Fiction Gene: Auditable Online Classification of Thai Fiction with Reversible Updates

ABSTRACT

Thai online fiction often changes genre signals across chapters, while production collections also require models that can incorporate and remove authorized documents without rebuilding the full corpus. This paper presents an auditable extension of Fiction Gene, our earlier Thai fiction classification and visualization system. The extension separates two modeling paths that serve different purposes: a fixed TF-IDF benchmark for predictive evaluation and an online Naive Bayes model whose additive sufficient statistics support revisioned Learn, Unlearn, Relabel, and Restore operations. The system records document contributions in a relational repository so that an accepted document can be removed by subtracting its stored contribution. It also visualizes aggregate genre composition and chapter-level progression while treating model outputs as relative genre signals unless calibration has been established. The available experiment contains 8,580 chapters in six platform-defined genres and reports 96.8% accuracy for TF-IDF under chapter-level cross-validation. Because chapters from the same story may occur in different folds, this result is retained as a legacy benchmark rather than evidence of story-independent generalization. We therefore define a story-grouped evaluation protocol, calibration measures, latency distributions, and round-trip state-equivalence tests required for the final assessment. The resulting framework links Thai NLP preprocessing, predictive evaluation, reversible model updates, and provenance-aware visualization without conflating browser demonstrations, batch models, and the online research model.

Keywords: Thai fiction classification, online Naive Bayes, reversible learning, sufficient statistics, TF-IDF, model audit

2000 Mathematics Subject Classification: 68T50, 68T05

1. INTRODUCTION

The exponential growth of digital fiction platforms has created a demand for automated genre classification systems to help users navigate extensive collections. Fiction classification is a key Natural Language Processing (NLP) task, where machine learning models analyze textual content to assign genres. Manual classification is impractical for platforms with thousands of stories, making automated classification essential for improving content organization, search efficiency, and personalized recommendations.

The first Fiction Gene study classified 3,134 chapters from 120 stories using TF-IDF and Naive Bayes and reported 91% accuracy [1]. It also introduced genre-distribution and chapter-progression visualizations. The present work examines a different operational question: how can a Thai fiction classifier accept, reverse, and audit document-level updates while keeping its predictive benchmark distinct from the online model? This distinction matters because a fixed TF-IDF model and an additive count model do not share the same update semantics.

Content corrections and withdrawals create a second requirement. For a classifier defined by additive sufficient statistics, a document can be removed exactly when the system retains the document contribution used during learning. We study this reversible update as a restricted, testable form of machine unlearning rather than claiming a general unlearning algorithm for arbitrary models.

Repeated preprocessing and corpus scans also increase prediction latency. The implementation therefore persists versioned word statistics and class counts in MySQL. We use the term feature statistics repository for these tables because the implementation does not yet demonstrate the full offline-online consistency and point-in-time semantics normally associated with a production feature store.

This research extension makes four contributions. First, it defines separate batch and online classification paths that share a preprocessing specification but use representations appropriate to their update semantics. Second, it formalizes revisioned Learn, Unlearn, Relabel, and Restore operations for the count-based model. Third, it connects each prediction and visualization to a model and dataset revision. Fourth, it specifies a leakage-aware evaluation protocol that groups chapters by story and measures predictive quality, calibration, latency, and reversible-update correctness.

The existing implementation reports a reduction in retrieval time from 846.7 seconds to 0.2 seconds and document-update times near 0.2 seconds. These point estimates are preserved as legacy system measurements. The final evaluation must reproduce them on a documented platform with repeated trials and latency distributions before they are used as headline results.

The remainder of this paper reviews related work, documents the corpus and Thai preprocessing pipeline, defines the batch and online models, describes reversible update semantics and system architecture, and evaluates the current evidence with explicit threats to validity.

1.1 Extension over the Conference Paper

This article substantially extends the ICSEC 2024 paper [1]. The conference version established the original

corpus, TF-IDF classification pipeline, and genre visualizations. The journal extension introduces a new corpus, separates fixed TF-IDF benchmarking from online sufficient-statistic learning, adds reversible and revisioned data operations, and defines grouped evaluation and audit tests. The earlier publication is cited wherever its data, methods, or system design provide the starting point for the present work.

2. RELATED WORKS

Fiction genre classification is a key subtask in text classification, where Natural Language Processing (NLP) techniques are used to categorize literary works based on their textual features. This section reviews existing research on text classification techniques, incremental learning, feature store optimizations, and machine unlearning, with a particular emphasis on Thai text classification due to its unique linguistic characteristics.

2.1 Advancements in Text Classification Techniques

Text classification has evolved significantly from traditional machine learning models such as Naïve Bayes, Support Vector Machines (SVMs), and Decision Trees to more advanced deep learning-based approaches. Early studies demonstrated the effectiveness of SVMs in handling high-dimensional text data, while Naïve Bayes classifiers have been widely used due to their efficiency in probabilistic text classification [11], [12]. However, these models often struggle with semantic relationships and long-range dependencies, leading to the adoption of deep learning architectures such as Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNNs), and Transformer-based models [13][14].

Recent advancements in Transformer architectures, such as BERT (Bidirectional Encoder Representations from Transformers), have further improved classification performance by capturing bidirectional contextual dependencies [15]. However, while deep learning models have demonstrated superior accuracy, they require large-scale labeled datasets and high computational resources, limiting their feasibility for real-time applications. This study focuses on a real-time, online machine learning approach rather than deep learning-based classification, emphasizing efficiency, scalability, and adaptability.

2.2 Thai Text Classification Challenges

Thai text classification presents unique challenges due to the absence of explicit word delimiters, tonal variations, and complex morphological structures. Unlike English, where words are naturally separated by spaces, Thai text is written continuously, making word segmentation a critical preprocessing step. Prior studies have compared SVM, Naïve Bayes, and deep learning models for Thai sentence classification, demonstrating that segmentation quality directly impacts classification accuracy [16], [17].

Beyond general text classification, Thai NLP applications have extended to sentiment analysis, traffic information extraction, and profanity detection, highlighting the increasing need for robust language processing models [18]–[21]. This work builds upon existing research by optimizing feature extraction, classification retrieval, and incremental updates to enhance the accuracy and efficiency of Thai fiction genre classification.

2.3 Incremental Learning and Online Learning

Incremental learning techniques allow models to dynamically adapt to new data without requiring full retraining. Unlike traditional batch learning, where models are retrained periodically on accumulated data, incremental learning enables real-time updates, making it ideal for dynamic text classification tasks such as news classification, fraud detection, and streaming data analysis [9], [22].

Prior research has introduced adaptive learning methods that adjust models to shifting data distributions [23], as well as memory-efficient architectures to mitigate catastrophic forgetting in continual learning [24]. The Fiction Gene system incorporates incremental learning to ensure that new fiction entries are immediately reflected in genre probability distributions, eliminating the need for full retraining and significantly improving classification efficiency.

2.4 Feature Statistics Repositories in Machine Learning

Feature management systems centralize reusable model inputs and can reduce redundant computation [25]. Their production guarantees may include feature definitions, versioning, offline-online consistency, entity keys, and point-in-time retrieval. The current Fiction Gene implementation uses a narrower relational feature statistics repository that stores class totals and per-genre word counts. We retain this narrower term throughout the method description.

Persisted statistics have been used to reduce repeated computation in recommendation, fraud detection, and real-time analytics [25], [26]. In Fiction Gene, the repository supports prediction and reversible updates by reading or modifying only the statistics associated with an operation. The latency benefit is evaluated separately from classification accuracy because storage placement does not by itself improve the learned decision rule.

2.5 Machine Unlearning and Content Correction Mechanisms

Machine unlearning plays a critical role in ensuring privacy compliance, content moderation, and adaptive learning in AI-driven systems. Traditional machine learning models lack the capability to selectively forget specific data without full retraining, making data removal an expensive and computationally intensive process. Recent advancements in machine unlearning introduce approximate forgetting algorithms and selective memory updates to enable models to remove specific data points without disrupting overall classification accuracy [4], [10].

Studies on unlearning in large language models (LLMs) emphasize the importance of removing harmful responses, biased content, and copyrighted material while preserving essential knowledge [27], [28]. This study extends these concepts by incorporating an SQL-based unlearning mechanism that updates stored word frequency distributions and genre probabilities when a fiction entry is deleted, ensuring that outdated data does not influence future classifications.

It is worth noting that the "approximate forgetting" framing common in this literature [4], [10] targets models where exact unlearning is computationally intractable (e.g., deep networks). For models built on additive sufficient statistics, such as the multinomial Naïve Bayes classifier used in this study, unlearning can instead be performed exactly; Section 3.6 develops this distinction in the context of the Fiction Gene system.

3. METHODOLOGY

The method separates predictive evaluation from mutable model operation. A fixed TF-IDF lane supports controlled batch comparisons, while an additive count lane supports online prediction and reversible updates. Both lanes consume text produced by one versioned Thai preprocessing specification. This section documents the corpus, preprocessing, representations, classifiers, and update semantics.

3.1 Overall Framework

Figure 1 separates the two roles of the system. Both lanes apply the same versioned Thai preprocessing specification, but they do not share a mutable feature representation. The fixed benchmark lane fits TF-IDF within each training fold and compares batch classifiers. The reversible online lane stores per-chapter token-count contributions, aggregates them as Naive Bayes sufficient statistics, and records each Learn, Unlearn, Relabel, or Restore event against a model revision. Consequently, an unlearning request reverses additive counts rather than editing a fitted TF-IDF vectorizer.

3.2 Data Collection

The enhanced Fiction Gene system retrieves fiction data directly from the Dek-D API, ensuring structured and policy-compliant data acquisition. Unlike the previous version, which relied on web scraping, this approach enhances data accuracy, system scalability, and compliance with data privacy policies.

The dataset consists of 8,580 fiction chapters, categorized into six primary genres: Horror, Suspense, Fantasy, Science Fiction, Action, and Drama. These genres align with Dek-D's official classification system, ensuring standardized labeling and consistency in genre assignments.

To facilitate efficient storage and retrieval, fiction metadata, chapter text, and genre classifications are extracted using structured API requests and stored in a relational database (MySQL). The data schema is designed to optimize indexing and retrieval, enabling efficient classification at both the chapter and story level.

A chapter-based classification approach is employed to improve the granularity of genre analysis. Instead of assigning a single genre per story, the system classifies each chapter independently, allowing for the detection of genre shifts within a fiction work.

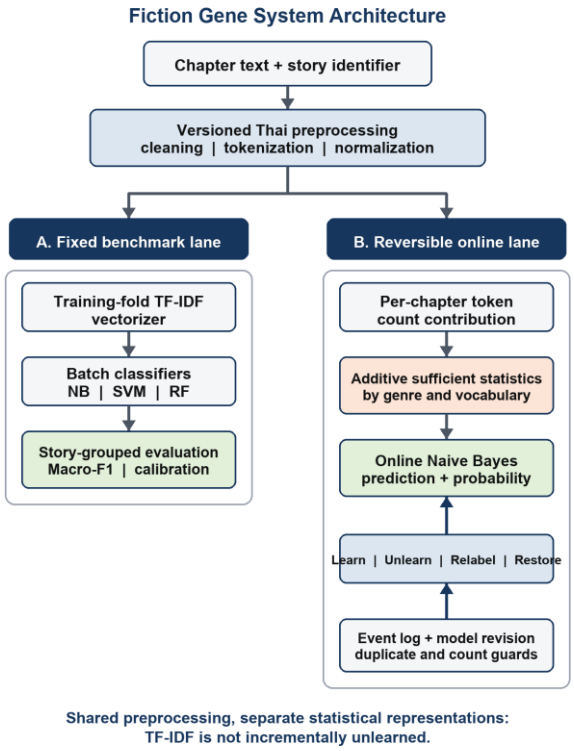

Fig. 1. Two-lane Fiction Gene architecture for fixed evaluation and reversible online classification.

3.3 Data Preprocessing

Effective data preprocessing is fundamental to improving the accuracy and efficiency of fiction genre classification. The enhanced Fiction Gene system employs a structured preprocessing pipeline to refine the raw text before feature extraction. Given the complexity of Thai text processing, several key preprocessing steps are implemented to standardize input data, minimize noise, and optimize feature representation.

3.3.1 Whitespace Normalization and Symbol Removal

Thai fiction often contains inconsistent spacing, decorative symbols, and manual line breaks that can interfere with text segmentation and tokenization. Regular expression-based cleaning techniques are applied to remove redundant whitespace, unnecessary punctuation, and non-linguistic symbols, while preserving meaningful textual structure. Additionally, numerical values such as chapter numbers and dates are filtered out, as they do not contribute to genre classification.

3.3.2 Tokenization

Unlike English, Thai lacks explicit word boundaries, necessitating robust segmentation methods to identify meaningful tokens. The system utilizes the PyThaiNLP newmm tokenizer, which utilizes a maximum matching algorithm to accurately segment words. For example, the sentence "ฉันชอบอ่านนิยายแฟนตาซี" (I like reading fantasy fiction) is tokenized into ["ฉัน", "ชอบ", "อ่าน", "นิยาย", "แฟนตาซี"], ensuring that meaningful terms like "นิยาย" (fiction) and "แฟนตาซี" (fantasy) contribute effectively to genre classification.

3.3.3 Stopword and Proper Noun Filtering

Stopwords are common words that frequently appear in text but do not provide meaningful information for distinguishing genres. These include function words, auxiliary verbs, conjunctions, and pronouns, which generally serve a grammatical purpose rather than a semantic one. Examples of Thai stopwords include "และ" (and), "คือ" (is), "ที่" (that), "ได้" (can), and "เป็น" (to be). Without filtering, stopwords may dominate the word frequency distribution, reducing the weight of genre-specific words and negatively impacting TF-IDF feature extraction. To remove stopwords, the Fiction Gene system utilizes the PyThaiNLP stopword corpus, which contains a predefined list of common Thai stopwords.

In addition to stopword removal, proper noun filtering is implemented to prevent named entities from influencing genre classification. Proper nouns, including names of people, locations, organizations, and fictional characters, are unique to individual stories and do not inherently indicate a genre. For example, in the sentence "สมชายเดินทางไปยังอาณาจักรไอเทอร์โนเพื่อพบกับเจ้าหญิงมินา" (Somchai traveled to the Aeterno Kingdom to meet Princess Mina), names like "สมชาย" (Somchai), "ไอเทอร์โน" (Aeterno), and "มินา" (Mina) do not contribute to genre classification. If left unfiltered, the system may incorrectly associate specific character or location names with a genre, rather than using narrative elements to determine classification. After proper noun filtering, the refined sentence "เดินทางไปยังอาณาจักรเพื่อพบกับเจ้าหญิง" (Traveled to the kingdom to meet the princess) retains genre-relevant terms, such as "kingdom," "princess," and "travel," while removing irrelevant named entities.

The Fiction Gene system implements proper noun filtering using a dictionary-based approach combined with named entity recognition (NER). A predefined lexicon of common Thai names, city names, and place names is used to systematically identify and exclude known proper nouns.

3.3.4 Handling of Out-of-Vocabulary (OOV) Terms

During tokenization, it was observed that certain words were not recognized by the Royal Institute Dictionary, leading to incorrect segmentations that either fragmented meaningful words or merged unrelated terms. This issue was particularly evident in fiction-related terminology, slang, and loanwords, which were often split into multiple segments that did not reflect their intended meaning.

Table 1: *Examples of Tokenization Before and After Including Vocabularies to the Customized Dictionary*

Original Tokenization	Tokenization with OOV Handling
['พาร์', 'กิน', 'ส้น']	['พาร์กินส้น']
['มังกร', 'เพลิง', 'ฟ้า']	['มังกรเพลิงฟ้า']
['นว', 'ราตรี']	['นวราตรี']
['จักรกล', 'ชีวภาพ']	['จักรกลชีวภาพ']
['เงา', 'อัคคี']	['เงาอัคคี']

To address this issue, rather than employing subword decomposition, the opposite approach was applied—merging frequently occurring multi-word terms into single recognized lexical units. As illustrated in Table 1, a term such as "มังกรเพลิงฟ้า" (Heavenly Fire Dragon) is commonly encountered in fantasy fiction. However, standard tokenization methods would incorrectly segment it into "มังกร" (dragon), "เพลิง" (fire), and "ฟ้า" (blue). This segmentation could mislead the classification model by associating words such as "fire" and "blue" with irrelevant thematic elements rather than recognizing "มังกรเพลิงฟ้า" as a meaningful entity within the narrative.

Although this approach improved tokenization accuracy and preserved fiction-specific terminology, it was not feasible to implement comprehensive OOV handling in the production system, as the introduction of new words in fiction writing is continuous and highly dynamic. Fiction authors frequently create unique character names, fantasy-related terms, and specialized domain vocabulary, making it impractical to maintain an exhaustive dictionary that encompasses all newly coined terms.

To mitigate this limitation, a customized dictionary was incorporated, leveraging the previously compiled vocabulary from the first version of the Fiction Gene system. This dictionary contained domain-specific fiction terminology and frequently used words in Thai online fiction, allowing the current system to build upon the original dataset's linguistic refinements.

3.3.5 Text Normalization and Spelling Correction

Spelling variations, particularly in informal and user-generated fiction, can introduce inconsistencies in feature extraction and classification accuracy. Variants of the same word may be interpreted as separate entities, leading to fragmented term frequency distributions in TF-IDF vectorization. To mitigate this issue, the Fiction Gene system incorporates a custom Thai spelling correction dictionary, which maps common misspellings to their standardized forms. For example, "เซ็นต์ชื่อ" → "เซ็นชื่อ" (sign name), "เท่ห์" → "เท่" (cool), and "โครต" → "โคตร" (extremely). This approach enhances text consistency, allowing identical words to be processed uniformly, which improves

feature representation and genre classification performance.

Table 2 presents an example of text preprocessing and tokenization applied to a fiction passage. The raw text consists of complex sentence structures, punctuation, and stopwords, which may introduce noise into the classification process. After preprocessing, the text is tokenized into individual words and meaningful phrases, removing unnecessary elements while preserving essential linguistic and contextual features.

Table 2: *Examples of the Preprocessed Text After Tokenization*

Raw text	Preprocessed text
ชลนั่งอยู่บนเก้าอี้ยาวภายในห้องโถงใหญ่ของโรงพยาบาลมีคนไข้และญาติผู้ป่วยมานั่งรอรับยาและรอเรียกเข้าไปตรวจร่างกาย เขานั่งกุมขมับด้วยความเครียดเพื่อนของเขาค่อย ๆ เสียชีวิตไปทีละคน ๆ และทุกครั้งก่อนที่พวกเขาจะตายเขาจะได้ยินคำสารภาพรักไม่ว่านกหรือจิ๊บ	['ชล', 'นั่ง', 'เก้าอี้ยาว', 'ห้องโถง', 'โรงพยาบาล', 'คนไข้', 'ญาติ', 'ผู้ป่วย', 'นั่ง', 'รอ', 'ยา', 'รอ', 'เข้าไป', 'ตรวจ ร่างกาย', 'นั่ง', 'กุมขมับ', 'ความเครียด', 'เพื่อน', 'เสียชีวิต', 'คน', 'ก่อนที่', 'ตาย', 'ได้ยิน', 'คำสารภาพ', 'รัก', 'นก', 'จิ๊บ']

The word cloud of the Drama genre (Fig.2) and the Fantasy genre (Fig.3) highlight distinct linguistic patterns that differentiate these genres.

In the Drama word cloud, frequently occurring words such as "เสียง" (sound), "ตอนนี้" (now), "คิด" (think), and "ตัวเอง" (oneself) suggest a strong emphasis on introspection, emotional depth, and character-driven storytelling. Drama fiction often revolves around personal conflicts, relationships, and internal monologues, which is reflected in the frequent use of conversational and reflective expressions.

In contrast, the Fantasy word cloud contains words like "ออกมา" (come out), "พลัง" (power), "ระดับ" (level), and "ตื่น" (awaken), indicating a focus on action, supernatural elements, and structured world-building. The prominence of power-related terminology, hierarchical progression, and awakening concepts suggests that Fantasy fiction often incorporates magical systems, adventure-driven plots, and structured character development, commonly observed in high fantasy and light novel storytelling.

Fig.2: *Word Cloud Representation of Common Words in the Drama Genre*

Fig.3: *Word Cloud Representation of Common Words in the Fantasy Genre*

3.4 Feature Extraction

The two model lanes use different numerical representations. The batch lane computes TF-IDF vectors fitted on each training partition. The online lane records additive token contributions that can be applied or reversed without refitting the TF-IDF benchmark. This separation allows predictive accuracy and update cost to be evaluated independently.

3.4.1 TF-IDF Vectorization

Term Frequency-Inverse Document Frequency (TF-IDF) is widely used to measure word importance within a document relative to a dataset. TF-IDF assigns a weight to each word based on its frequency within a single chapter (TF) and its distribution across all fictions (IDF). This weighting system helps highlight words that are significant for classification while minimizing the influence of common words. The TF-IDF score for a term t in a document d is calculated as:

$$\mathrm{TF}_{\mathrm{IDF}(t,d)} = \mathrm{TF}(t,d) \times \mathrm{IDF}(t) \qquad (1)$$

where TF (Term Frequency) measures how often a word appears in a document and is computed as:

$$\mathrm{TF}(t,d) = \frac{f(t,d)}{\sum_{t'} f(t',d)} \qquad (2)$$

where $f(t,d)$ is the number of times term t appears in document d, and $\sum_{t'} f(t',d)$ represents the total number of terms in d.

IDF (Inverse Document Frequency) accounts for how common a word is across all genres:

$$\mathrm{IDF}(t) = \log\left(\frac{N}{1+\mathrm{DF}(t)}\right) \qquad (3)$$

where N is the total number of documents, and $\mathrm{DF}(t)$ is the number of documents that contain t.

3.4.2 Additive Count Representation

The online path represents each authorized chapter by the count or document-presence contribution of its normalized tokens. Per-genre feature totals and class-

document totals are additive sufficient statistics for Naive Bayes. Prediction reads a consistent model revision without changing these totals. Learn adds a stored document contribution after a human-confirmed label; Unlearn subtracts that same contribution; Relabel performs an Unlearn from the previous class followed by a Learn into the new class; and Restore reapplies a previously removed contribution.

3.4.3 Versioned Feature Statistics Repository

The relational repository contains per-genre feature totals and cached document totals. Each mutating operation must also record the document identifier, label, preprocessing revision, contribution vector, operation type, and resulting model revision. These records allow the system to reject duplicate operations, prevent negative counts, reconstruct model lineage, and test whether a reversal returns the model to its prior state.

3.5 Online Naive Bayes Classification

The Fiction Gene system employs an incremental Naïve Bayes classification model to dynamically update genre probabilities as new fiction data is introduced. Unlike traditional batch learning approaches, which require periodic retraining on the entire dataset, incremental learning allows the model to adjust genre probability distributions in real time. This approach enhances computational efficiency and adaptability, making it well-suited for large-scale fiction classification.

The probability of a document D belonging to a genre G is computed using the product of individual probabilities of words occurring in that genre. Assuming words occur independently given the genre, the probability is represented as:

$$P(G|D) \propto P(G) \times \prod_{i=1}^{n} P(w_i|G) \quad (4)$$

where $\boldsymbol{P(G)}$ represents the prior probability of the genre, and $\boldsymbol{P(w_i|G)}$ is the likelihood of each word $\boldsymbol{w_i}$ appearing in that genre. To mitigate numerical underflow issues associated with the multiplication of small probabilities, the logarithmic form of Bayes' theorem is applied:

$$\boldsymbol{\log P(G|D) = \log P(G) + \sum_{i=1}^{n} \log P(w_i|G)} \quad (5)$$

This transformation allows the system to handle large vocabulary sizes more effectively while maintaining numerical stability.

Online learning updates the stored class and feature totals for the confirmed label. It does not update the fixed TF-IDF benchmark, whose vocabulary and IDF values are fitted only on the training partition. Keeping these paths separate prevents an online count update from being misreported as an incremental update to the TF-IDF model.

The additive update reduces the work required to incorporate one authorized document. Its predictive quality and latency must nevertheless be compared with the fixed batch models on identical story-grouped folds and the same preprocessing revision.

3.6 Reversible Sufficient Statistic Removal

The online model supports exact removal of a learned document from its additive sufficient statistics. This operation is narrower than general machine unlearning: it applies only when the stored contribution, preprocessing revision, and original label are available and when the model state has not been modified outside the recorded event sequence.

For the count-based classifier, subtracting the exact stored contribution is mathematically equivalent to omitting that contribution from the current aggregate. Correctness is therefore evaluated by state equivalence, not response time alone. After Learn followed by Unlearn, all class and feature totals and a fixed regression set of predictions must return to the pre-Learn snapshot within numerical tolerance.

3.6.1 Formal State and Update Operators

Let C be the set of genres and V the fixed vocabulary for preprocessing revision r. At model revision t, the sufficient state of each genre comprises its learned-chapter count and a vector of aggregate token counts. For an authorized chapter d, let x denote its stored token-count contribution. The state transitions are defined as follows.

$$\text{Learn}_c(S_c^t, x_d) = (N_c^t + 1, c_c^t + x_d)$$
$$\text{Unlearn}_c(S_c^t, x_d) = (N_c^t - 1, c_c^t - x_d)$$

Unlearn is admissible only when the event is active, the learned-chapter count is positive, and every component of x is no greater than the corresponding stored class total. Relabel is the atomic composition of Unlearn under the old class and Learn under the new class. Restore reapplies the same contribution under its recorded preprocessing revision.

3.6.2 Exact State Equivalence

Theorem 1 (round-trip state equivalence). For any admissible state S and stored contribution x_d, applying Learn and then Unlearn to the same class returns the complete sufficient state to its initial value.

$$\text{Unlearn}_c(\text{Learn}_c(S_c, x_d), x_d) = S_c$$

Proof. Componentwise addition followed by subtraction restores both the learned-chapter count and every token count. Therefore every Laplace-smoothed likelihood is also restored, where T_c denotes the total token count for genre c:

$$P(w_j|c) = \frac{c_{c,j} + \alpha}{T_c + \alpha|V|}$$

Because the class counts, token counts, vocabulary, smoothing parameter, preprocessing revision, and tie-breaking rule are unchanged, the posterior scores and predictions for any fixed regression set are identical before Learn and after the matching Unlearn. This result establishes exact unlearning for the stated additive model; it does not imply deletion from raw-data stores, logs, backups, or unrelated derived artifacts.

The removal executes as one database transaction. It decrements the affected feature totals and class-document count, verifies that no value becomes negative, records the removal event, and commits a new model revision. Relabel and Restore reuse the same stored contribution. Duplicate or missing event identifiers fail without mutating the model.

Reversible model updates can support correction and content-withdrawal workflows, but removal from model statistics does not by itself establish legal compliance. Raw text, logs, backups, caches, and derived artifacts require their own retention and deletion policies.

4. System Architecture and Implementation

The research system contains a web client, a Flask service, and a MySQL repository. The public Lab additionally contains a device-local browser baseline used to demonstrate the interface. Browser predictions and localStorage Learn or Unlearn actions do not mutate the central research model. A production request must identify the dataset, preprocessing, and model revisions used for its result.

4.1 Web Application

The web application accepts text for read-only analysis, presents authorized fiction collections, and displays aggregate and chapter-level genre signals. Administrative Learn and Unlearn operations are separated from prediction and require an explicit label, document identity, authorization status, and impact preview.

The interface presents read-only classification and chapter-sequence visualizations. When connected to the research service, an authorized administrative workflow may submit a separate Learn or Unlearn request. The public browser demonstration computes local genre signals and does not update the research model.

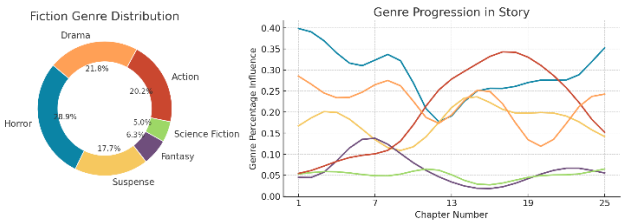

Fig.4: *Genre Distribution Pie Charts and Genre Progression Line Charts*

Figure 4 illustrates the genre classification results through two visualization techniques: genre distribution pie charts and genre progression line charts. These visual representations provide insights into the classification outcomes, highlighting the probability distribution of genres within a fiction work and the evolution of genre characteristics across chapters.

The genre distribution chart aggregates normalized scores across observed chapters and summarizes composition at story level. It does not preserve narrative order.

The progression chart plots each genre score at discrete chapter positions. Interpolated curves aid visual reading but do not represent model estimates between chapters. The plotted values are relative signals unless a held-out calibration experiment establishes a probabilistic interpretation.

Together, the aggregate and progression views expose different aspects of the same chapter scores. Their role is exploratory: they help readers inspect composition and change, but they do not establish semantic understanding, calibrated confidence, or a causal account of the narrative.

4.2 Server-Side Application

The Flask service applies the versioned preprocessing pipeline and routes requests to the appropriate model lane. Read-only prediction does not mutate model state. Authorized Learn, Unlearn, Relabel, and Restore requests execute as transactions against the online statistics repository and create a new model revision.

The fixed TF-IDF models are used for offline benchmarking. The online Naive Bayes model reads per-genre feature totals and class counts from MySQL. Every response identifies the active model and preprocessing revisions so that results can be reproduced and audited.

4.3 Data Architecture and Storage Model

MySQL stores application records and the sufficient statistics used by the online classifier.

Application tables identify stories, chapters, labels, and authorized contributors. Model tables store per-genre feature totals and class-document totals. The reversible extension adds logical document-contribution, operation-event, and model-revision records to the earlier schema shown in Figure 5. Authentication remains outside the experimental model subsystem.

The application data consists of four main tables: `category`, `user`, `fiction`, and `chapter`. These tables store essential information related to fiction works, their authors, and their classification.

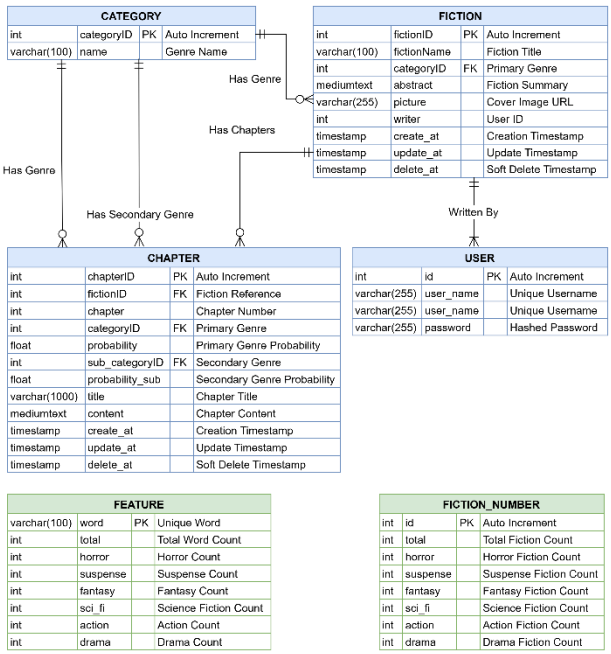

Fig.5: *ER-Diagram of Fiction Gene*

The `category` table defines the available fiction genres. Each fiction work and chapter is linked to a category, representing its primary and secondary genres.

The `user` table manages authentication and author information. Each user has a unique user ID, username, and password, allowing them to submit fiction works, track their submissions, and manage their content.

The `fiction` table contains metadata about fiction works, including fiction ID, title, category, and a brief abstract. Each fiction work belongs to a single primary genre and is linked to a category via a foreign key relationship. Additionally, fiction works are linked to the `user` table, allowing the system to track which user submitted each fiction work.

The chapter table is designed to store individual chapters of fiction works, enabling finer-grained classification by associating each chapter with its parent fiction work and primary genre. To enhance classification accuracy, the chapter table also supports secondary genre classifications, allowing fiction works to be categorized under multiple relevant genres. Furthermore, the table incorporates probability scores for both primary and secondary genre predictions, ensuring that genre classification is based on a confidence-weighted approach.

Upon the submission of a new chapter, the system immediately classifies the genre and records the detected genre and its corresponding probability scores in the table. As the classification model continues to evolve, these probability scores may be dynamically updated over time, reflecting refinements in the model's predictions. The stored scores serve as a historical record, allowing for further analysis of genre shifts, classification trends, and model performance improvements over time.

The classification data consists of two primary tables: `feature` and `fiction_number`. These tables store precomputed classification statistics, ensuring that the system can retrieve classification data without performing real-time text vectorization or dataset-wide computations.

The `feature` table is designed to store precomputed word frequency distributions across different fiction genres. Instead of performing TF-IDF vectorization or dynamic feature extraction, the system retrieves precomputed word frequencies for classification. Each row in the table represents a unique word, with columns corresponding to its occurrence in each fiction genre.

The `fiction_number` table plays a crucial role in reducing computational overhead by storing precomputed fiction counts per genre. Instead of dynamically counting the number of fiction works in each genre every time a classification request is made, the system retrieves stored fiction totals from this table.

5. EXPERIMENTATION

The empirical study separates predictive evaluation from systems evaluation. Predictive experiments compare representations and classifiers. Systems experiments measure read-only prediction and transactional model operations. The tables retained from V2 are labeled legacy results because their chapter-level split and timing protocol do not yet satisfy the revised evaluation design.

5.1 Dataset

The dataset consists of 8,580 fiction chapters, categorized into six primary genres: Horror (1,444

chapters), Suspense (1,426 chapters), Fantasy (1,490 chapters), Science Fiction (1,327 chapters), Action (1,481 chapters), and Drama (1,412 chapters). Unlike the first version of Fiction Gene, these fiction works were collected through the Dek-D API, facilitating structured and high-quality data acquisition. The dataset adheres to Dek-D's official genre classification system, eliminating inconsistencies that may arise from user-defined genre labels.

The distribution of fiction chapters across genres is depicted in Figure 6, illustrating the number of fiction chapters per genre. The dataset exhibits a relatively balanced distribution, with only minor variations in chapter counts. This balance is critical for training a robust classification model, as datasets with proportionally distributed categories tend to yield more reliable classification outcomes.

The word count distribution per chapter is presented in Figure 7, demonstrating variations in chapter length and its implications for feature extraction. On average, each fiction chapter contains 2,002.69 words, while the system's lexicon comprises approximately 35,000 unique words. Given the relatively short length of individual chapters, the likelihood of repeated term occurrences is lower, which poses challenges for term frequency-based feature extraction methods. This limitation suggests that raw term frequency (TF) alone may not effectively differentiate genres, as many words may appear only once per chapter. Instead, TF-IDF weighting is employed to adjust word importance dynamically, reducing the influence of high-frequency but non-discriminative terms while enhancing the relevance of genre-specific words.

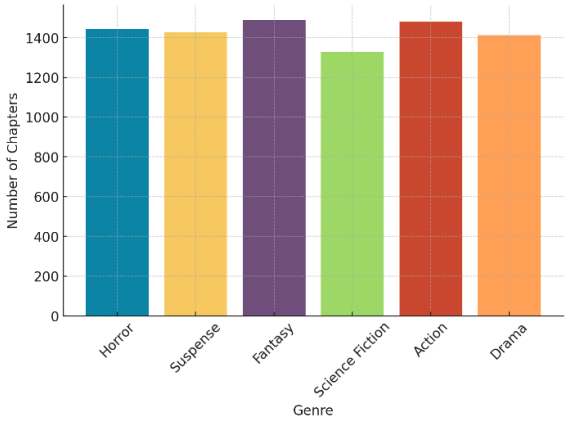

Fig.6: *Distribution of Chapters Across Genres*

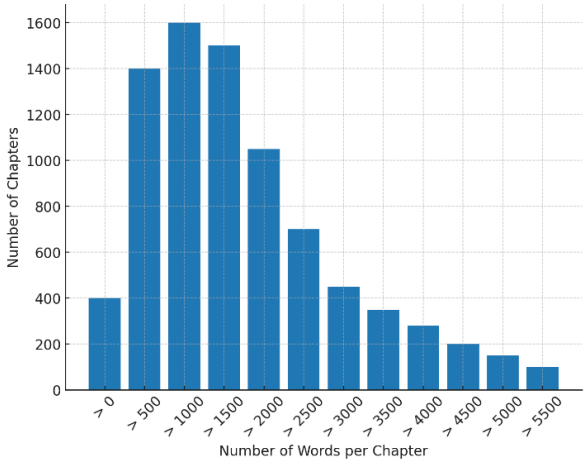

Fig.7: *Distribution of Word Counts Across Chapters*

5.2 Legacy Evaluation and Revised Protocol

The available V2 results were produced with random 10-fold partitioning at chapter level. Each chapter served once in a test fold and nine times in training folds. Because chapters are nested within stories, this protocol may place chapters from one story in both training and test partitions. We therefore report these values as a legacy benchmark and do not interpret them as story-independent generalization.

We evaluated three different feature extraction methods:

- TF (Term Frequency) – Measures the raw word occurrence in each document.
- IDF (Inverse Document Frequency) – Assigns lower weight to common words across all documents.
- TF-IDF – A combination of TF and IDF, weighting words based on importance in classification.
- Simple - The original experiment used this label for an unweighted baseline, but the retained materials do not define its representation precisely. Its values are shown for provenance and must be recomputed after the baseline is specified.

The revised protocol groups all chapters from a story into the same fold. It compares majority prediction, online count-based Naive Bayes, TF-IDF Multinomial Naive Bayes, TF-IDF Complement Naive Bayes, and a linear sparse-text classifier under one preprocessing revision. Predictive evaluation reports accuracy, Macro-F1, Weighted-F1, per-class recall, and a confusion matrix. Score quality is assessed with log loss, Brier score, and expected calibration error. System evaluation reports training time, prediction latency, Learn and Unlearn latency,

model size, and peak memory, with repeated trials summarized by median, p95, p99, and 95% confidence intervals.

5.3 Legacy Results and Interpretation

Under the legacy chapter-level partition, Table 3 reports an overall TF-IDF accuracy of 96.8%, compared with 80.2% for TF and 89.5% for IDF. These values show that lexical weighting was effective under the reported split. They remain provisional because the per-class aggregation must be recomputed from unrounded confusion counts and the grouped protocol has not yet been applied.

Table 3: *Classification Accuracy (%) of Different Feature Extraction Methods Across Genres*

Genre	Classification Accuracy (%)			
	Simple	TF	IDF	TF-IDF
Horror	72.5	85.2	88.1	96.2
Suspense	70.3	81.5	83.9	94.8
Fantasy	73.1	83.7	85.4	96.7
Sci- Fi	68.7	78.9	82.1	95.3
Action	71.4	82.3	85.7	96.1
Drama	69.2	80.4	83.6	94.2
Overall	74.6	80.2	89.5	96.8

The six chapter classes are similar in size, but the small differences in support do not establish a causal explanation for class-level performance. Story identity, vocabulary overlap, labeling noise, and the chapter-level split may also affect the reported values.

Table 4 retains the legacy precision, recall, and F1 values for the TF-IDF condition. The displayed Fantasy row is harmonized to 1.000 at three decimal places so that its reported F1 is consistent with the displayed precision and recall. These rounded values remain descriptive and must be replaced by fold-aggregated estimates from the story-grouped run.

Table 4: *Performance Metrics (Precision, Recall, and F1-Score) for Genre Classification Using TF-IDF*

Genre	Precision	Recall	F1-Score	Support
Horror	0.986	0.986	0.985	1444
Suspense	0.964	0.957	0.96	1426
Fantasy	1.000	1.000	1.000	1490
Sci-Fi	0.927	0.993	0.958	1327
Action	0.964	0.95	0.956	1481
Drama	0.992	0.921	0.955	1412
Accuracy		0.968		8580

Within the legacy experiment, TF-IDF produced the highest reported accuracy. This finding does not establish that TF-IDF is superior for unseen stories or that the feature statistics repository caused the accuracy difference. The grouped comparison separates predictive representation from storage and update latency.

Action and Drama show lower recall than several other classes in Table 4, which may reflect overlapping vocabulary, story-level weak labels, or class-specific errors. A grouped confusion matrix is required before attributing these differences to narrative overlap.

The platform supplies a primary genre at story level, while the classifier operates on chapters. The experiment therefore uses the story label as weak supervision for every chapter, even though individual chapters may contain different or mixed genre signals.

Tables 5 and 6 illustrate chapter-level score vectors for two selected stories with five chapters each. The columns are the six genre scores; the maximum in each row determines the displayed chapter label. These examples demonstrate the visualization format but do not constitute a randomized or representative evaluation of temporal genre change.

Table 5: *Naïve Bayes Classification Results for a Random Fiction Entry (No. 49 - Fantasy)*

Chapter	Horror	Suspense	Fantasy	Sci-Fi	Action	Drama
1	0.04	0.04	0.72	0.01	0.09	0.10
2	0.03	0.05	0.69	0.02	0.08	0.13
3	0.01	0.05	0.71	0.01	0.10	0.12
4	0.04	0.06	0.74	0.00	0.06	0.10
5	0.02	0.07	0.69	0.01	0.14	0.07

Table 6: *Naïve Bayes Classification Results for a Random Fiction Entry (No. 53 - Drama)*

Chapter	Horror	Suspense	Fantasy	Sci-Fi	Action	Drama
1	0.02	0.04	0.16	0.02	0.08	0.69
2	0.01	0.05	0.18	0.01	0.34	0.41
3	0.02	0.06	0.18	0.02	0.36	0.35
4	0.01	0.04	0.21	0.01	0.44	0.30
5	0.02	0.03	0.09	0.01	0.18	0.67

Table 7 preserves the timing values reported for the legacy implementation. The baseline update time was not recorded, so the table does not claim a quantitative update-speed ratio.

Table 7: Legacy Retrieval and Update Timing Values

Legacy system	Representation	Retrieval (s)	Update (s)

Baseline	Corpus TF-IDF recomputation	846.700	Not recorded
V2 cached	Stored TF-IDF features	0.200	0.210
V2 cached	Stored TF counts	0.184	0.210
V2 cached	Stored IDF weights	0.188	0.210

Table 7 reports legacy point estimates of 846.700 seconds for baseline corpus-level TF-IDF recomputation and 0.184-0.200 seconds for cached retrieval, with a reported cached-update time of 0.210 seconds. The operation boundaries, platform, warm-up, repetition count, and variance were not recorded. These values motivate a reproducible benchmark but are not evidence for a speedup factor.

The legacy implementation reports a 0.24-second removal operation. The revised evaluation first tests correctness through Learn-Unlearn state equivalence, regression predictions, Relabel, Restore, duplicate events, and transaction failure. Latency is reported only for operations that pass these invariants.

6. CONCLUSION

This research extension separates Fiction Gene into a fixed batch benchmark and an auditable online count model. The batch path evaluates lexical genre classification; the online path supports revisioned Learn, Unlearn, Relabel, and Restore operations through additive sufficient statistics. This separation clarifies which results concern predictive quality and which concern operational cost.

The legacy experiment reports 96.8% TF-IDF accuracy under chapter-level cross-validation. Because chapters from one story may cross folds, the result should be read as evidence of lexical signal in the corpus rather than a final estimate for unseen stories. The proposed grouped evaluation, calibration measures, and uncertainty estimates provide the basis for the final comparison.

Persisted feature statistics eliminate repeated corpus scans in the reported implementation and enable localized count updates. The current timing values require reproduction under a documented benchmark before a speedup factor is claimed.

For the online model, update correctness follows from applying the exact stored contribution within a transaction. Round-trip state and prediction tests are necessary to show that an Unlearn operation removes the intended influence without corrupting other model statistics.

Future work should complete story-grouped evaluation, assess probability calibration, measure concept drift, and compare the online representation with fixed TF-IDF and linear baselines. Semantic representations may be evaluated separately, but they require a different unlearning method.

The resulting design supports a traceable connection between authorized documents, model revisions, predictions, and chapter-level visualizations. Its validity depends on grouped evaluation, reproducible preprocessing, and correct event handling rather than on the visual interface alone.

7. LIMITATIONS AND THREATS TO VALIDITY

The reported classification accuracy should be interpreted with one important caveat regarding the cross-validation protocol. The 10-fold partitioning described in Section 5.2 was performed at the chapter level rather than the story (fiction) level. Because chapters belonging to the same fictional work can share story-specific vocabulary, character-driven phrasing, and setting-related terms that proper-noun filtering does not fully remove, chapters from a single story may appear in both the training and test folds. This raises a risk of information leakage that could inflate the reported accuracy, particularly for genres exhibiting near-ceiling performance (e.g., Fantasy, F1 = 0.999). We recommend re-running the evaluation with fiction-level (grouped) k-fold partitioning — confining all chapters of a given story to a single fold — before treating the reported accuracy figures as final.

A second limitation concerns attribution of the accuracy improvement over the first version of Fiction Gene [1] (91% to 96.8%). This study introduces three largely independent changes relative to [1]: an improved preprocessing pipeline (Section 3.3), a different data source (Dek-D API rather than web scraping), and the feature store / incremental learning / unlearning architecture that is the primary contribution of this work. Sections 3–5 do not isolate which of these changes drives the accuracy gain, and the feature store and incremental learning mechanisms are, by construction, latency and scalability optimizations rather than accuracy interventions — they change where TF-IDF statistics are computed and stored, not what the Naïve Bayes classifier learns. An ablation evaluating (i) the original preprocessing pipeline on the new dataset and (ii) the new preprocessing pipeline on the original dataset would allow the accuracy gain to be attributed correctly rather than implicitly credited to the systems contribution.

The processing-time results in Table 7 (846.7 s baseline retrieval, 0.2 s enhanced retrieval, 0.21 s incremental update, 0.24 s unlearning) are reported

as point estimates without the hardware configuration, number of repeated trials, or variance of the measurements. Given the precision implied by these figures and the strength of the claims built on them, a description of the measurement protocol — including CPU/RAM specification, whether the database was warm or cold, and mean ± standard deviation over multiple runs — is necessary for the results to be independently reproducible.

Because both the incremental learning and unlearning mechanisms operate on cumulative, additive sufficient statistics (word counts per genre), the system has no mechanism for downweighting or expiring older data. This is, by design, efficient and exact (Section 3.6), but it also means the model has no defense against concept drift: as genre conventions in Thai online fiction evolve, older chapters continue to exert undiminished influence on genre probabilities indefinitely unless explicitly deleted through the unlearning path. Section 6 gestures toward an "auto-rebalancing" mechanism as future work; we note here explicitly that no decay, windowing, or recency weighting is currently implemented.

Finally, the dataset is drawn from a single platform (Dek-D) under its official genre taxonomy. Results should be understood as characterizing Dek-D's six-genre labeling convention specifically; generalization to other Thai fiction platforms, user-defined genre tags, or finer-grained and overlapping subgenre schemes has not been evaluated.

8. ACKNOWLEDGEMENT

The authors express their sincere gratitude to the fiction writers and digital content creators whose works contributed to the development and refinement of the Fiction Gene system. Their creativity and literary contributions have played a pivotal role in advancing automated fiction classification research.

We also extend our appreciation to Dek-D for providing structured access to fiction data through their official API, which has facilitated the systematic collection and organization of high-quality datasets for experimentation. Additionally, we acknowledge the PyThaiNLP development team for their contributions to Thai language processing, particularly in text tokenization, stopword filtering, and linguistic preprocessing, which have been instrumental in enhancing the accuracy and efficiency of the system.

Furthermore, we recognize the support of our colleagues and collaborators in the field of natural language processing and machine learning, whose insights and discussions have contributed to the continuous improvement of this work. Their valuable feedback and recommendations have been indispensable in shaping the system's architecture and methodological approach.

9. REFERENCES

[1] W. Wattanapornprom, C. Innarong, I. Kerdpra, C. Treesutrummas, and W. Susutti, "Fiction Gene: A Thai Genre-Based Fiction Visualization Testimonial System," in *Proc. IEEE Conf. Comput. Intell. Appl.*, 2024.

[2] J. Rennie, L. Shih, J. Teevan, and D. Karger, "Tackling the poor assumptions of Naïve Bayes text classifiers," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Washington, DC, USA, 2003, pp. 616–623.

[3] Z. Chen, L. Huang, and Y. L. Murphey, "Incremental Learning for Text Document Classification," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Orlando, FL, USA, 2007, pp. 2015–2020.

[4] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *Proc. IEEE Symp. Security Privacy*, San Jose, CA, USA, 2015, pp. 463–480.

[5] M. Sarvmaili, H. Sajjad, and G. Wu, "Towards Understanding the Feasibility of Machine Unlearning," *arXiv preprint arXiv:2410.03043*, Oct. 2024. [Online]. Available: https://arxiv.org/abs/2410.03043

[6] M. Stonebraker et al., "C-Store: A Column-oriented DBMS," in *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, Trondheim, Norway, 2005, pp. 553–564.

[7] L. Orr, A. Sanyal, X. Ling, K. Goel, and M. Leszczynski, "Managing ML Pipelines: Feature Stores and the Coming Wave of Embedding Ecosystems," *arXiv preprint arXiv:2108.05053*, Aug. 2021. [Online]. Available: https://arxiv.org/abs/2108.05053

[8] J. T. Pintas, L. A. F. Fernandes, and A. C. B. Garcia, "Feature selection methods for text classification: a systematic literature review," *Artif. Intell. Rev.*, vol. 54, no. 8, pp. 6149–6200, 2021.

[9] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," *Proc. SIAM Int. Conf. Data Min.*, 2007, pp. 443–448.

[10] A. Ginart, M. Y. Guan, G. Valiant, and J. Zou, "Making AI Forget You: Data Deletion in Machine Learning," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2019, pp. 3518–3531.

[11] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *Machine Learning: ECML-98*, C. Nédellec and C. Rouveirol, Eds., Lecture Notes in Computer Science, vol. 1398, Springer, Berlin, Heidelberg, 1998, pp. 137–142.

[12] A. McCallum and K. Nigam, "A comparison of event models for Naïve Bayes text classification," in *AAAI Technical Report WS-98-05*, 1998, pp. 41–48.

[13] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, and L. E. Barnes, "RMDL: Random Multimodel Deep Learning for Classification," in *Proceedings of the 2nd International Conference on Information System and Data Mining (ICISDM)*, Lakeland, FL, USA, 2018, pp. 19–28.

[14] A. Rao and N. Spasojevic, "Actionable and Political Text Classification using Word Embeddings and LSTM," *arXiv preprint arXiv:1607.02501*, 2016.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2019

[16] T. Nomponkrang and C. Sanrach, "The Comparison of Algorithms for Thai-Sentence Classification," *International Journal of Information and Education Technology*, vol. 6, no. 10, pp. 801–808, 2016.

[17] S. Kiatpanicharoen, P. Chansanam, and P. Chansanam, "Experiments of Supervised Learning and Semi-Supervised Learning in Thai Financial News Sentiment: A Comparative Study," in *Proceedings of the 2023 International Conference on Artificial Intelligence and Education (ICAIE)*, 2023, pp. 1–5.

[18] P. Pugsee and M. Niyomvanich, "Sentiment analysis of food recipe comments," *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 9, pp. 182–193, 2015.

[19] P. Vateekul and T. Koomsubha, "A study of sentiment analysis using deep learning techniques on Thai Twitter data," in *Proceedings of the International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2016, pp. 1–6.

[20] S. Klaithin and C. Haruechaiyasak, "Traffic information extraction and classification from Thai Twitter," in *Proceedings of the International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2016, pp. 1–6.

[21] N. Chaoprasit and S. Lekcharoen, "Development of Thai language profanity investigation model for online media using data mining technique," in *Proceedings of the 12th RSU National Graduate Research Conference*, 2017, pp. 1432–1441.

[22] A. Bifet, "Adaptive Learning and Mining for Data Streams and Frequent Patterns," Ph.D. dissertation, Dept. of Computer Science, Universitat Politècnica de Catalunya, Barcelona, Spain, 2009.

[23] M. Scholz and R. Klinkenberg, "Boosting classifiers for drifting concepts," *Intelligent Data Analysis*, vol. 11, no. 1, pp. 3–28, 2007.

[24] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[25] A. Li et al., "Managed Geo-Distributed Feature Store: Architecture and System Design," *arXiv preprint arXiv:2305.20077*, May 2023.

[26] M. Xiao et al., "Traceable Group-Wise Self-Optimizing Feature Transformation Learning: A Dual Optimization Perspective," *arXiv preprint arXiv:2306.16893*, Jun. 2023.

[27] Y. Yao, X. Xu, and Y. Liu, "Large Language Model Unlearning," arXiv preprint arXiv:2310.10683, 2023. [Online]. Available: https://arxiv.org/abs/2310.10683

[28] J. Seo, D. Kim, and B. Han, "Revisiting Machine Unlearning with Dimensional Alignment," arXiv preprint arXiv:2407.17710, 2024. [Online]. Available: https://arxiv.org/abs/2407.17710