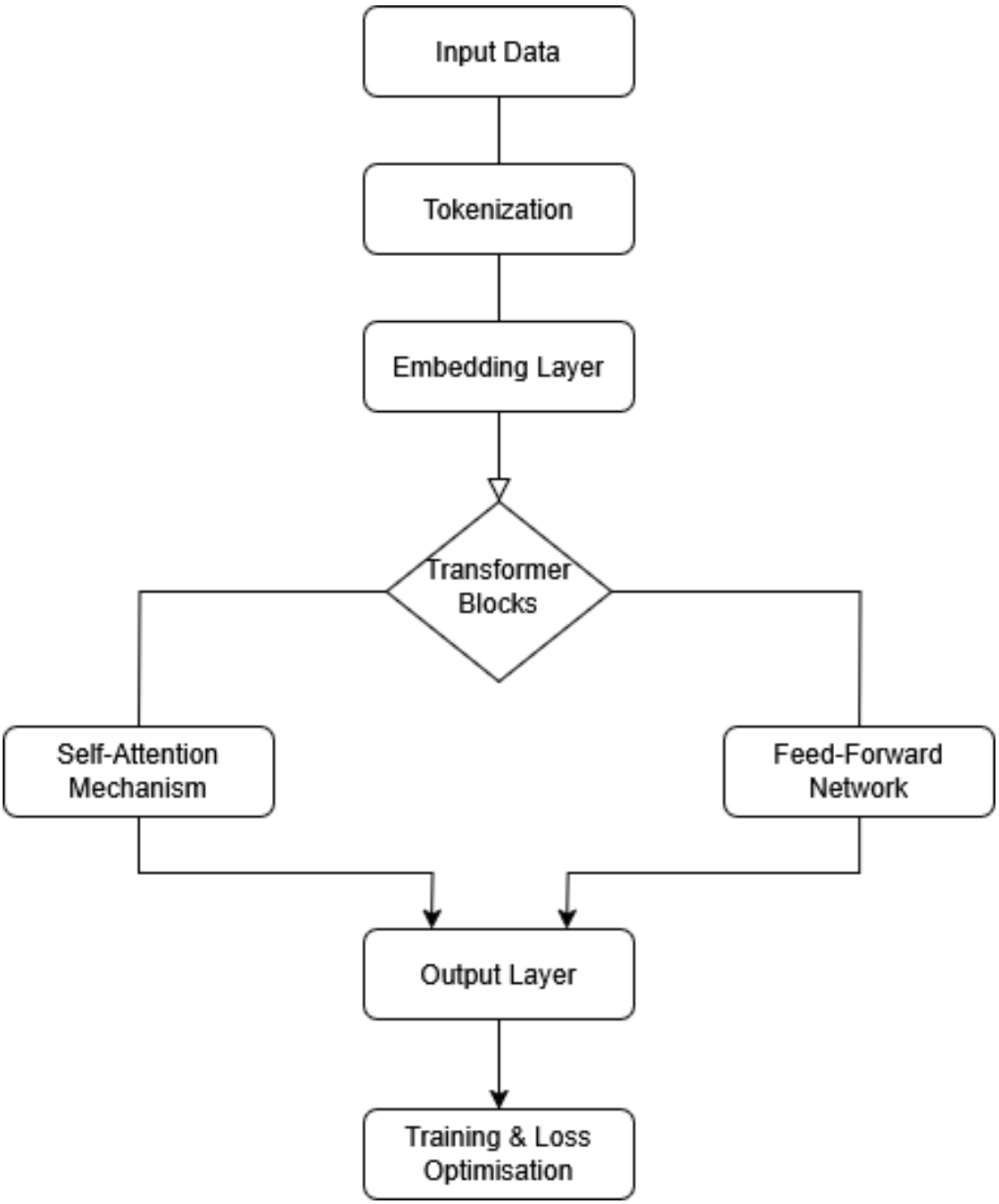

Figure 3: Transformer Model Architecture

2.7 Comparing Traditional Text Augmentation Techniques with LLM-Based Synthetic Data Generation

Large Language Model (LLM)-based synthetic data generation is a clear improvement over traditional text augmentation techniques such as synonym replacement, random insertion/deletion, back-translation, paraphrasing, and word embedding perturbation. Traditional methods primarily manipulate existing data at the token or sentence level, often resulting in limited diversity and sometimes introducing unnatural or semantically inconsistent variations. For example, synonym replacement and random insertion/deletion can help increase data volume but may not capture the range of linguistic variation needed for strong model training. Back-translation and paraphrasing offer more semantic diversity but are limited by the quality of translation systems and paraphrase models, and may not always give contextually appropriate results [28].

In contrast, LLM-based synthetic data generation uses the generative capabilities of models like GPT-3, GPT-4, and Gemini to create entirely new text samples that closely mimic the distribution and complexity of real-world data. LLMs can generate contextually rich and coherent examples tailored to specific tasks or underrepresented classes, which is especially beneficial in addressing class imbalance and data scarcity [28][29]. Studies have shown that integrating LLM-generated synthetic data can improve model performance, particularly for minority classes, and enhance generalization across diverse tasks and domains. For instance, in emotion classification, augmenting minority classes with LLM-generated samples led to clear improvements in classification accuracy and robustness, outperforming traditional balancing techniques like undersampling and oversampling [28].

LLMs can also create domain-specific synthetic datasets, as demonstrated in specialized fields such as legal document analysis, where curriculum learning and synthetic data generation with LLMs improved benchmark performance compared to traditional augmentation and fine-tuning approaches. LLM-generated data also shows greater adaptability and can be fine-tuned or prompted to produce samples with desired properties, making them more useful for machine learning applications.

2.8 Impact of Synthetic Data on Model Performance

The use of synthetic data in machine learning has gained attention as a strategy to improve model performance, especially in scenarios where real data is limited, imbalanced, or costly to obtain. Numerous studies have investigated how synthetic data influences accuracy, generalization, and robustness of models across various domains, including text classification.

Synthetic data can help mitigate the problem of data scarcity by augmenting training datasets, which helps models learn more representative features. For instance, it has been demonstrated that synthetic data generated via GANs can have improved classification accuracy in medical imaging by increasing the diversity of training samples [30]. Similarly, in natural language processing, synthetic text generated by language models has been shown to enhance classifier performance, particularly for underrepresented classes [31].

Comparative analyses between models trained on real, synthetic, and mixed datasets reveal that while models trained only on synthetic data often achieve reasonable baseline performance, combining synthetic with real data typically yields superior results. This is attributed to the complementary nature of synthetic data, which can fill gaps in the real data distribution and reduce overfitting [32].

However, the effectiveness of synthetic data depends on its quality and diversity. Poorly generated synthetic samples can introduce noise or bias, potentially degrading model performance [33]. Moreover, synthetic data may not fully capture complex real-world distributions, limiting its usefulness in some contexts.

Overall, synthetic data has a positive impact on model performance by enhancing data diversity and addressing imbalance, especially when used
in conjunction with real data. Ongoing research focuses on improving generation techniques to maximize the benefits of synthetic data while minimizing its limitations.

3 Methodology

3.1 Overview of Experiment

The goal of this experiment is to investigate the impact of synthetic text data, generated by a Large Language Model (LLM), on the performance of machine learning models in a multiclass text classification task.

To achieve this, three separate datasets were constructed and evaluated under identical conditions: one composed exclusively of real-world book descriptions, one generated entirely by the LLM, and a third combining equal parts of both.

Each dataset was preprocessed using the same pipeline and used to train identical model architectures. This way, any observed differences in performance can be attributed to the source and nature of the data. The models were evaluated using standard classification metrics, with a focus on accuracy and generalization across all seven classes.

This experimental setup allows for a direct comparison between real and synthetic data and enables an assessment of whether a combination of both can improve classification outcomes. By isolating the data type as the only variable, this design aims to provide a clear understanding of the benefits and limitations of incorporating LLM-generated synthetic data into supervised learning tasks.

3.2 Dataset Description

The dataset for this study comes from the Amazon Books Reviews dataset on Kaggle [34], which contains metadata and user reviews for a wide variety of books. The initial dataset consists of 143,962 book descriptions prior to any filtering or preprocessing. Each entry contains a textual description of a book, and each book is assigned to one of multiple distinct categories. These categories represent different genres or thematic classes, supporting a multiclass classification framework.

The distribution of book descriptions across categories containing more than 2500 entries is illustrated in Figure 4. As shown, the dataset contains many entries in several major categories, with Fiction being the most prevalent at over 22,000 entries. Other categories include Religion and History, each with around 10,000 entries, followed by Juvenile Fiction, Biography and Autobiography, Business and Economics, and Computers. Additional categories such as Social Science, Juvenile Nonfiction, Science, Education, Cooking, and Sports and Recreation are also well represented, each with more than 2,500 entries.

This distribution highlights a clear class imbalance within the dataset, where some genres, particularly Fiction, dominate, while others have much fewer samples. Such imbalance is typical of real-world book collections and presents challenges for supervised classification, as models may become biased toward majority classes.

To address these challenges and ensure a focused multiclass classification task, seven distinct categories were selected from among the most populated and thematically clear classes in the dataset. The selection process aimed to balance the number of samples per class as much as possible while maintaining clear distinctions between genres. By narrowing the scope to these seven categories, the study keeps both practical relevance and manageable complexity for evaluating the impact of synthetic data augmentation on classification performance.

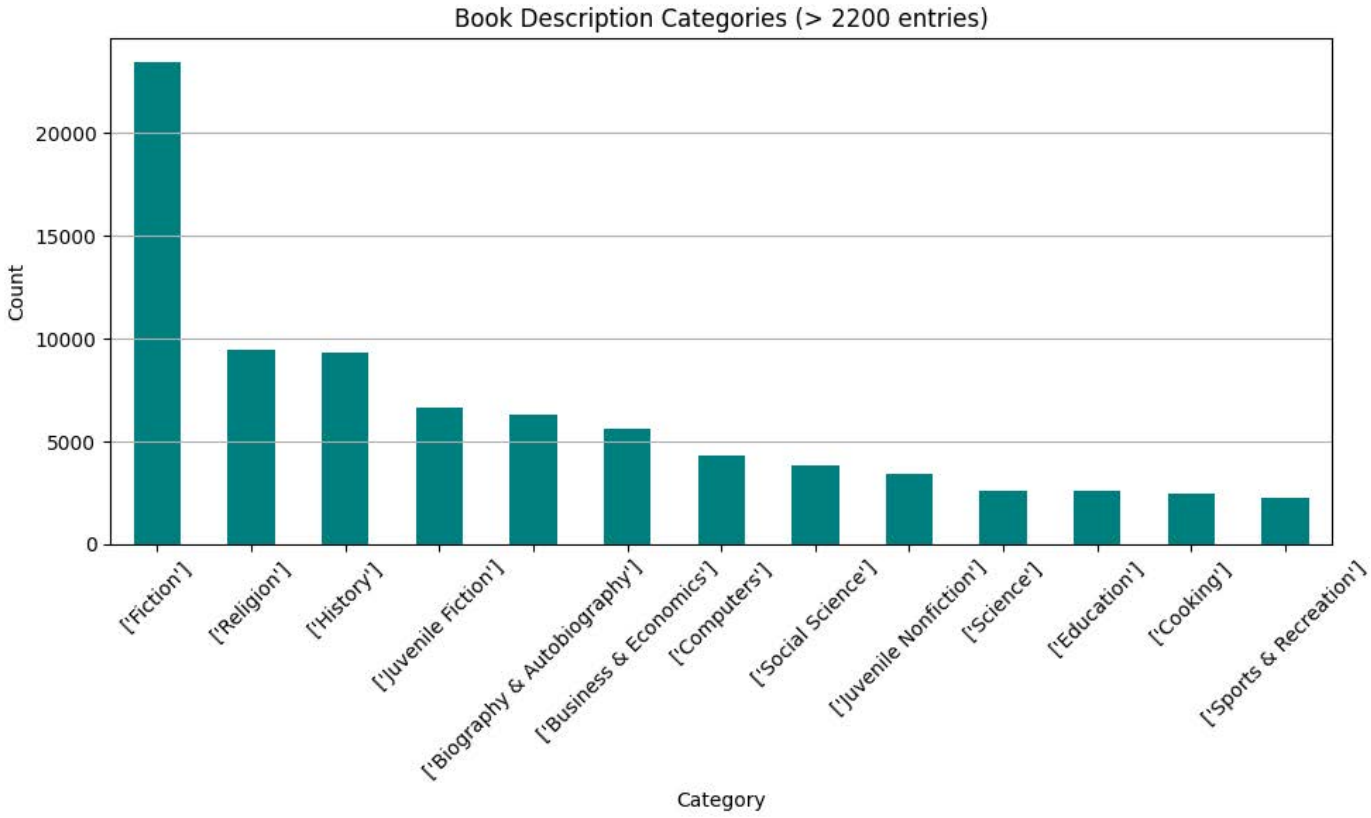

Figure 4: Categories containing more than 2500 entries

3.3 Dataset Preprocessing

To prepare the Amazon Books Reviews dataset for the multiclass text classification task, a preprocessing pipeline was implemented to ensure data quality and consistency, since the original dataset often contained multiple categories per book with overlapping labels.

Entries with missing category information were removed to keep the dataset consistent. The categories column was converted to string type to facilitate pattern matching. A custom function, **assign_category**, was applied to each entry's category list to assign a single category based on the presence of any of the mapped substrings using case-insensitive regular expression matching. Entries that did not match the seven predefined categories were removed.

Subsequently, entries missing either the book description or title were dropped to keep the textual data complete for classification. The final filtered dataset retained only the columns relevant to the task: description, categories, and Title.

This preprocessing resulted in a cleaner, more focused dataset suitable for training and evaluation. The distribution of samples across the seven selected categories after preprocessing is visualized in Figure 5, which displays a bar chart of category counts. This figure highlights the remaining class imbalance, with some categories having far more samples than others, reinforcing the need for augmentation strategies in subsequent experiments.

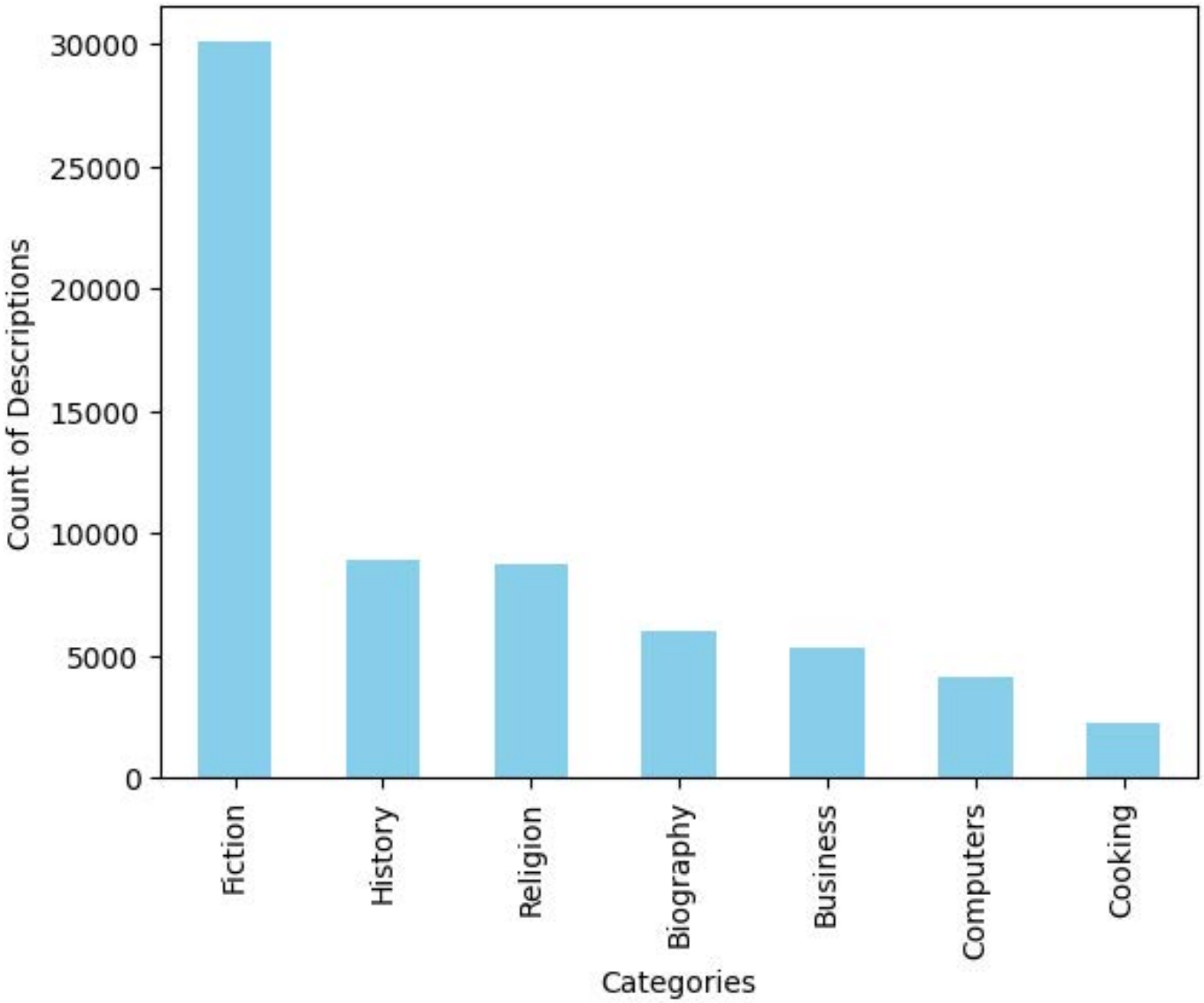

Figure 5: Categories after data cleaning

3.4 Construction of Synthetic Dataset

The synthetic book descriptions were generated using the GPT-4 Large Language Model. The generation process involved iterating over each entry in the filtered real dataset, which contains book titles and their assigned categories.

For each book, a prompt was constructed to instruct GPT-4 to generate a unique and original synopsis. The

prompt explicitly specified the book's title and its genre category to guide the model in producing a relevant description. The prompt used was:

> "Generate a captivating and original book synopsis for a book belonging in the '{category}' genre titled '{book_title}'. Ensure the content is unique and does not include any information or references from existing works with the same title. The synopsis should introduce compelling topics, a fascinating narrative, and a vivid setting, making it engaging and appealing to potential readers. Emphasize originality and creativity in crafting a narrative that fits the title and genre, ensuring the synopsis highlights how the book exemplifies its genre."

This prompt was specifically designed to prevent the model from reproducing actual existing descriptions associated with the book title. Since GPT-4 sometimes recognizes well known book titles and generates their real descriptions from its training data, the prompt explicitly instructs the model to avoid including any information or references from existing works. This ensures that all synthetic descriptions are entirely original and do not replicate real book content.

The synthetic dataset was generated by producing one synthetic description for each real book entry in the filtered dataset. This resulted in a synthetic dataset with the same number of samples and class distribution as the real dataset, enabling direct comparison between models trained on real, synthetic, and mixed datasets. The number of the synthetic samples per class equal those of the original dataset as they were created one to one.

3.5 Text Preprocessing and Feature Engineering

3.5.1 Text Cleaning and Normalization

The text cleaning and normalization process was applied to both the real book descriptions and the synthetic descriptions to prepare the dataset for analysis. Initially, all text entries were converted to lowercase to keep the text uniform and reduce the vocabulary size. Following this, special characters such as line breaks and possessive suffixes were removed, and only alphabetic characters and whitespace were retained. Numbers, punctuation, and other symbols were removed to avoid noise.

Tokenization was then performed using the **word_tokenize** function from the NLTK library, which split the text into individual words. To further refine the text, common English stop words were removed using NLTK's predefined list, keeping the dataset focused on terms more relevant to classification.

In addition, a frequency-based filtering step was applied to exclude words that appeared too rarely or too frequently. Specifically, words occurring fewer than two times across the entire dataset were removed to eliminate rare terms that were unlikely to contribute to the model. Conversely, words appearing in more than 80% of the documents were also filtered out to avoid overly common terms that could weaken the features.

Finally, the Porter stemming algorithm was applied to reduce words to their root forms, consolidating different morphological variants of a word into a single representation. This preprocessing pipeline was wrapped in a custom function named **cleanDescriptions**, which was applied to the description and responses columns. The resulting cleaned and normalized text formed the basis for subsequent tokenization and feature extraction steps.

```
def cleanDescriptions(df, column_name,ps,stop_words,min_freq=2, max_freq
   =0.8):
    """Preprocess a text column in a DataFrame."""
    df[column_name] = df[column_name].astype(str)
    df[column_name] = df[column_name].apply(normalize_text)
    df[column_name] = df[column_name].apply(lambda x: x.lower())
    df[column_name] = df[column_name].apply(lambda x: x.replace('\r\n',
       ''))
    df[column_name] = df[column_name].apply(lambda x: x.replace("'s", ''
       ))
    df[column_name] = df[column_name].apply(lambda x: ''.join(c for c in
        x if c.isalpha() or c.isspace()))
    df[column_name] = df[column_name].apply(word_tokenize)
    df[column_name] = df[column_name].apply(lambda x: [word for word in
      x if word not in string.punctuation])
    df[column_name] = df[column_name].apply(lambda x: [word for word in
```

```
        x if word not in stop_words])
    all_words = [word for tokens in df[column_name] for word in tokens]
    word_counts = Counter(all_words)
    total_docs = len(df)

    # Filter based on word frequencies
    def filter_by_frequency(tokens):
        return [
            word for word in tokens
            if min_freq <= word_counts[word] <= max_freq * total_docs
        ]

    df[column_name] = df[column_name].apply(filter_by_frequency)
    df[column_name] = df[column_name].apply(lambda x: ' '.join(ps.stem(
        word) for word in x))
    return df
```

Listing 1: Clean Descriptions Function

3.5.2 Creation of the Mixed Dataset

To address class imbalance in the real dataset, a mixed dataset was created by augmenting underrepresented categories with synthetic descriptions generated by the LLM.

Starting from the filtered real dataset containing book descriptions (description column) and their assigned categories, the distribution of samples per category was analyzed. Categories with fewer than 5,000 entries were identified as needing augmentation.

For each such category, additional synthetic samples were drawn by randomly sampling with replacement from the synthetic descriptions stored in the responses column of the original dataset. These synthetic samples were then renamed to align with the description field and appended to the real dataset. This process increased the number of samples in each underrepresented category to at least 5,000.

As a result, the mixed dataset contains a combination of real book descriptions and synthetic descriptions, with the synthetic data specifically used to balance the classes. This dataset enables evaluation of classification models trained on a blend of real and synthetic data, providing insight into the effectiveness of synthetic data augmentation for addressing class imbalance.

```
mixed_df = df[['description', 'categories']].copy()

# Check which categories have less than 5000 entries
category_counts = df['categories'].value_counts()
categories_to_balance = category_counts[category_counts < 5000]

# Fill up categories to reach 5000 entries in `mixed_df`
for category, count in categories_to_balance.items():
    # Calculate the number of entries needed
    needed = 5000 - count

    # Extract additional rows from the original dataset (`responses` and
        `categories`)
    additional_rows = df[df['categories'] == category][['responses', '
       categories']].sample(needed, replace=True)
    additional_rows.rename(columns={'responses': 'description'}, inplace
       =True)

    # Append these rows to the `mixed_df`
    mixed_df = pd.concat([mixed_df, additional_rows], ignore_index=True)
```

Listing 2: Creation of the Mixed Dataset

3.5.3 Tokenization and Word Embedding

The tokenization process began by creating a unified vocabulary that encompassed all textual data from the real book descriptions, synthetic descriptions, and the mixed dataset. This was achieved by combining the text from the description and responses columns of the original dataset along with the description column of the mixed dataset. The purpose of this step was to ensure that the tokenizer captured the full range of words

appearing across all datasets, keeping word-to-index mappings consistent during training and evaluation.

Using Keras' Tokenizer class, each unique word was assigned a unique integer index, with more frequent words receiving lower indices. This frequency-based indexing makes representation efficient and gives priority to common terms during training. Subsequently, each text entry was converted into a sequence of these integer indices, representing the tokenized form of the text.

Since neural network models require inputs of uniform length, the tokenized sequences were padded to match the length of the longest sequence in the dataset. Padding ensures that shorter sequences are extended with zeros, so batch processing works without losing information or creating bias from variable-length inputs.

To convert these tokenized sequences into numerical features with semantic meaning, pre-trained GloVe embeddings were loaded. The GloVe embeddings were read from a text file containing word vectors, where each line corresponds to a word followed by its embedding values. The loading function iterated through each line, splitting it into the word and its associated vector, which was then stored in a dictionary mapping words to their respective embedding vectors. This dictionary, known as the embeddings index, serves as a lookup table for retrieving vector representations during embedding matrix construction.

An embedding matrix was then created with dimensions corresponding to the size of the tokenizer's vocabulary plus one (to account for padding) and the embedding dimensions. For each word in the tokenizer's vocabulary, the corresponding GloVe vector was retrieved from the embeddings index and placed in the embedding matrix at the row matching the word's index. Words not found in the GloVe embeddings were assigned zero vectors, which treated them as out-of-vocabulary words. This embedding matrix enables the model to map token indices to dense vector representations that capture semantic relationships.

By integrating tokenization with pre-trained embeddings, the model benefits from both consistent text representation across datasets and semantic information encoded in the GloVe vectors, helping the model better understand and classify book descriptions.

The use of pre-trained GloVe embeddings uses semantic relationships learned from large bodies of text, helping the model understand word meanings and context [35].

```
def createVocabulary(df, mixed_df):
all_descriptions_responses = df['description'].tolist() + df['
    responses'].tolist() + mixed_df['description'].tolist()
```

```
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(all_descriptions_responses)

    total_words = len(tokenizer.word_index)

    return tokenizer, total_words
```

Listing 3: Vocabulary creation function

```
def tokenizeDesc(descriptions,tokenizer):
    sequences = tokenizer.texts_to_sequences(descriptions)

    # Find the max sequence length and pad sequences to this length
    max_sequence_length = max(len(seq) for seq in sequences)
    padded_sequences = pad_sequences(sequences, maxlen=
       max_sequence_length)

    return padded_sequences
```

Listing 4: Tokenization function

3.6 Model Architecture

The text classification model used in this study is based on a bidirectional Long Short-Term Memory (BiLSTM) architecture, a specialized form of recurrent neural network (RNN) designed to capture contextual information from sequences in both forward and backward directions. This bidirectional processing enables the model to better understand the semantic dependencies and word order within book descriptions, which is crucial for accurate classification. BiLSTM architectures have been widely adopted in recent text classification research due to their effectiveness in capturing long-range dependencies and contextual nuances in text data.

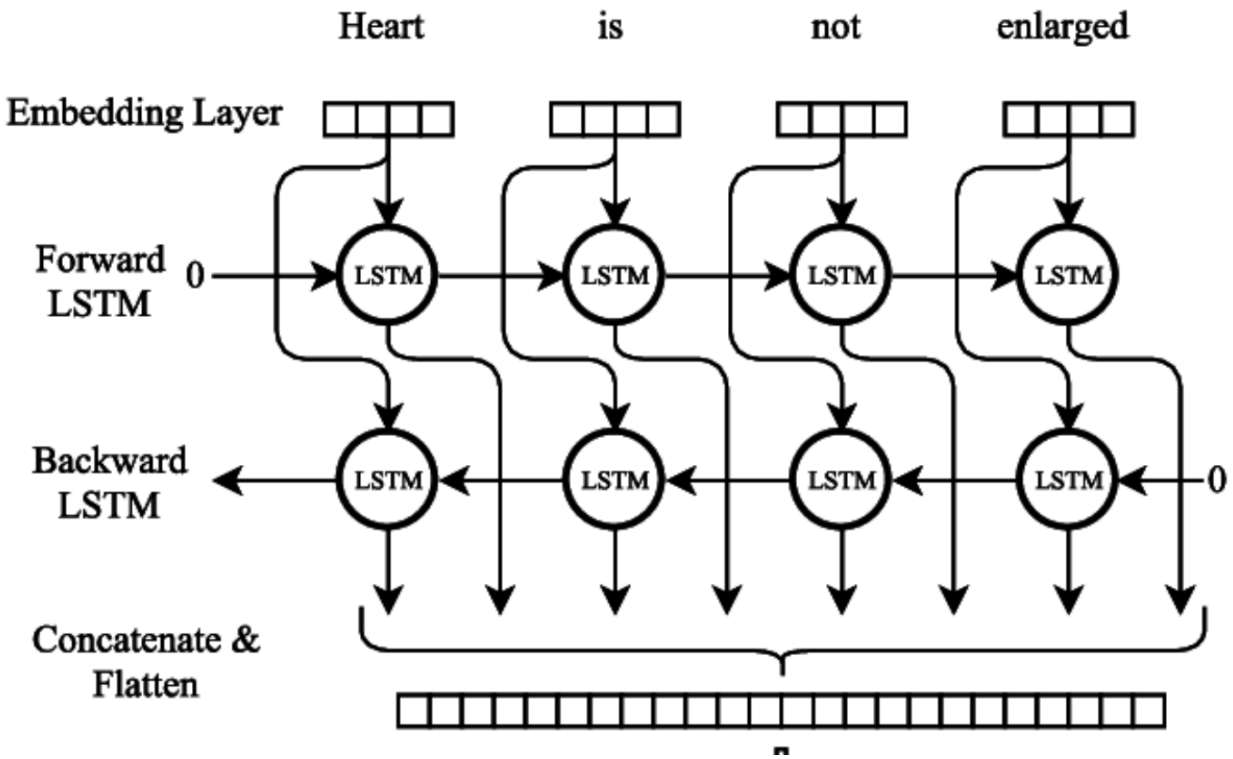

Figure 6: Structure of BiLSTM

The model begins with an embedding layer initialized with pre-trained GloVe vectors, which provide dense, semantically rich representations of words based on large-scale text. These embeddings are trainable, allowing the model to fine-tune them for the specific book classification task. Following the embedding layer, the network includes two stacked bidirectional LSTM layers: the first with 128 units returning sequences, and the second with 64 units producing a fixed-length output. Both layers incorporate L2 regularization to reduce overfitting, a common challenge in deep learning models

Dropout layers with a rate of 0.3 are applied after each BiLSTM layer to further mitigate overfitting by randomly deactivating neurons during training. Additionally, layer normalization is employed after the first dropout to stabilize and accelerate training convergence. The final dense layer uses a softmax activation function to output probabilities across the seven book categories.

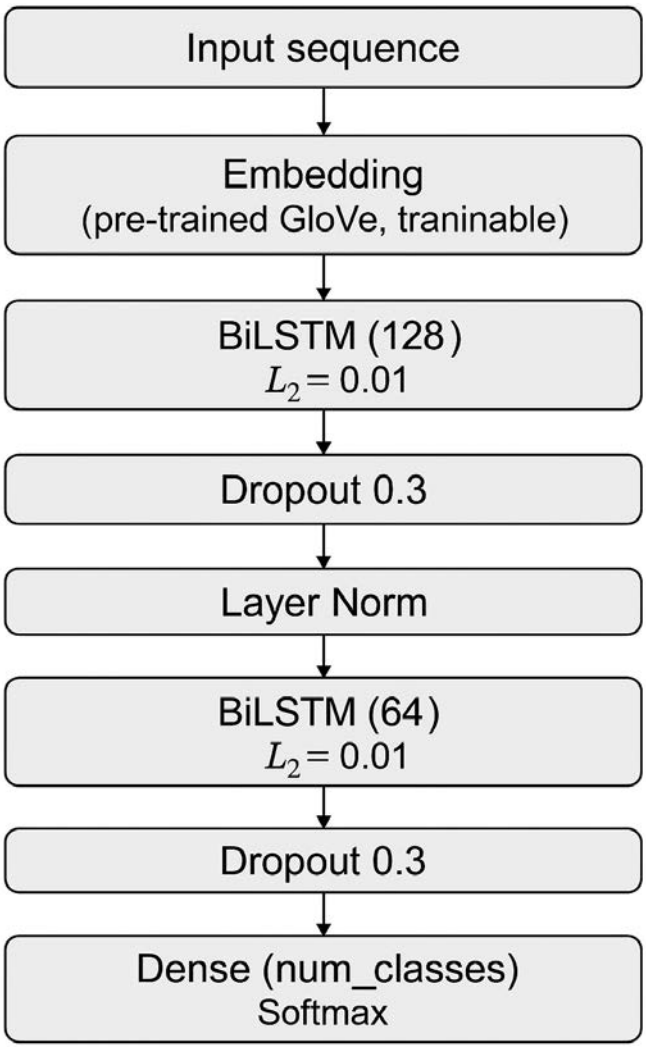

Figure 7: Model Architecture

3.7 Data Splitting and Label Encoding

Prior to model training, the dataset was systematically partitioned into training, validation, and test subsets to ensure robust evaluation and prevent data leakage. The splitting procedure was implemented using a custom function that leveraged stratified random sampling to maintain representative class distributions across subsets. Specifically, 70% of the data was allocated for training, while the remaining 30% was equally divided between validation and test sets, resulting in 15% for each.

The splitting process involved two sequential steps: initially, the dataset was divided into a training set

and a temporary set (comprising validation and test samples). Subsequently, the temporary set was split evenly to form the validation and test sets. This approach ensured balanced and randomized partitions, controlled by a fixed random seed for reproducibility.

Following the data split, label encoding was performed to transform categorical class labels into a numerical format suitable for model training. A label encoder was first applied to convert string labels into integer representations. These integer labels were then transformed into one-hot encoded vectors, which are required by the categorical cross-entropy loss function used during training. This encoding was consistently applied across training, validation, and test labels.

```
def split_data(real_padded_sequences, labels, test_size=0.3,
    random_state=42):
"""Split data into training, validation, and test sets."""
X_train, X_temp, y_train, y_temp = train_test_split(
    real_padded_sequences, labels, test_size=test_size, random_state
        =random_state
)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=random_state
)

return X_train, X_val, X_test, y_train, y_val, y_test
```

Listing 5: Splitting Data function

```
def encode_labels(y_train, y_val, y_test):
    """Encode labels using label encoding and one-hot encoding."""
    encoder = LabelEncoder()

    # Perform label encoding
    y_train_encoded = encoder.fit_transform(y_train)
    y_val_encoded = encoder.transform(y_val)
    y_test_encoded = encoder.transform(y_test)

    # Convert labels to one-hot encoding
    y_train_onehot = to_categorical(y_train_encoded, num_classes=len(
        encoder.classes_))
    y_val_onehot = to_categorical(y_val_encoded, num_classes=len(encoder
        .classes_))
    y_test_onehot = to_categorical(y_test_encoded, num_classes=len(
        encoder.classes_))

    return y_train_onehot, y_val_onehot, y_test_onehot, encoder
```

Listing 6: Encoding Labels function

3.8 Training Procedure and Hyperparameters

The model training process was conducted using the TensorFlow deep learning framework, specifically leveraging the Keras API for model construction and training. The Adam optimizer was adopted due to its adaptive learning rate mechanism and consistent performance across a range of classification tasks. The learning rate remained at the default setting provided by the Adam implementation, as no custom learning rate schedule was introduced during the experiments.

Categorical cross-entropy was selected as the loss function, which is well-suited for multi-class classi-

fication problems involving one-hot encoded target labels. The training was configured for a maximum of 15 epochs, with a batch size of 128 samples per iteration. To mitigate the risk of overfitting and to enhance generalization, an early stopping strategy was employed. This involved monitoring the validation loss and terminating training if no improvement was observed over four consecutive epochs, with the best performing model weights restored upon completion.

The dataset was partitioned into training, validation, and test sets using a custom data splitting function. The validation set served exclusively for monitoring model performance during training, while the test set was reserved for the final evaluation phase, adhering to the standard train-validation-test split methodology. All experiments were executed on Google Colab, utilizing a T4 GPU.

```
    # Define early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=4,
   restore_best_weights=True)

# Train the model
epochs = 15
batch_size = 128

history_real = model_realData.fit(X_train_real, y_train_onehot_real,
   validation_data=(X_val_real, y_val_onehot_real), epochs=epochs,
   batch_size=batch_size, callbacks=[early_stopping])
```

Listing 7: Training the Model

3.9 Evaluation Metrics

The evaluation of the model's performance was conducted by calculating accuracy and assessing generalization using the test dataset. Accuracy was determined by comparing the predicted class labels generated by the model against the true labels of the test samples. Specifically, the model outputs probability distributions over the classes for each input, which were then converted into discrete predictions by selecting the class with the highest probability. The true labels, originally one-hot encoded, were decoded back to their categorical form to enable direct comparison. Accuracy was computed as the proportion of correct predictions relative to the total number of test samples, providing a clear and interpretable measure of overall classification performance.

To complement this metric, the training and validation loss values recorded during the training process were visualized to monitor the model's learning dynamics. Plotting these loss curves across epochs allowed for the detection of overfitting or underfitting behaviors, as a divergence between training and validation loss typically indicates poor generalization [36]. The model's generalization capability was further evaluated by applying it to the held-out test set, which was not used during training or validation phases. Performance on

this unseen data reflects the model's ability to apply learned patterns beyond the training examples.

Additionally, a confusion matrix was generated to provide a detailed view of the model's classification performance across all classes. This matrix highlights the distribution of true versus predicted labels, identifying specific classes where the model excels or struggles. Alongside the confusion matrix, a classification report was produced, summarizing precision, recall, and F1-score for each class, thereby offering a nuanced understanding of the model's strengths and weaknesses beyond overall accuracy.

```
def evaluate_model_performance(true_genres, predicted_genres,
    encoder):
"""Display confusion matrix and classification report."""
from sklearn.metrics import confusion_matrix

confusion_mtx = confusion_matrix(true_genres, predicted_genres)

plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx, annot=True, fmt='d', cmap='Blues', cbar=
    False)
plt.xlabel('Predicted Genres')
plt.ylabel('True Genres')
plt.title('Confusion Matrix')
plt.xticks(ticks=np.arange(len(encoder.classes_)), labels=encoder.
    classes_, rotation=90)
plt.yticks(ticks=np.arange(len(encoder.classes_)), labels=encoder.
    classes_, rotation=0)
plt.show()

classification_rep = classification_report(true_genres,
    predicted_genres, target_names=encoder.classes_)
print("Classification Report:")
```

```
    print(classification_rep)
```

Listing 8: Function for generating confusion matrix and classification report

4 Results

4.1 Real Dataset

4.1.1 Training

The model was trained on the real dataset for 10 epochs, with performance monitored at each epoch on the training and validation sets. The training process showed a steady decrease in loss values, along with improvements in accuracy, as shown in Table 1 and Figure 8.

At epoch 1, training accuracy was 51.91% with a high loss of 9.2805, while validation accuracy was higher at 74.38% with a loss of 2.4569. Over the course of training, the training accuracy increased steadily, reaching 95.82% with a loss of 0.2828 by epoch 10. Validation accuracy peaked at 81.02% during epochs 5 and 6, with corresponding validation losses of 0.8460 and 0.8174. This shows learning while maintaining generalization to validation data. After epoch 6, validation accuracy showed slight fluctuations and a mild decrease, while validation loss moved between 0.81 and 0.91, suggesting the start of overfitting, which is common in deep learning models.

Epoch	Accuracy	Loss	Val Accuracy	Val Loss
1	0.5191	9.2805	0.7438	2.4569
2	0.7735	1.9864	0.7869	1.2980
3	0.8310	1.1062	0.8000	1.0171
4	0.8560	0.8270	0.7737	1.0111
5	0.8848	0.6605	0.8102	0.8460
6	0.9129	0.5077	0.8102	0.8174
7	0.9286	0.4238	0.7981	0.8418
8	0.9384	0.3731	0.8085	0.8236
9	0.9449	0.3302	0.8039	0.8440
10	0.9582	0.2888	0.7903	0.9114

Table 1: Training and validation performance of real data.

The loss curves in Figure 8 confirm these observations, revealing a downward trend in training loss throughout the epochs and a validation loss curve that stabilizes but does not significantly decrease past epoch 6. Such patterns often imply that the model is capturing increasingly complex data patterns in training, while its capacity to generalize may plateau or degrade slightly.

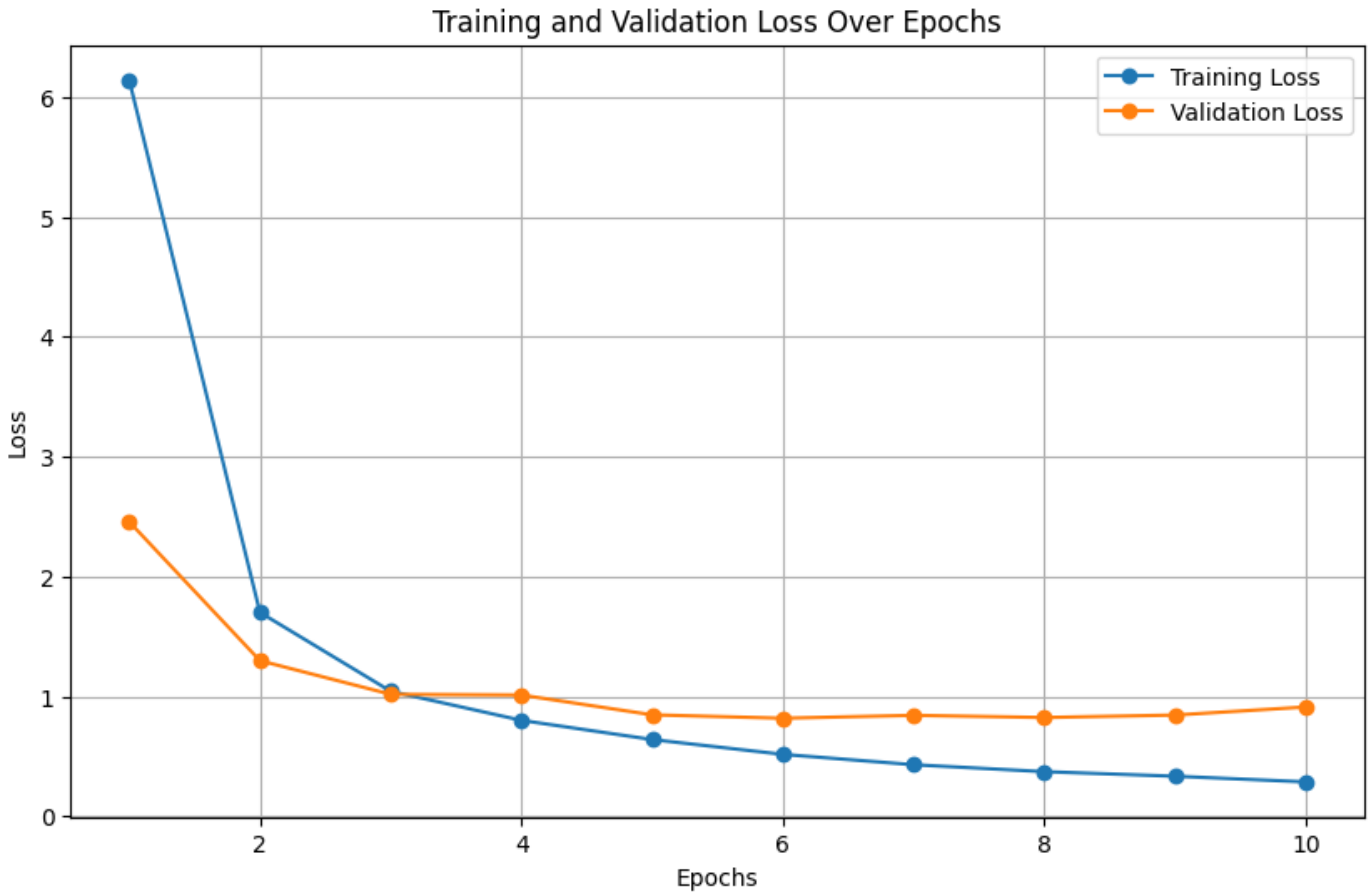

Figure 8: Training and Validation loss curves of real data

4.1.2 Predictions

The confusion matrix (Figure 9) shows classification outcomes on the test set across seven genres. The model shows high true positive rates in categories such as Computers and Cooking, while some confusion persists between related genres like Biography and History. This is also reflected in the per-class metrics summarized in the classification report (Table 2), where precision and recall range from moderate to high. For instance, Cooking achieves the highest F1-score of 0.94, while History has a lower precision of 0.62 but a higher recall of 0.82.

Overall, these results show that the model trained on real data reaches about 80% overall accuracy and learns features that distinguish genres well. The gradual convergence of training and validation losses, along-

side consistent classification metrics, suggest the model generalizes well, with minor signs of overfitting in later epochs.

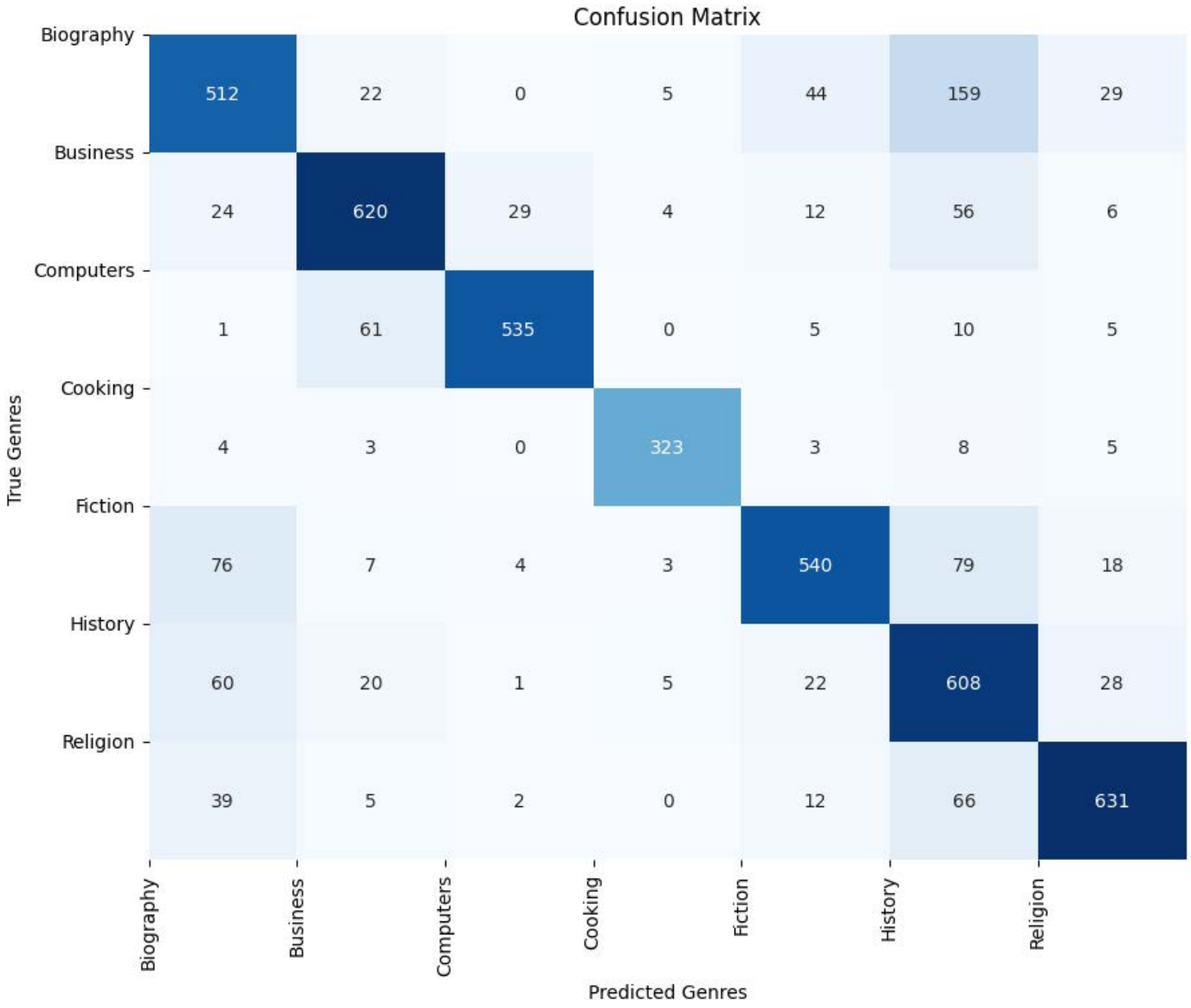

Figure 9: Confusion matrix of real data predictions

Class	Precision	Recall	F1-score	Support
Biography	0.72	0.66	0.69	771
Business	0.84	0.83	0.83	751
Computers	0.94	0.87	0.90	617
Cooking	0.95	0.93	0.94	346
Fiction	0.85	0.74	0.79	727
History	0.62	0.82	0.70	744
Religion	0.87	0.84	0.85	755
Accuracy			**0.80**	4711
Macro Avg	0.83	0.81	0.82	4711
Weighted Avg	0.81	0.80	0.80	4711

Table 2: Classification report of real data predictions

4.2 Synthetic Dataset

4.2.1 Training

The model was trained on the synthetic dataset for 15 epochs, and its performance was recorded on the training and validation sets at each epoch. The epoch-wise training metrics are presented in Table 3, and the corresponding loss curves are shown in Figure 9.

At epoch 1, training accuracy was 52.36% with a loss of 9.3171. Validation accuracy started at 85.97% with a loss of 2.2912. By the next epoch, results improved quickly. At epoch 2, training accuracy increased to 85.68% and the training loss dropped to 1.8998, while validation accuracy reached 89.2% and validation loss decreased to 1.1413.

Training accuracy continued to rise, reaching 98.07% by epoch 15, with the corresponding training loss dropping to 0.1687. Validation accuracy stayed high throughout, peaking at 93.10% (epoch 14), and validation loss declined steadily to 0.4011 at the final epoch.

Across all epochs, both training and validation losses showed a steady downward trend. Training accuracy improved with each epoch, while validation accuracy had minor fluctuations but stayed above 90% from epoch 3 onward. The complete statistics for each epoch are on the table below:

Epoch	Accuracy	Loss	Val Accuracy	Val Loss
1	0.5236	9.3171	0.8597	2.2912
2	0.8568	1.8998	0.8922	1.1413
3	0.8998	1.0448	0.9091	0.8475
4	0.9227	0.7726	0.9162	0.6796
5	0.9346	0.6075	0.9251	0.6106
6	0.9436	0.5146	0.9263	0.5187
7	0.9523	0.4179	0.9255	0.4721
8	0.9583	0.3547	0.9191	0.4651
9	0.9622	0.3219	0.9096	0.4801
10	0.9667	0.2772	0.9132	0.4694
11	0.9656	0.2738	0.9274	0.4107
12	0.9761	0.2211	0.9242	0.4202
13	0.9721	0.2201	0.9310	0.3872
14	0.9845	0.1665	0.9242	0.3994
15	0.9807	0.1687	0.9259	0.4011

Table 3: Training and validation performance of synthetic data.

The loss curves in Figure 10 illustrate the decline in both training and validation loss throughout the epochs, with validation loss maintaining a steady decrease alongside increasing stability in accuracy results.

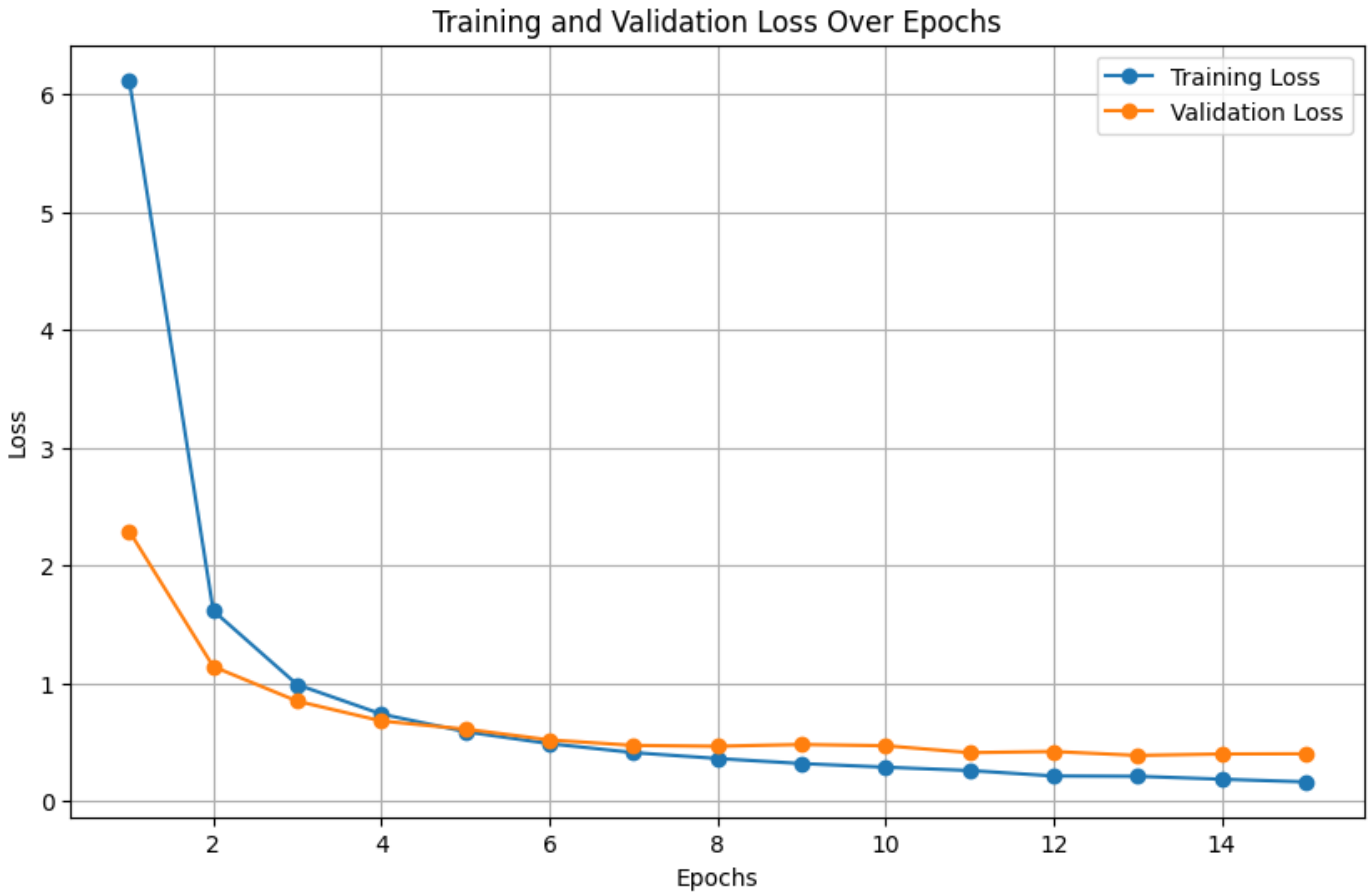

Figure 10: Training and Validation loss curves of synthetic data

4.2.2 Predictions on Synthetic Dataset

Following training, the model's predictive performance was evaluated on the synthetic test set. Outcomes are summarized in the confusion matrix (Figure 11) and the corresponding classification report (Table 4).

The confusion matrix details the distribution of true versus predicted genres on the synthetic test data. The model achieved high correct classification counts across all seven classes. For example, the Computers genre recorded 556 true positives, while Biography and Business attained 688 and 692 correct predictions, respectively. Misclassifications were generally low; for instance, the Computers class had 54 samples misclassified as Business and only 2 as Religion. Genres such as Cooking and Religion demonstrated very few off-diagonal misclassifications, indicative of precise predictions.

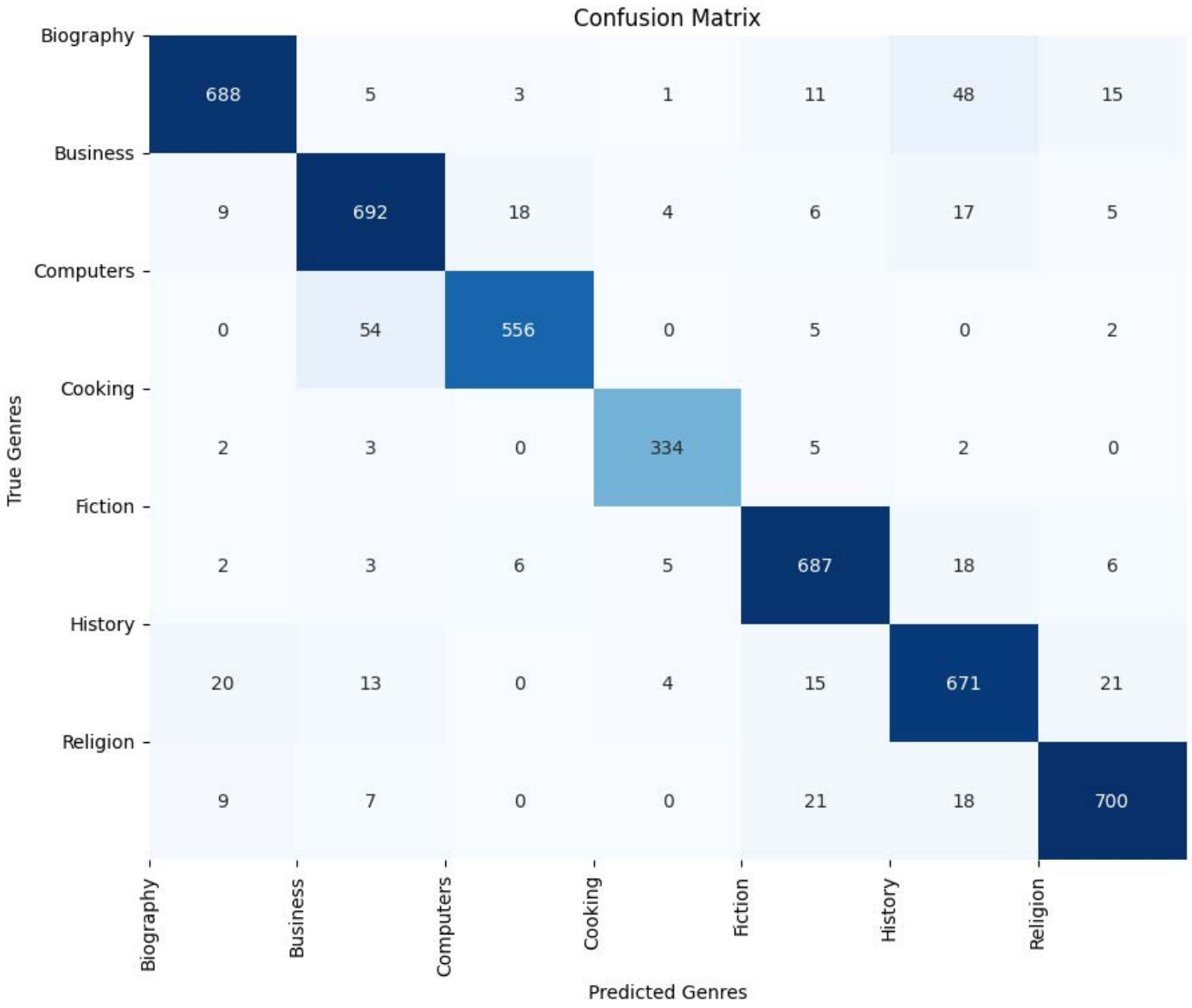

Figure 11: Confusion matrix of synthetic data predictions on synthetic data

Class	Precision	Recall	F1-score	Support
Biography	0.94	0.89	0.92	771
Business	0.89	0.92	0.91	751
Computers	0.95	0.90	0.93	617
Cooking	0.96	0.97	0.96	346
Fiction	0.92	0.94	0.93	727
History	0.87	0.90	0.88	744
Religion	0.93	0.93	0.93	755
Accuracy	-	-	0.92	4711
Macro Avg	0.92	0.92	0.92	4711
Weighted Avg	0.92	0.92	0.92	4711

Table 4: Classification report on the synthetic data set prediction.

4.2.3 Predictions on Real Dataset

To assess cross domain generalization, the model trained exclusively on synthetic data was evaluated on the held-out real test dataset. The resulting predictive performance is summarized in the confusion matrix (Figure 12) and the classification report (Table 5).

The classification report (Table 5) presents precision, recall, F1-score, and support for each genre as well as macro and weighted averages. Precision values ranged from 0.50 (History) to 0.79 (Religion), while recall values varied between 0.35 (Biography) and 0.92 (Cooking). The highest F1-score was achieved in the Computers (0.81) and Cooking (0.77) genres, while the lowest was observed for Biography (0.47) and History (0.57). The overall accuracy on the real test data was measured at 0.67. Both macro and weighted averages for precision, recall, and F1-score were approximately 0.68, 0.69, and 0.66, respectively, indicating moderate generalization of the synthetic-trained model to real-world data.

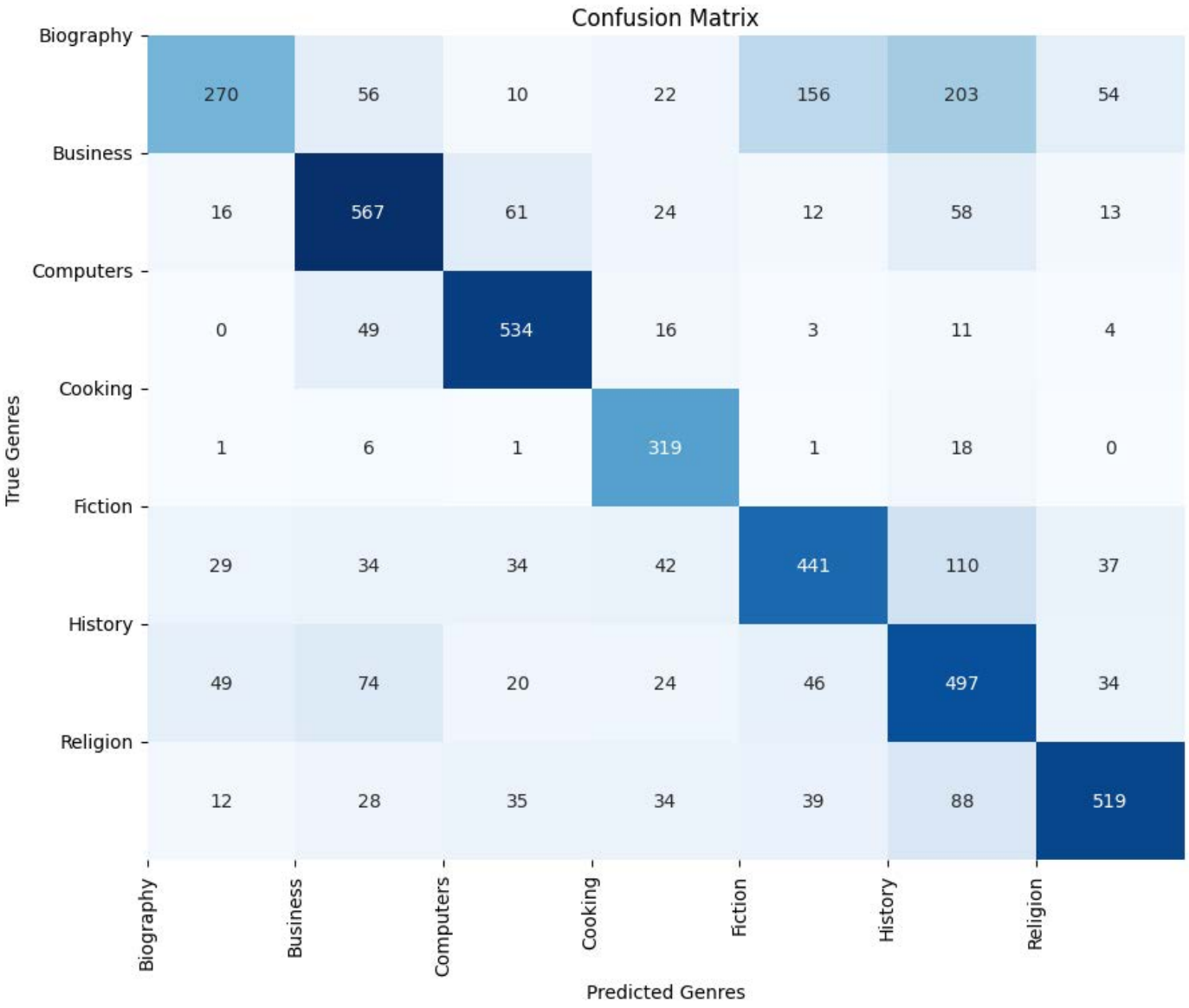

Figure 12: Confusion matrix of synthetic data predictions on real data

Class	Precision	Recall	F1-score	Support
Biography	0.72	0.35	0.47	771
Business	0.70	0.75	0.72	751
Computers	0.77	0.87	0.81	617
Cooking	0.66	0.92	0.77	346
Fiction	0.63	0.61	0.62	727
History	0.50	0.67	0.57	744
Religion	0.79	0.69	0.73	755
Accuracy	-	-	0.67	4711
Macro Avg	0.68	0.69	0.67	4711
Weighted Avg	0.68	0.67	0.66	4711

Table 5: Classification report of synthetic model on real dataset.

4.3 Mixed Dataset

4.3.1 Training

The model was trained on the mixed dataset, combining real data with synthetic samples to address class imbalance, for 15 epochs. Training and validation performance metrics were recorded for each epoch and are recorded in Table 6. The associated training and validation loss curves are visualized in Figure 13.

At epoch 1, training accuracy was 53.64% with a loss of 9.0114, while validation accuracy reached 78.19% and validation loss was 2.1442. Substantial gains in accuracy and loss reduction followed in subsequent epochs. By epoch 5, training accuracy had risen to 89.74% with a loss of 0.5956; validation accuracy simultaneously reached 82.76% and validation loss fell to 0.7425.

The highest validation accuracy (83.33%) occurred at epoch 6, while validation loss was minimized to 0.7308. Training accuracy continued to climb, reaching 97.79% with a loss of 0.1580 at epoch 15. Thereafter, validation accuracy remained in the range of 81–83%, and validation loss values stabilized between 0.73 and

0.88.

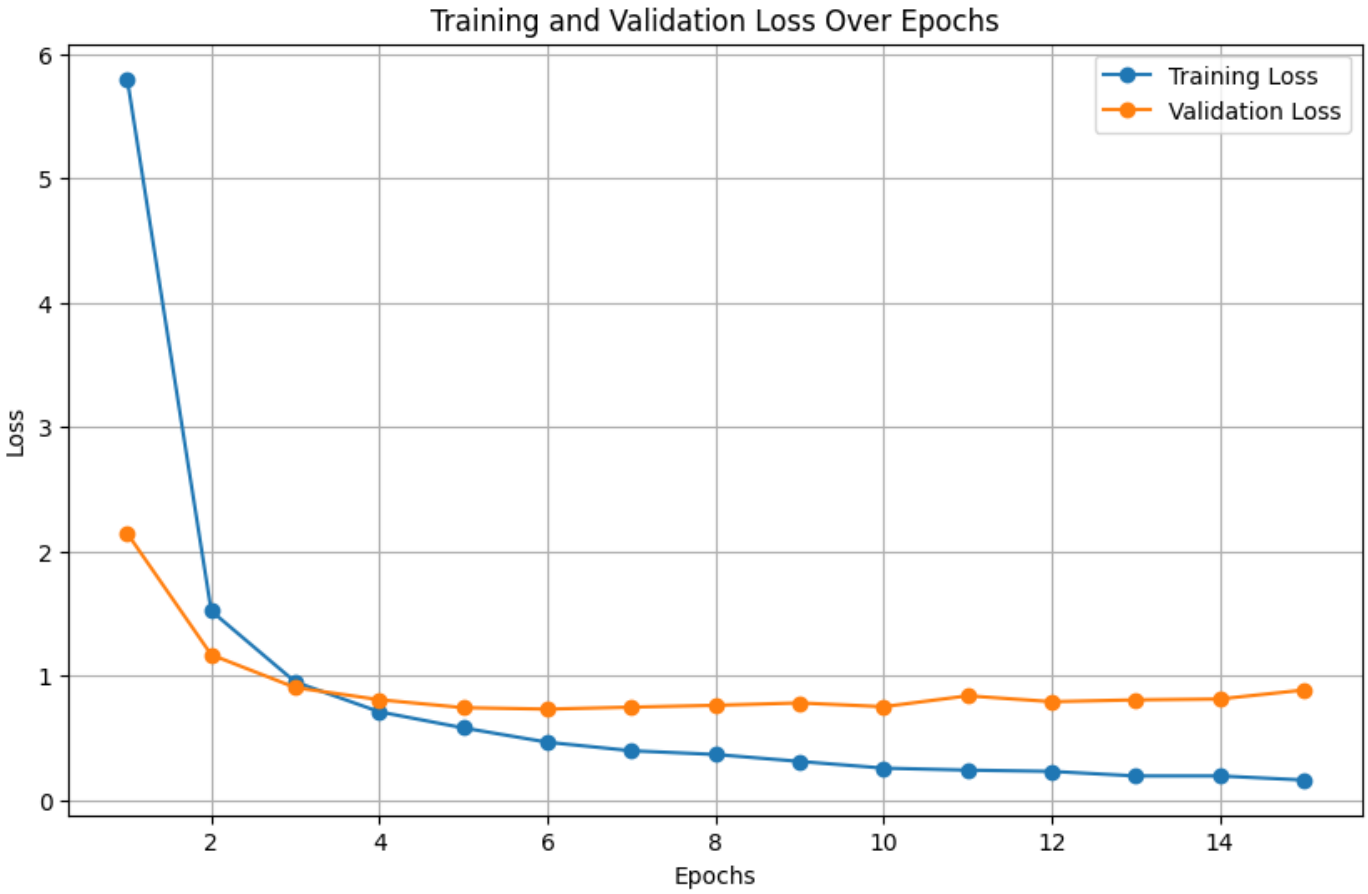

Figure 13: Training and Validation loss curves of mixed data

Epoch	Accuracy	Loss	Val Accuracy	Val Loss
1	0.5364	9.0114	0.7819	2.1442
2	0.7965	1.7701	0.8025	1.1679
3	0.8413	1.0149	0.8223	0.9037
4	0.8765	0.7262	0.8255	0.8063
5	0.8974	0.5956	0.8276	0.7425
6	0.9215	0.4673	0.8333	0.7308
7	0.9325	0.3975	0.8192	0.7465
8	0.9457	0.3381	0.8242	0.7611
9	0.9510	0.3192	0.8244	0.7806
10	0.9612	0.2536	0.8272	0.7500
11	0.9618	0.2428	0.8095	0.8376
12	0.9610	0.2396	0.8282	0.7905
13	0.9742	0.1838	0.8181	0.8048
14	0.9709	0.1917	0.8192	0.8120
15	0.9779	0.1580	0.8114	0.8840

Table 6: Training and validation performance of the mixed dataset.

4.3.2 Predictions on Mixed Dataset

Following training on the mixed dataset, comprising real data with synthetic samples augmenting the imbalanced classes, the model's performance was evaluated on the mixed test set. The results are summarized in the confusion matrix (Figure 14) and the per-class classification metrics in Table 7.

The confusion matrix (Figure 14) illustrates the model's ability to correctly classify samples across the seven genre categories. The largest number of true positives was observed in the Cooking (709), Computers (708), and Fiction (586) classes, showing good performance in these categories. Misclassifications were

present, with the most confusion occurring between Biography and History, as well as some overlap between Business and Computers.

Detailed evaluation metrics are provided in Table 7. Precision values ranged from 0.64 (Biography) to 0.96 (Cooking), while recall spanned 0.68 (History) to 0.94 (Computers). The highest F1-score was achieved for Cooking (0.95), showing a good balance between precision and recall in this genre. Overall test set accuracy was 0.81. Both macro and weighted averages for precision, recall, and F1-score were 0.81, showing consistent performance across all classes.

The model trained on the mixed dataset achieved an overall accuracy of 81% and maintained high precision, recall, and F1-score values across most genres. The addition of synthetic samples to address class imbalance helped generalization, especially in underrepresented categories.

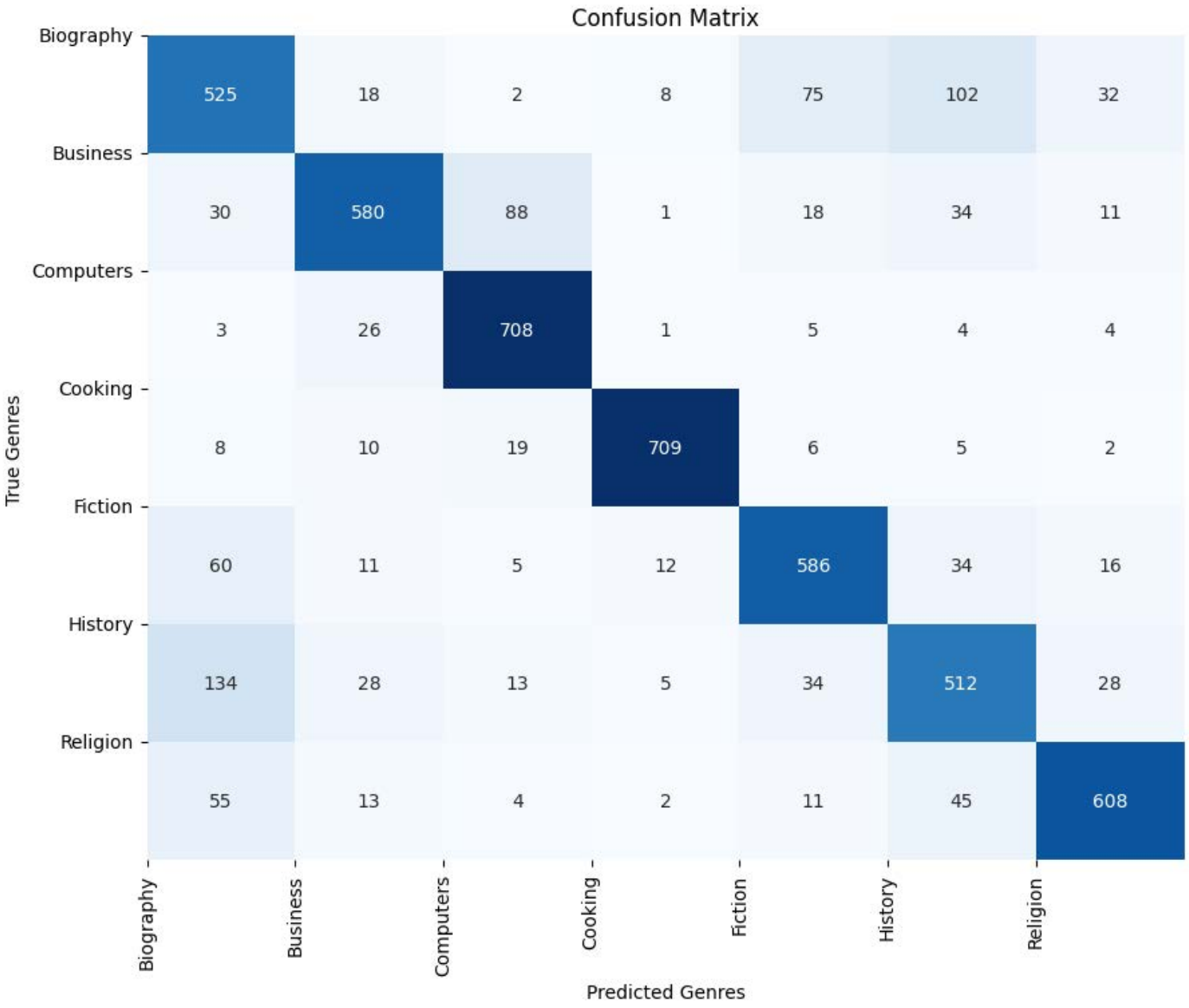

Figure 14: Confusion matrix of mixed data predictions on mixed data

Class	Precision	Recall	F1-score	Support
Biography	0.64	0.69	0.67	762
Business	0.85	0.76	0.80	762
Computers	0.84	0.94	0.89	751
Cooking	0.96	0.93	0.95	759
Fiction	0.80	0.81	0.80	724
History	0.70	0.68	0.69	754
Religion	0.87	0.82	0.85	738
Accuracy	-	-	0.81	5250
Macro Avg	0.81	0.81	0.81	5250
Weighted Avg	0.81	0.81	0.81	5250

Table 7: Classification report for the mixed dataset on the mixed test set.

4.3.3 Predictions on Real Dataset

The generalization capability of the model trained on the mixed dataset was evaluated on the held-out real test set. Predictive results are summarized in the confusion matrix (Figure 15) and the corresponding per-class classification report (Table 8).

The confusion matrix (Figure 15) illustrates model predictions for each of the seven genres. The results show high true positive rates across all categories, with the largest counts observed for Fiction (652), Computers (592), and Religion (703). Off-diagonal entries indicate relatively few misclassifications, with little overlap between most genres.

The performance metrics for each class are presented in Table 8. Precision values ranged from 0.81 (History) to 0.95 (Cooking), while recall values spanned 0.76 (Biography) to 0.97 (Cooking). Cooking exhibited the highest F1-score (0.96), showing both high precision and recall. The overall test set accuracy achieved by the model was 0.87. Macro and weighted averages for precision, recall, and F1-score were 0.88 and 0.87, respectively, showing balanced classification performance across all genres.

In summary, the mixed-trained model generalized well when evaluated on real data, achieving 87% accuracy with consistently high per-class precision, recall, and F1-score. These results confirm that incorporating synthetic samples to balance the training dataset helps model performance on real data.

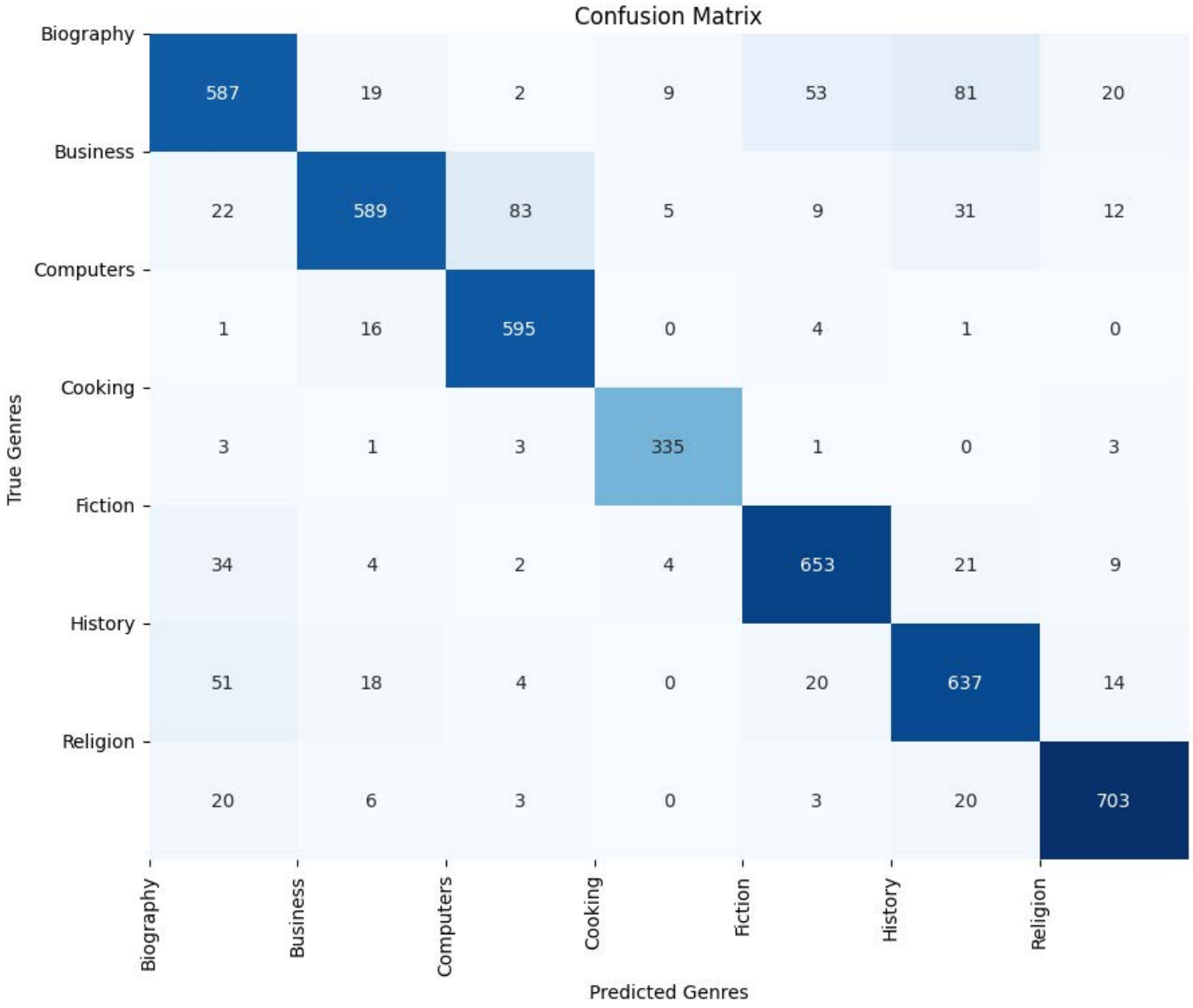

Figure 15: Confusion matrix of mixed data predictions on real data

Class	Precision	Recall	F1-score	Support
Biography	0.82	0.76	0.79	771
Business	0.90	0.78	0.84	751
Computers	0.86	0.96	0.91	617
Cooking	0.95	0.97	0.96	346
Fiction	0.88	0.90	0.89	727
History	0.81	0.86	0.83	744
Religion	0.92	0.93	0.93	755
Accuracy	-	-	0.87	4711
Macro Avg	0.88	0.88	0.88	4711
Weighted Avg	0.87	0.87	0.87	4711

Table 8: Classification report on the real test set.

5 Analysis

5.1 Comparative Performance Across Datasets

The comparative evaluation of the different training and prediction setups shows differences in performance across in-domain and cross-domain settings. Overall, the mixed dataset generalized best to the real test set, reaching an accuracy of 87%. This result outperformed both the real-only model (approximately 80%) and the synthetic-only model (67%), suggesting that the combination of synthetic and real data gives an advantage when models are applied to real-world samples.

5.1.1 In-domain performance

When evaluated on their respective test sets, the synthetic-trained model achieved the highest in-domain accuracy (92%), surpassing both the real-only (80%) and mixed models (81%). This shows that synthetic data

is easy to learn within its own distribution, likely because LLM-generated descriptions are more uniform in structure and style than human-written text. However, such improvements in synthetic in-domain performance do not directly translate to stronger results on real data.

5.1.2 Cross-domain generalization

The main outcome appears when looking at transfer to the real test set. The synthetic-only model showed a sharp performance drop (accuracy 67%), highlighting a distribution gap between generated and human-authored descriptions. The real-only model established a baseline of 80% accuracy, but the mixed model surpassed both, attaining 87%. This improvement comes mainly from the class balancing procedure: by augmenting minority classes with synthetic examples, the mixed dataset reduced the bias toward majority categories present in the real dataset. In addition, exposure to both synthetic and real linguistic styles diversified the model's training data, making it more robust to variation and less prone to overfitting.

5.1.3 Per-class trends

At the per-class level, Cooking consistently achieved the highest F1-scores across all settings (0.94 with real-only, 0.96 with synthetic-only, and 0.96 with mixed→real). This stability suggests that the genre's lexical and thematic cues are distinct enough to be captured regardless of data source. Computers also benefited in the mixed→real setting, achieving a recall of 0.96, which reflects the model's increased sensitivity after augmentation. By contrast, Biography and History remained difficult to disentangle. These two categories are inherently similar: a biography often contains historical context, while history texts often focus on the lives of key figures. This semantic overlap explains the persistent confusion between the two, even after class balancing.

5.1.4 Training dynamics

Training behaviors further reflect these tendencies. The real-only model peaked around epoch 5–6 in validation accuracy before mild overfitting appeared, as indicated by increasing validation loss. The synthetic-only model, instead, quickly reached high validation accuracy and kept it, though this stability masked poor transferability to real-world text. The mixed model balanced these extremes: it stabilized around 81–83%

validation accuracy and showed no significant decline, suggesting a favorable compromise between fit and generalization.

5.1.5 Summary

In summary, synthetic data alone yields strong in-domain performance but suffers from a distribution gap when applied to real data. Real-only training provides a reliable baseline, but the mixed dataset, through balancing and linguistic diversification, achieves the best results. These findings confirm that LLM-generated data works best when used to complement real-world samples, particularly for underrepresented classes.

5.2 Impact of Class Imbalance and Synthetic Balancing

A key challenge in the real dataset is the skewed distribution of samples across genres, with categories such as Fiction containing tens of thousands of entries while others, like Cooking or Computers, are comparatively underrepresented. This imbalance introduces a strong bias during training: the model is more likely to optimize toward majority classes, achieving high overall accuracy while misclassifying minority categories.

5.2.1 Effect of imbalance in the real-only model

The results from the real-only model show this issue clearly. While overall accuracy reached 80%, minority categories exhibited weaker per-class performance. For instance, Cooking and Computers performed adequately due to their distinctive lexical cues, but Biography and History were more prone to misclassification. The confusion matrix highlighted this, with Biography samples frequently predicted as History, reflecting both class overlap and the reduced representation of Biography relative to dominant classes.

5.2.2 Synthetic balancing in the mixed dataset

The mixed dataset directly addressed this limitation by supplementing underrepresented classes with LLM-generated descriptions until each class contained at least 5,000 entries. This balancing procedure equalized the number of samples per genre, ensuring that the model received a comparable training signal across all categories. The result was a clear increase in classification consistency: the mixed-to-real evaluation achieved

87% accuracy with both macro and weighted averages converging at 0.88, indicating that minority classes were no longer overshadowed by majority genres.

5.2.3 Why balancing works

Balancing improved performance for two reasons. First, the presence of additional synthetic samples prevented the model from defaulting to majority-class predictions, which raised recall for previously disadvantaged categories. Second, the synthetic descriptions, while cleaner than real text, provided typical examples of minority classes. These reinforced the semantic boundaries between categories, enabling the model to distinguish them more clearly during training. The improvements seen in recall for Computers (0.96) and in the overall F1-scores for Biography and History in the mixed model exemplify this effect.

5.2.4 Summary

In summary, the augmentation of minority classes with synthetic data was an important factor in the better performance of the mixed dataset. By mitigating imbalance, the model trained on mixed data avoided bias toward majority genres and achieved greater uniformity in classification quality across categories. This shows that one of the most useful applications of LLM-generated text is as a scalable balancing tool when real-world data is unevenly distributed.

5.3 Error Analysis and Genre Overlap

While aggregate accuracy and macro-level metrics provide a global view of performance, a closer inspection of the confusion matrices reveals systematic misclassifications that expose deeper challenges in genre separation. These errors are not random; rather, they follow meaningful semantic overlaps between certain classes.

5.3.1 Biography and History overlap

The most persistent source of error occurred between the Biography and History categories. In both the real-only and mixed models, a substantial number of Biography entries were misclassified as History, and

vice versa. This confusion arises from the intrinsic similarity of the two genres: a biography often situates an individual's life story within its historical context, while a history book may focus extensively on the lives and contributions of prominent figures. Consequently, the lexical and thematic boundaries between these genres are porous, making it difficult for the model to separate them even when class balancing is applied. This observation suggests that classification errors here are not solely a data imbalance problem but reflect an inherent semantic ambiguity in the categories themselves.

5.3.2 Minor misclassifications

Other genres such as Cooking and Religion were more clearly delineated, with consistently high F1-scores and minimal misclassifications. Cooking in particular achieved near-perfect separation across all models due to its unique culinary terminology, which distinguishes it sharply from other categories. Religion also maintained relatively distinct boundaries, although a small degree of confusion with History occasionally appeared, reflecting their shared narrative and cultural focus.

5.3.3 Summary

Error analysis demonstrates that the majority of misclassifications are not random noise but are concentrated among semantically or topically adjacent genres. This suggests that future work in classification should consider hierarchical or multi-label approaches, where texts can belong to overlapping categories simultaneously. Such strategies may more accurately reflect the inherent complexity of literary genres, where rigid categorical boundaries are often artificial.

6 Conclusion

This thesis investigated the role of Large Language Model (LLM)-generated synthetic data in multiclass text classification, using real, synthetic, and mixed datasets derived from book descriptions. The analysis revealed three key findings. First, synthetic data alone is highly learnable and achieves excellent in-domain performance (92% accuracy), but exhibits weaker transferability to real-world text (67%). Second, models trained exclusively on real data achieve reliable performance (80%) but remain constrained by class imbalance

and data noise. Third, the mixed dataset, which combined real and synthetic samples, consistently delivered the best results, reaching 87% accuracy on the real test set and achieving balanced precision and recall across all categories. These outcomes demonstrate that synthetic data is most effective when used to complement, rather than replace, real data.

6.1 Strengths and limitations of LLM-generated data

The results confirm the utility of LLMs in addressing data scarcity and imbalance. Synthetic augmentation successfully equalized class distributions, mitigated overfitting, and improved the sensitivity of models to minority categories. Furthermore, synthetic samples provided prototypical examples that reinforced semantic boundaries between genres, making the model more robust to variation. However, limitations remain. Synthetic text often exhibits stylistic homogeneity, which reduces its realism and hinders cross-domain generalization. Additionally, semantically adjacent genres such as Biography and History remain difficult to separate, underscoring that some classification errors arise from intrinsic category overlap rather than data quality alone.

6.2 Implications for practical applications

These findings have direct implications for domains where real data are scarce, sensitive, or imbalanced. In medical or legal contexts, for example, LLM-generated text could be used to augment training corpora without exposing confidential information, thereby enhancing model robustness while preserving privacy. Similarly, in business and scientific domains, synthetic augmentation could mitigate the bottleneck of limited annotated datasets. Nonetheless, the adoption of synthetic data requires careful quality control to avoid reinforcing biases or introducing artificial linguistic artifacts.

6.3 Future work

Several directions for future research emerge from this study. First, exploring alternative LLMs, embeddings, and network architectures may further improve the integration of synthetic data. Second, adaptive prompt engineering could generate domain-specific synthetic samples with higher fidelity. Third, quantitative

metrics for evaluating the quality and diversity of synthetic data remain an open challenge and would provide a principled basis for deciding when and how to integrate synthetic augmentation. Finally, hierarchical or multi-label classification frameworks may better capture the semantic overlap observed between certain genres, addressing limitations revealed by this work.

6.4 Closing remarks

In conclusion, LLM-generated synthetic data represents a powerful tool for enhancing machine learning tasks, particularly when used to balance and enrich real-world datasets. While synthetic data alone does not yet replace the complexity of authentic human text, its strategic integration offers a pathway toward more robust, generalizable, and equitable machine learning models.

References

[1] Shadi Iskander et al. "Quality matters: Evaluating synthetic data for tool-using LLMs". In: *arXiv preprint arXiv:2409.16341* (2024).

[2] Xu Liu et al. "Empowering Time Series Analysis with Synthetic Data: A Survey and Outlook in the Era of Foundation Models". In: *arXiv preprint arXiv:2503.11411* (2025).

[3] Ruibo Liu et al. "Best practices and lessons learned on synthetic data". In: *arXiv preprint arXiv:2404.07503* (2024).

[4] Lin Long et al. "On llms-driven synthetic data generation, curation, and evaluation: A survey, 2024". In: *URL https://arxiv. org/abs/2406.15126* (2024).

[5] Francisco Jáñez-Martino et al. “Spam email classification based on cybersecurity potential risk using natural language processing”. In: *Knowledge-Based Systems* 310 (2025), p. 112939. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2024.112939. URL: https://www.sciencedirect.com/science/article/pii/S0950705124015739.

[6] Ming Yang et al. “News text mining-based business sentiment analysis and its significance in economy”. In: *Frontiers in Psychology* 13 (2022), p. 918447.

[7] Ela Vashishtha and Himanshu Kapoor. “Enhancing patient experience by automating and transforming free text into actionable consumer insights: a natural language processing (NLP) approach”. In: *International Journal of Health Sciences and Research* 13.10 (2023), pp. 275–288.

[8] Shangshang Jin. “Sentiment-Driven Forecasting LSTM Neural Networks for Stock Prediction-Case of China Bank Sector.” In: *International Journal of Advanced Computer Science & Applications* 14.11 (2023).

[9] Nazanin Babolmorad and Nadia Massoud. “When sentiment is news: The polarity pattern approach”. In: *Available at SSRN 3706522* (2020).

[10] Muthuraman Thangaraj and Muthusamy Sivakami. “Text classification techniques: A literature review”. In: *Interdisciplinary journal of information, knowledge, and management* 13 (2018), p. 117.

[11] Qian Li et al. “A survey on text classification: From shallow to deep learning”. In: *arXiv preprint arXiv:2008.00364* (2020).

[12] Daniel Schwabe et al. “The METRIC-framework for assessing data quality for trustworthy AI in medicine: a systematic review”. In: *NPJ Digital Medicine* 7.1 (2024), p. 203.

[13] Usman Anjum et al. “Towards Modeling Data Quality and Machine Learning Model Performance”. In: *arXiv preprint arXiv:2412.05882* (2024).

[14] Haibo He and Edwardo A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239.

[15] Nathalie Japkowicz and Shaju Stephen. “The class imbalance problem: A systematic study”. In: *Intelligent data analysis* 6.5 (2002), pp. 429–449.

[16] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. “Classification of imbalanced data: A review”. In: *International journal of pattern recognition and artificial intelligence* 23.04 (2009), pp. 687–719.

[17] Yingzhou Lu et al. “Machine learning for synthetic data generation: a review”. In: *arXiv preprint arXiv:2302.04062* (2023).

[18] Mandeep Goyal and Qusay H Mahmoud. “A systematic review of synthetic data generation techniques using generative AI”. In: *Electronics* 13.17 (2024), p. 3509.

[19] Kuldeep Singh Kaswan et al. “Generative AI: a review on models and applications”. In: *2023 International Conference on Communication, Security and Artificial Intelligence (ICCSAI)*. IEEE. 2023, pp. 699–704.

[20] Gözde Gül Şahin. “To Augment or Not to Augment? A Comparative Study on Text Augmentation Techniques for Low-Resource NLP”. In: *Computational Linguistics* 48.1 (Apr. 2022), pp. 5–42. ISSN: 0891-2017. DOI: 10.1162/coli_a_00425.

eprint: https://direct.mit.edu/coli/article-pdf/48/1/5/2006622/coli_a_00425.pdf. URL: https://doi.org/10.1162/coli_a_00425.

[21] Jozef Kapusta et al. “Text Data Augmentation Techniques for Word Embeddings in Fake News Classification”. In: *IEEE Access* 12 (2024), pp. 31538–31550. DOI: 10.1109/ACCESS.2024.3369918.

[22] Marian Bucos and Georgiana □ucudean. “Text Data Augmentation Techniques for Fake News Detection in the Romanian Language”. In: *Applied Sciences* 13.13 (2023). ISSN: 2076-3417. DOI: 10.3390/app13137389. URL: https://www.mdpi.com/2076-3417/13/13/7389.

[23] Anna Hlédiková et al. “Data augmentation for dementia detection in spoken language”. In: *arXiv preprint arXiv:2206.12879* (2022).

[24] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[25] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.

[26] Emily M Bender et al. “On the dangers of stochastic parrots: Can language models be too big?” In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 2021, pp. 610–623.

[27] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[28] István Üveges and Orsolya Ring. “Evaluating the Impact of Synthetic Data on Emotion Classification: A Linguistic and Structural Analysis”. In: *Information* 16.4 (2025). ISSN: 2078-2489. DOI: 10.3390/info16040330. URL: https://www.mdpi.com/2078-2489/16/4/330.

[29] Ojasw Upadhyay, Abishek Saravanakumar, and Ayman Ismail. *SynLexLM: Scaling Legal LLMs with Synthetic Data and Curriculum Learning*. 2025. arXiv: 2504.18762 [cs.CL]. URL: https://arxiv.org/abs/2504.18762.

[30] Maayan Frid-Adar et al. “Synthetic data augmentation using GAN for improved liver lesion classification”. In: *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE. 2018, pp. 289–293.

[31] Jason Wei and Kai Zou. “Eda: Easy data augmentation techniques for boosting performance on text classification tasks”. In: *arXiv preprint arXiv:1901.11196* (2019).

[32] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of big data* 6.1 (2019), pp. 1–48.

[33] Lei Xu et al. “Modeling tabular data using conditional gan”. In: *Advances in neural information processing systems* 32 (2019).

[34] Mohamed Bekheet. *Amazon Books Reviews*. 2022. URL: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews/data.

[35] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global vectors for word representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. 2014, pp. 1532–1543.

[36] Hao Li et al. *Using the Training History to Detect and Prevent Overfitting in Deep Learning Models*. 2023. URL: https://openreview.net/forum?id=mzrNhoaHRDc.