Enhancing Review Classification via LLM-Based Data Annotation and Multi-Perspective Feature Representation Learning

Jiangping Huang[a], Bochen Yi[a], Weisong Sun[b], Bangrui Wan[a], Yang Xu[c], Yebo Feng[b], Wenguang Ye[a] and Qinjun Qiu[d]

[a]School of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China
[b]College of Computing and Data Science, Nanyang Technological University, Singapore, 639798, Singapore
[c]School of Information Science and Engineering, Shandong Normal University, Jinan, 250014, China
[d]School of Computer Science, China University of Geosciences, Wuhan, 430074, China

ARTICLE INFO

Keywords:
Review Classification
Vote-Based Annotation
Pretrained Language Models
Large Language Models
Multi-Perspective Feature Representation

ABSTRACT

Automatic classification of mobile application reviews (review classification, for short) can help analysts identify valuable information such as requirements, bugs, and user satisfaction, thereby shortening the cycle from user feedback to software updates. However, current review classification faces two significant challenges: the scarcity of large-scale annotated datasets and the lack of algorithms specifically designed for review classification. To address these challenges, we propose a novel approach to constructing mobile application review datasets by leveraging multiple large language models (LLMs) for joint annotation. This approach accelerates dataset construction and reduces the cost of annotating corpora. Based on this approach, we create a **M**ulti-model **J**oint **A**nnotation **R**eviews (MJAR) dataset, consisting of 18,007 automatically labeled samples and 2,424 manually labeled samples, which alleviates the issue of small-scale annotated datasets in review classification. Additionally, we introduce a **M**ulti-**P**erspective **F**eature **R**epresentation (MPFR) learning approach to capture different perspective-based features for review classification. MPFR utilizes part-of-speech (POS) tagging, dependency structure analysis, and word embeddings as multi-perspective feature representations. By extracting feature embeddings from cross-attention layers and learning structural information through graph convolutional networks (GCNs), MPFR enhances the semantic representation of application reviews. The experimental results demonstrate that MJAR significantly improves the mining performance of mobile application reviews, while MPFR effectively learns multi-perspective features for classifying user reviews, outperforming baseline methods.

1. Introduction

The app stores (e.g., Google Play and App Store) are primary platforms connecting users with applications and serve as a means for users to express their opinions through reviews, as illustrated in Fig. 1. These reviews are also a crucial source of information throughout the entire software development life cycle (Maalej and Nabil, 2015; Villarroel et al., 2016; Li et al., 2021; Dąbrowski et al., 2022; Araujo et al., 2022). For instance, reviews can assist software developers in identifying and eliciting new requirements during software maintenance (Dąbrowski et al., 2020). In software repair, reviews help developers identify and resolve software bugs (Iacob et al., 2016; Shams et al., 2020), among other tasks. Therefore, automated classification of reviews can help developers quickly identify issues that need to be addressed in the next software update, attracting increasing attention from both academic and industrial communities.

Existing approaches mainly focus on manually constructing review datasets and designing supervised classification approaches. Extracting or annotating information from reviews can involve rule-based matching methods, which rely on a large number of handcrafted rules, as well as supervised learning models. Rule-based methods (Rana and Cheah, 2017; Hu et al., 2023), such as those employed by Iacob et al. (2016), utilize 237 semantic rules and an automated rule-matching algorithm to extract requirement information from user reviews. However, these methods depend on expert summarization and are often unable to cover all reviews comprehensively. In contrast, Aslam et al. (2020) proposed a CNN-based model for review classification, while Henao et al. (2021) evaluated the use of transfer learning on BERT (Devlin et al., 2019) for classifying reviews. Hadi and Fard (2023) explored multiple pretrained language models (PLMs), including RoBERTa (Liu et al., 2019), for the same task. Although deep learning-based classification methods can predict category labels for each review, they require large-scale annotated datasets for training. Alternatively, Wei et al. (2023) used ChatGPT as

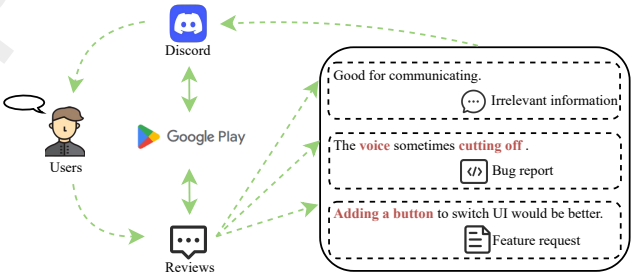


Fig. 1: The relationship among users, applications, and reviews, with examples of different review categories.

ORCID(s): 0000-0002-0288-6824 (J. Huang)

a zero-shot classifier for review classification, though this approach is limited by the generalization capabilities of LLMs in specific domains.

In current review classification methods, there are two primary challenges. On the one hand, there is a lack of large-scale datasets for model training (Devine et al., 2023), as demonstrated in Table 1. The existing datasets are insufficient in size, which significantly limits the development of review classification methods. On the other hand, current review classification methods, which are based on existing deep learning models, often rely on single-feature representation. This approach lacks multi-perspective feature analysis, potentially resulting in suboptimal performance.

Due to the complexity, subjectivity, and diversity of the annotation process, preparing an annotated dataset is both expensive and time-consuming. Researchers have explored semi-supervised learning (Deocadez et al., 2017), active learning (Deocadez et al., 2017), and combined dataset methods for review classification (Devine et al., 2023) to mitigate the issue of insufficient datasets. However, small-scale datasets are insufficient to support the effective training of high-precision models. To address the scarcity of annotated datasets in review analysis, we propose using LLMs for automatic data annotation via a label-voting method involving multiple LLMs. This label-voting approach aims to enhance annotation reliability by leveraging consensus among multiple LLMs. Using this approach, we create a **M**ulti-model **J**oint **A**nnotation **R**eviews (MJAR) dataset, consisting of 18,007 automatically labeled samples and 2,424 manually labeled samples, which alleviates the issue of small-scale annotated datasets in review classification.

In response to the high incidence of non-standard language constructs in application reviews, we propose a review classification method named **MPFR** that integrates **M**ulti-**P**erspective **F**eature **R**epresentations. MPFR converts sentences into POS and dependency views for feature extraction. The process begins with semantic parsing of sentences to obtain POS tags and dependency trees. Subsequently, MPFR utilizes PLMs-based encoders for multi-perspective feature extraction, resulting in a more comprehensive representation. To capture word relationships, MPFR employs attention matrices, while GCNs are used for feature learning. PLMs perform unsupervised learning on large corpora, enabling them to encode rich semantic information (Hey et al., 2020). Additionally, GCNs enhance classification performance by capturing the correlations between tokens (Piao et al., 2022; Dai et al., 2022).

We evaluated the performance of our proposed multi-model joint annotation dataset using several baseline models and compared it with single-model annotation. The experimental results demonstrate that the dataset annotated with LLMs enables effective training of supervised learning models, resulting in accurate label predictions. Moreover, the dataset produced through the multi-model joint annotation method consistently outperformed those annotated by a single model. We further compared the performance of our MPFR against commonly used PLMs- and GCNs-based review classification methods across two datasets. The results indicate that MPFR, which integrates both GCNs and PLMs, improved the F1 score by 1.67% and 1.73% compared to the best-performing baselines on the respective datasets.

The main contributions can be summarized as follows:

- We propose an efficient multi-model joint annotation method and construct the first large-scale dataset named MJAR for review classification, which effectively supports the training and development of high-precision review classification models.
- We introduce a novel multi-perspective feature representation method named MPFR for review classification, incorporating POS tags and dependency relations as features to enhance semantic representation.
- Experimental results on multiple baseline models confirm the effectiveness of MJAR in improving review classification model training. Additionally, evaluations across various datasets demonstrate that MPFR consistently outperforms state-of-the-art methods.

2. Related Work

2.1. Data Annotation

Data annotation aims to elucidate the relationship between samples and their corresponding labels, thereby facilitating the development and optimization of machine learning models for domain-specific tasks. Current annotation methods include expert manual annotation, bulk annotation, semi-supervised annotation, and crowd-sourced annotation. While expert manual annotation produces high-quality datasets, it faces challenges such as fairness, bias, subjectivity, high costs, time constraints, and annotation fatigue (Williams and Mahmoud, 2017; Wei et al., 2022). Bulk annotation, which relies on automated rules and templates, often results in lower-quality annotations (Govindarajan et al., 2019). Semi-supervised annotation attempts to balance the strengths of expert and bulk annotation by leveraging a small amount of labeled data to train models for broader annotation tasks (Hauer et al., 2021), but its implementation is complex and lacks generalizability. Crowd-sourcing platforms, which assign tasks to non-experts, offer the fastest and most cost-effective method for generating large-scale datasets (Pustejovsky and Stubbs, 2012). However, this approach can lead to lower-quality annotations and presents challenges in workforce management.

In the past few years, there have been a lot of studies showing that LLMs have powerful language understanding capabilities Du et al. (2024); Sun et al. (2024). Therefore, there have also been recent studies (Ahmed et al., 2024; Wang et al., 2024; Song et al., 2024) exploring the feasibility of using LLMs for data annotation tasks. Zhu et al. (2023) employed ChatGPT to emulate manual annotation across five computational social science datasets, encompassing stance detection, sentiment analysis, and bot detection. Their

Table 1
Overview of the nine classification datasets related to application (app) reviews. **Source** indicates the origin of data collection, **Size** refers to the total number of samples in each dataset, **Apps Count** represents the number of distinct apps included, and **Reviews per App** represents the average number of reviews collected per app, rounded to the nearest whole number. The MJAR (Ours) dataset contains the largest number of samples and the highest average reviews per App compared to the others.

Work	Source	Size	Apps Count	Reviews per App
Guzman et al. (2015)	Reviews	4,385	7	626
Maalej and Nabil (2015)	Reviews	1,438	48	30
Scalabrino et al. (2017)	Reviews	2,986	705	4
Scalabrino et al. (2017)	Reviews	707	14	51
Williams and Mahmoud (2017)	Twitter	3,907	10	391
Tizard et al. (2019)	Forums	2,652	2	1,326
Wei et al. (2022)	Reviews	6000	3	2000
Wei et al. (2023)	Reviews	12000	3	4000
MJAR (Ours)	Reviews	**20,431**	5	**4,086**

findings indicate that while ChatGPT can effectively perform data annotation, its efficacy varies based on the model version and prompt design. In a separate study, Hoes et al. (2023) utilized ChatGPT as a zero-shot classifier to re-annotate 12,784 manually labeled fact-checking statements, achieving 72.0% accuracy in binary classification ('true' or 'false'). Further, Huang et al. (2023) demonstrated ChatGPT's capability to correctly annotate 80% of implicit hate tweets in the LatentHatred dataset (ElSherief et al., 2021). Gilardi et al. (2023) extended the application of ChatGPT to diverse annotation tasks, including text topic identification and relevance assessment. Notably, the majority of current LLMs-based annotation tasks focus on re-annotating existing datasets, leaving the construction of new datasets as an underexplored area. This research gap presents an opportunity for innovation in dataset creation methodologies. Motivated by the demonstrated potential of LLMs in data annotation, this study seeks to explore novel strategies for constructing application review datasets with LLMs, thereby expanding the existing body of knowledge in this domain.

2.2. PLMs-based and GCNs-based Classification

PLMs have exhibited exceptional performance across a wide range of downstream tasks. BERT (Devlin et al., 2019) based on Transformer (Vaswani et al., 2017), for instance, leverages two auxiliary objectives—Next Sentence Prediction (NSP) and Masked Language Modeling (MLM)—to enhance its capacity for comprehending input sequences. RoBERTa (Liu et al., 2019) refines this model by removing the NSP objective, and further improves performance through extended training on larger batches and a more extensive corpus. Additionally, DistilBERT (Sanh et al., 2019) applies knowledge distillation to compress BERT, resulting in a more efficient version of the original model.

Building on the success of GCNs in representation learning for graph-structured data, numerous studies have investigated graph-based methods to improve performance in text classification tasks (Yang et al., 2022; Piao et al., 2022; Xie et al., 2021). TextGCN (Yao et al., 2019) constructs a heterogeneous graph from both training and test documents, where word and document nodes are connected and classified using GCNs. SHINE (Wang et al., 2021) models interactions between words, POS tags, and entities by constructing a hierarchical heterogeneous graph. InductGCN (Wang et al., 2022) leverages statistical data from training documents, representing document vectors as weighted sums of word vectors and employing unidirectional GCNs propagation during testing. TextFCG (Wang et al., 2023) constructs individual word graphs for each text, using labeled edges to integrate contextual relationships and multiple edge types to enhance graph structure and improve GCNs learning efficiency. This study proposes a multi-perspective feature representation method that fine-tunes PLMs using annotated data, enabling the models to effectively learn attention distributions across tokens while employing GCNs to capture token relationships. Leveraging the integration of multi-perspective feature information, the proposed method significantly enhances performance in review classification.

3. Annotated Dataset Construction

The dataset construction process is divided into two distinct phases. In the first phase, we manually annotated the data, and the ensuing high-quality annotations were used to validate the performance of LLMs and served as the test set for evaluating trained supervised learning models. The second phase involved generating the training set through collaborative annotation by multiple LLMs, which was subsequently used to train the review classification models. A total of 2,854 reliably annotated instances were collected during the manual annotation phase, while 18,007 instances were gathered during the multi-model annotation phase. The overall dataset construction process is depicted in Fig. 2.

3.1. Data Preprocess

We collected 41,205 raw reviews from five applications (apps) via the Google Play Store API[1]: Discord, GroupMe, Skype, TextFree, and TextNow. The statistics for these apps are presented in Table 2. These applications have garnered a

[1] https://play.google.com/

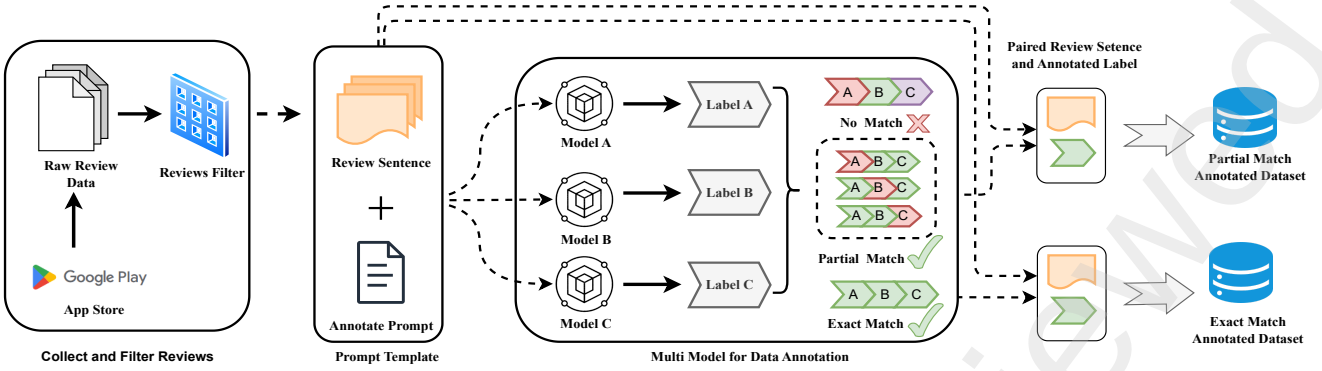


Fig. 2: Workflow of dataset construction, from data collection to consistency annotation. The process begins with data collection and filtering to generate the raw set of review sentences. The next step involves constructing prompt templates, followed by multi-model joint annotation, which results in both fully consistent and partially consistent annotations.

Table 2
Statistics of star ratings, downloads, and review counts for five apps. "M" denotes million, "B" denotes billion, and "+" signifies more. All statistical data are from Google Play.

App Name	Star Rating	Downloads	Reviews
Skype	4.3	1B	11.65M
Discord	3.2	100M+	5.88M
TextNow	4.5	100M+	1.42M
GroupMe	4.6	10M+	0.585M
TextFree	4.4	10M+	0.545M

substantial number of downloads and user reviews, offering a robust dataset for conducting research on apps reviews[2].

During data collection, we observed that the raw data included reviews in multiple languages, such as English, Chinese, French, and others. We employed the langdetect[3] tool to filter out non-English reviews, enabling us to focus exclusively on English reviews.

We identified duplicate reviews within the dataset and subsequently employed two distinct methods to eliminate them during the data processing phase. The first method targeted lexical similarity using edit distance-based deduplication. Edit distance quantifies the similarity between two strings by measuring the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another. This approach enabled the elimination of entries with high character-level similarity. The second method addressed semantic duplication by utilizing a paraphrase model-based deduplication technique. We leveraged the work of Gao et al. (2021) on contrastive learning (CL) for model training[4]. This model computes text representations and determines reviews similarity via cosine similarity. Review pairs exhibiting a similarity score exceeding 0.95 were classified as paraphrases. From each set of paraphrases, we retained one randomly selected review and removed the others. These deduplication procedures ensured lexical and semantic diversity in the dataset, mitigating the risk of overfitting during the subsequent training phase. Following language filtering and text deduplication, the final dataset for model annotation comprised 24,800 samples. Figure 3 illustrates the distribution of word counts in this refined dataset.

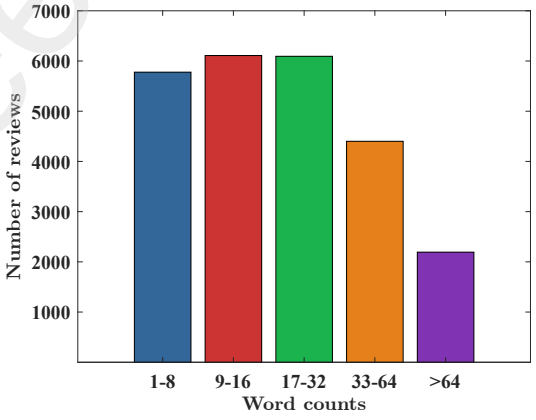


Fig. 3: Distribution of word counts in the collected reviews, with the majority containing 64 words or fewer.

3.2. Gold Standard Dataset Construction

To maintain consistency and accuracy in the data annotation process, it is imperative to establish a well-defined labeling guideline. This guideline not only delineates each data category with precision but also provides illustrative examples to serve as a reference for annotators throughout the process. Reviews are classified as either informational or non-informational. Informational reviews pertain directly to Software Engineering (SE) and are further subdivided into categories such as Bug Report and Feature Request. Non-informational reviews encompass expressions of appreciation, user experience feedback, and other commentary unrelated to software modifications. Building upon the categorization method proposed by Maalej and Nabil (2015), we refined this schema to focus on three key categories within the software domain: Feature Request (FeaREQ), Bug Report

[2]The reviews were collected as of March 2024.
[3]https://github.com/Mimino666/langdetect
[4]https://huggingface.co/princeton-nlp/sup-simcse-roberta-base

Table 3
Explanations of application review categories: Bug Report, Feature Request, and Irrelevant Information.

Category	Explanation
Feature Request	A structured request applied to software maintenance that implements a new function or enhances existing features.
Bug Report	The failure of the original software functionality.
Irrelevant Information	The information that does not contain content that can be applied to software updates.

(BugREP), and Irrelevant Information (IrrINF). Detailed descriptions of these categories are provided in Table 3. This refined categorization framework is designed to address the practical needs of software development teams, particularly in optimizing the identification and prioritization of user feedback for iterative software development.

To ensure the reliability and accuracy of the data annotation process, three annotators were carefully selected, all of whom were master's students with specialized knowledge in SE. Their academic background and familiarity with software development practices made them well-suited for this task. A rigorous training program was implemented to familiarize the annotators with the established labeling guidelines and categories. To assess their initial proficiency, a carefully curated set of 200 test annotations, rigorously validated by domain experts, was employed. The preliminary evaluation yielded annotation accuracies of 92%, 92.5%, and 95% for the three annotators, respectively, indicating the high reliability of their work. Upon completing the annotation task, a comprehensive review was conducted, during which instances of ambiguity or inconsistency were identified and resolved through expert discussions to achieve consensus. To quantitatively measure inter-annotator agreement, Fleiss' kappa was calculated, yielding a score of 0.877. This high kappa value underscores the dataset's reliability, making it well-suited for subsequent testing.

Through our manual annotation process, we curated a high-quality dataset consisting of 2,854 annotated reviews. This dataset was strategically partitioned to serve various purposes in this study. Specifically, 2,424 reviews were allocated to the test set, providing a robust benchmark for evaluating review classification tasks. Additionally, 370 samples were reserved for validating the effectiveness of few-shot prompt-based annotation, while a subset of 60 samples was employed to construct the few-shot prompt templates.

3.3. Multi-Model Joint Data Annotation

We selected three representative 7B-sized LLMs for this study, balancing performance and resource consumption: Llama3-8B[5], released by Meta in 2024 (AI@Meta, 2024); Gemma-7B[6], released by Google in 2024 (Team et al., 2024); and Mistral-7B[7], developed by Stability-AI and released in 2023 (Jiang et al., 2023). These models have demonstrated strong performance across various benchmark evaluations, particularly in handling complex tasks and showcasing advanced contextual understanding.

[5]https://huggingface.co/meta-llama/Meta-Llama-3-8B
[6]https://huggingface.co/google/gemma-7b
[7]https://huggingface.co/mistralai/Mistral-7B-v0.1

Prompt Template

We perform text generation using a text completion approach. The prompt template is structured in four parts, as shown in Fig 4. First, the role of the LLMs is defined. Next, the specific task assigned to the LLMs is clarified. Third, the annotation categories are outlined to constrain the range of possible generated labels. Finally, the desired output format of the model is predetermined. By utilizing this prompt template, we efficiently collect review category labels through rule-based methods, such as regular expressions.

Role: Assuming you are a software product analyst.
Task: You are categorizing user reviews into three categories: bug report, feature request, and irrelevant information.
Category Definition:

- **bug report:** Refers to the failure of the original software functionality.
- **feature request:** A structured request applied to software maintenance that implements a new function or enhances existing features.
- **irrelevant information:** Refers to information that does not contain content that can be applied to software updates.

Output Format: The output format should be in the form of "review → category".

Fig. 4: Prompt template for review data annotation.

Few-Shot Prompt Validation for Data Annotation

During the process of utilizing LLMs for data annotation, we employed few-shot prompting by integrating annotated examples into the prompt template. As in manual annotation, a small subset of data was used to validate annotation performance before conducting formal annotations. To determine the optimal number of examples to include in the prompts, we analyzed the relationship between the number of examples and the model's annotation accuracy. In the validation experiments, a total of 370 samples were used. As shown in Fig. 5, the results indicate that annotation accuracy significantly improves with the addition of examples, but stabilizes when the number reaches 10. This finding suggests that incorporating examples enhances the model's contextual understanding, leading to more accurate annotations. However, the annotation capability of LLMs appears to have an upper limit, making it impractical to continually add examples for further performance gains. For subsequent annotation tasks, a 10-shot prompt setup was adopted. Fig. 5 also shows that the three LLMs achieved over 75% accuracy during testing, demonstrating their capacity for data annotation. Based on these experiments, we conclude that while the annotation capabilities of the three

Table 4
Statistics of training and validation data for Exact Match (EM) and Partial Match (PM), with EPM representing the combined total of EM and PM.

Dataset	Train Samples	Valid Samples
EM	16206	1801
PM	4920	547
EPM	21126	2348

LLMs plateaued after using 10-shot prompts, the Gemma model exhibited greater sensitivity to sample size, achieving 75% accuracy with just a one-shot setup. However, as the number of examples increased, the performance gap between the models progressively diminished.

During the few-shot prompt annotation experiments, we observed that there is a bias: when the sample categories in the prompts were imbalanced, the LLMs exhibited a tendency to favor certain labels. To mitigate this issue, it is crucial to provide an equal number of example samples for each category. Additionally, during the annotation process, we found that a small subset of reviews could not be annotated by the models using the predefined labels, resulting in invalid annotations. Specifically, the Llama3 model successfully annotated 24,430 data points, the Gemma model 24,221, and the Mistral model 24,554.

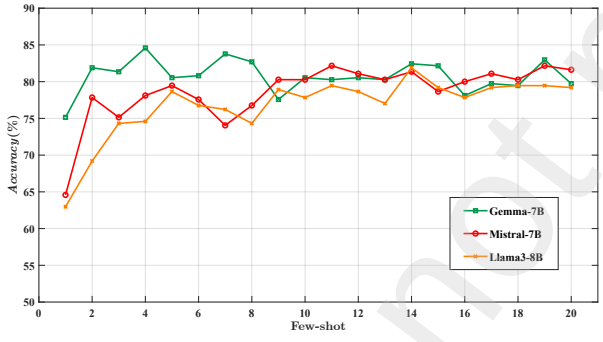


Fig. 5: Annotation accuracy across Llama3, Gemma, and Mistral with different few-shot prompts. After the 10-shot setup, differences in annotation accuracy among the three models diminished, and accuracy fluctuations stabilized.

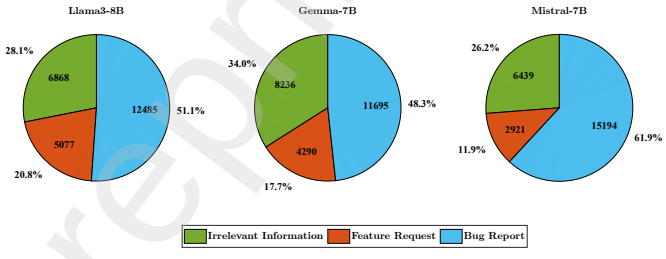


Fig. 6: Label distribution in datasets annotated by Llama3, Gemma, and Mistral.

We conducted a statistical analysis of the annotation results produced by different models. Fig. 6 presents the

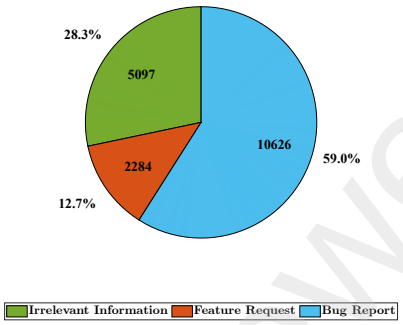


Fig. 7: Label distribution of the MJAR dataset.

label distribution statistics from the annotations generated by various LLMs. The data indicate that, in the collected raw reviews, 'Bug Report' constitutes the largest proportion, while 'Feature Request' accounts for the smallest. Although performance differences were observed among the three models in the annotation tasks, their understanding and interpretation of labels varied. Consequently, relying on the annotation results from a single model may introduce bias, which could further propagate into prediction bias during model training and testing.

In manual annotation, discussions are used to resolve inconsistent annotations from different annotators and to determine the final label for a review. However, resolving such inconsistencies in multi-model annotation is more complex, as LLMs only perform annotations based on predefined strategies and lack a decision-making mechanism. To address label inconsistencies between the outputs of different LLMs, we apply the absolute majority voting rule. Let M be the set of LLMs, where $|M| = n$ represents the total number of LLMs, and let L be the set of possible labels, with $l_k \in L$ representing a specific label. Furthermore, let m denote the number of possible labels and N_k the number of votes for label l_k. The absolute majority rule is defined as:

$$\exists k \in \{1, 2, \ldots, m\}, \quad N_k > \frac{n}{2}.$$

If N_k, the number of votes for l_k, exceeds half of the total number of models $\frac{n}{2}$, label l_k is selected as the final label for the review. The annotated reviews fall into three categories. The **exact match category** (EM) is defined as:

$$EM = \{\forall j, h \in \{1, 2, \ldots, n\}, V_j = V_h\},$$

where V_j and V_h represent the labels assigned by annotators j and h, respectively, indicating complete agreement on the label for a given review. The **partial match category** (PM) is defined as:

$$PM = \{\exists k \in \{1, 2, \ldots, m\}, N_k > \frac{n}{2}, \\ \exists j \in \{1, 2, \ldots, n\}, V_j \neq k\},$$

which indicates that the majority of annotators agree on a label, though not all. The **non-match category** (NM) is defined as:

$$NM = \{\forall k \in \{1, 2, \ldots, m\}, N_k \leq \frac{n}{2}\},$$

Table 5
Experimental results (%) of BERT and RoBERTa on the EM, PM, and EPM datasets. The table reports *Acc*, *Prec*, *Rec*, and F_1 scores for each model across three label categories: BugREP, InrINF, and FeaREQ. **Note**: The values in **bold** represent the highest scores for the corresponding metric across different models, similar to the format used in other tables.

Model		*Acc*	*Prec*			*Rec*			F_1		
			BugREP	InrINF	FeaREQ	BugREP	InrINF	FeaREQ	BugREP	InrINF	FeaREQ
BERT	EM	**80.36**	76.43	**94.42**	74.85	**95.69**	64.10	**74.23**	**84.98**	76.36	**74.54**
	PM	75.33	76.58	83.00	63.32	85.43	64.81	70.96	80.76	72.79	66.92
	EPM	**80.36**	**77.85**	88.60	**75.97**	93.40	**68.13**	72.39	84.92	**77.03**	74.14
RoBERTa	EM	**80.61**	76.55	**93.78**	**75.65**	**96.06**	66.11	**71.17**	**85.2**	**77.55**	**73.34**
	PM	78.54	77.00	85.99	72.17	91.75	67.65	67.89	83.73	75.73	69.97
	EPM	79.41	**79.63**	84.09	72.05	90.28	**70.14**	**71.17**	84.62	76.49	71.60

Table 6
Experimental results (%) of BERT and RoBERTa on the MJAR, Llama3, Gemma, and Mistral datasets. The MJAR dataset showed improvements across multiple metrics for both models. **Note**: Arrows (↑) and (↓) indicate improvements or declines in metrics, respectively. An underline denotes the best result within the control group, similar to the format used in other tables.

Dataset	BERT				RoBERTa			
	Acc	*Prec*	*Rec*	F_1	*Acc*	*Prec*	*Rec*	F_1
Mistral	75.04	78.21	70.98	72.56	77.26	79.46	73.48	74.88
Gemma	77.84	76.61	75.74	75.84	79.49	78.03	77.82	77.65
Llama3	79.53	79.76	76.93	77.48	79.20	80.15	**78.16**	77.19
MJAR	$\mathbf{80.36}_{(0.83)}\uparrow$	$\mathbf{81.89}_{(2.13)}\uparrow$	$\mathbf{78.00}_{(1.07)}\uparrow$	$\mathbf{78.62}_{(1.14)}\uparrow$	$\mathbf{80.61}_{(1.12)}\uparrow$	$\mathbf{81.99}_{(1.84)}\uparrow$	$77.78_{(0.38)}\downarrow$	$\mathbf{78.69}_{(1.04)}\uparrow$

which reflects that no label has received a majority of votes. We also construct a combined dataset, denoted as $EPM = EM + PM$, which merges the EM and PM categories.

Table 4 presents the statistics for the three categories of data, with the training and validation sets randomly split into 90% and 10%, respectively. We evaluated the performance of baseline models across the three datasets using *Accuracy (Acc)*, *Precision (Prec)*, *Recall (Rec)*, and F_1 score, as defined in Sec 5.1. The results are summarized in Table 5. The findings indicate that the EM dataset outperformed both the PM and EPM datasets. Consequently, we selected the EM dataset, referred to as MJAR, for further use. The label distribution of the MJAR dataset is illustrated in Fig. 7. We conducted a comparative analysis of datasets annotated by different models to evaluate their performance on the test set. As shown in Table 6, datasets annotated by a single model exhibited suboptimal performance. In contrast, the MJAR dataset, annotated using multi-model voting, demonstrated greater reliability and validity despite its smaller size, resulting in improved model training outcomes.

4. Methodology

The proposed approach first parses the input review into POS tags and dependency trees using a POS tagger and a dependency parser. The resulting tags, trees, and corresponding words are subsequently input into the designed model to train a supervised review classifier. The overall framework of the proposed method is illustrated in Fig. 8, while the MPFR algorithm is detailed in Algorithm 1.

4.1. Semantic Analysis of Review Data

We utilized the Spacy[8] library to extract explicit multi-perspective features from the review texts, such as POS tags and dependency trees. An example of this analysis is presented in Table 7. Given a review or sentence S, a word segmenter first segments S into multiple words, represented as $S_w = \{w_1, w_2, \ldots, w_n\}$, where n is the number of words in S_w. Subsequently, a POS tagger assigns corresponding POS tags to each word in S_w, resulting in $S_p = \{p_1, p_2, \ldots, p_n\}$. Finally, a dependency parser generates the corresponding parse tree for S, represented as $S_d = dep(w_i) \cdot w_i \cdot \left(\bigcup_{c \in children(w_i)} T(c)\right)$, where $dep(\cdot)$ denotes the dependency label of each word, $children(\cdot)$ refers to the set of child nodes for the current word w_i, and $T(c)$ represents the subtree structure rooted at a child node c of the word w_i, with $\left(\bigcup_{c \in children(w_i)} T(c)\right)$ representing the recursive application of the function to all child nodes.

4.2. Multi-Perspective Feature Representation

We employed three PLMs encoders to learn multi-perspective features from the given review, specifically focusing on words, POS tags, and dependency trees. Using the RoBERTa encoder E, we encoded the embeddings for words, POS tags, and dependency trees with their respective encoders $\{E_w, E_p, E_d\}$. The review was subsequently mapped into a vector feature space as follows:

$$H_w = E_w(S_w) = \{h_{w,1}, h_{w,2}, \ldots, h_{w,n}\} \quad (1)$$

[8] https://spacy.io/

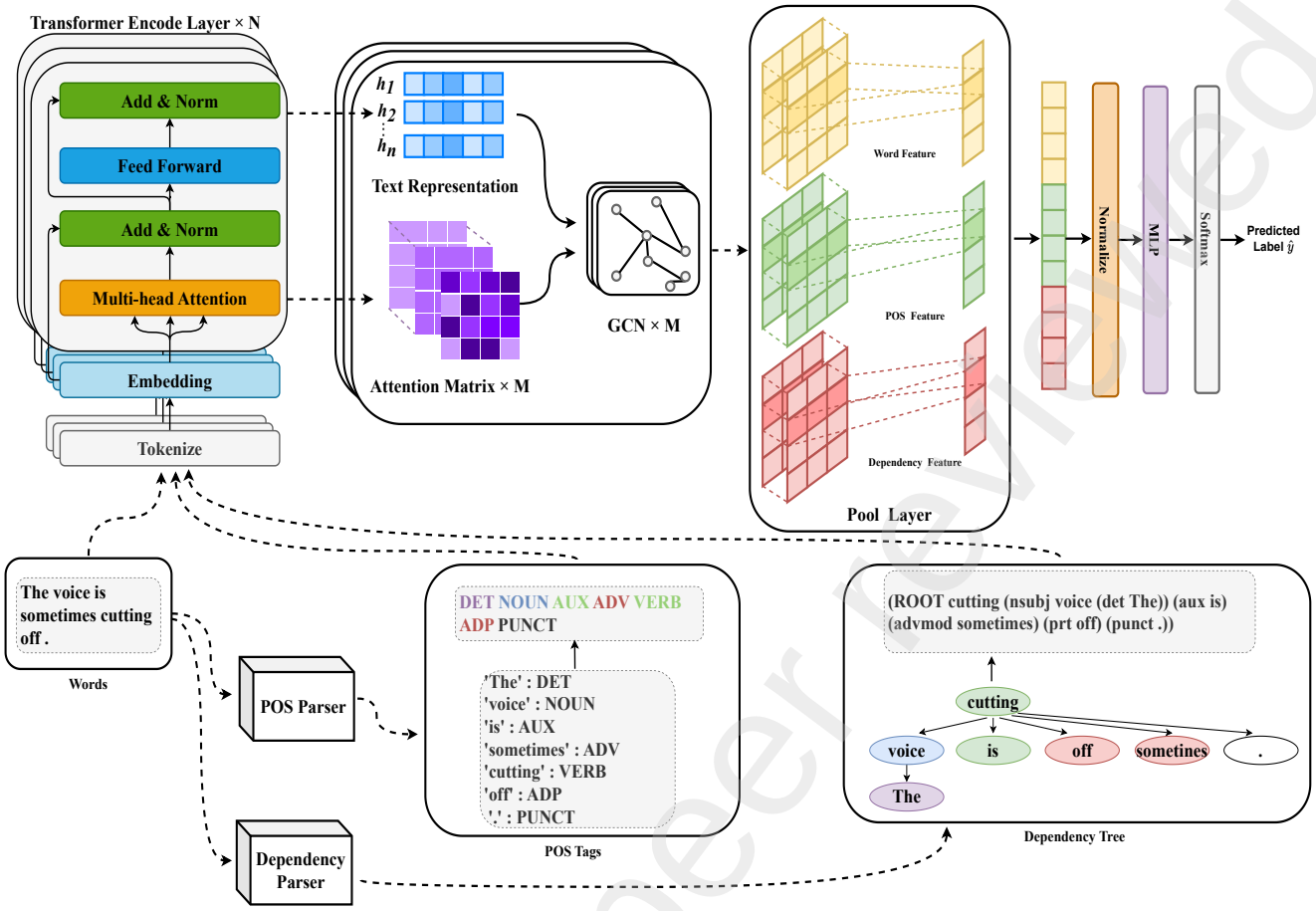


Fig. 8: The framework of MPFR. The model operates in two stages: first, semantic analysis tools generate POS tags and a dependency tree; second, multi-perspective features are extracted from the words, POS tags, and dependency tree, which are then fused to represent the review for label prediction.

Table 7
An example of text analysis: review, POS tags, and dependency tree.

Type	Example
Review	The voice is sometimes cutting off.
POS Tags	DET NOUN AUX ADV VERB ADP PUNCT
Dependency Tree	(ROOT cutting (nsubj voice (det The)) (aux is) (advmod sometimes) (prt off) (punct .))

where E_w transforms S_w into word feature vectors H_w. Similarly, the POS encoder converts S_p into POS feature vectors H_p:

$$H_p = E_p(S_p) = \{h_{p,1}, h_{p,2}, \ldots, h_{p,n}\} \tag{2}$$

where $h_{p,i}$ represents the i-th POS feature vector. The dependency encoder converts S_d into dependency feature vectors H_d:

$$H_d = E_d(S_d) = \{h_{d,1}, h_{d,2}, \ldots, h_{d,n}\} \tag{3}$$

where $h_{d,i}$ represents the i-th dependency feature vector. Additionally, $h_i \in \mathbb{R}^M$ and $H \in \mathbb{R}^{L \times M}$, where M denotes the dimensionality of the feature vectors, and L representing the length of the given review.

After obtaining the feature vectors from the encoders, attention matrices A for each encoding layer are derived from the PLMs. Specifically, $A = \{A_{i,j} \mid j \in N, i \in \{w, p, d\}\}$, where $A_{i,j} \in \mathbb{R}^{L \times L}$, N represents the number of attention matrices output by the encoding layers, and L is the length of the given review. We utilize the N attention matrices from the final layer.

The GCNs is applied to model the structural relationships between words by leveraging the attention matrices and using the feature vectors of each word as node features. GCNs operations are performed on each attention matrix, $T = \{T_{i,j} \mid j \in N, i \in \{w, p, d\}\}$. The operation is defined as follows:

$$T_{i,j} = A_{i,j} H_i W_{i,j} + b_{i,j} \tag{4}$$

where A represents the attention matrices, H represents the feature vectors obtained from the encoder, $H = \{H_i \mid i \in \{w, p, d\}\}$, $W \in R^{M \times K}$ denotes the GCNs weights, and b is the bias. The output T undergoes average pooling to produce Z, where $Z_i = \{Z_{i,l} \mid l \in L, i \in \{w, p, d\}\}$, $T_{i,j} \in \mathbb{R}^{L \times K}$, $Z_i \in \mathbb{R}^{L \times K}$, and $Z_{i,l} \in \mathbb{R}^k$. The equations are defined as:

$$Z_i = \text{MeanPooling}([T_{i,1}, \ldots, T_{i,N}]) \tag{5}$$

$$V_i = \text{MeanPooling}([Z_{i,1}, \ldots, Z_{i,L}]) \tag{6}$$

The resulting set of feature vectors $V_i \in \mathbb{R}^K$ is then concatenated and normalized as follows:

$$V = [V_w; V_p; V_d] \tag{7}$$

$$G = \text{Normalize}(V) \tag{8}$$

Finally, the cross-entropy loss is computed. Let M represent the total number of samples and C represent the set of all classes. The linear classifier, denoted as $\phi(\cdot)$, assigns the score $\phi_{y_p}(G_p)$ to the true class y_p of the p-th sample:

$$\mathcal{L}_{\text{ce}} = -\frac{1}{M} \sum_{p=1}^{M} \log \left(\frac{\exp(\phi_{y_p}(G_p))}{\sum_{q \in C} \exp(\phi_q(G_p))} \right) \tag{9}$$

Algorithm 1 The Proposed MPFR Algorithm

Input: Input sentence S_w, true label y, Feature Encoders E_w, E_p, E_d
Output: Predicted label $\hat{y}$
Semantic Analysis:
Perform POS tagging on S_w to obtain S_p
Perform dependency parsing on S_w to obtain S_d
Model Training:
Initialize model parameters θ
for *each training step* **do**
  Text Encoding:
  $H_w, A_w \leftarrow E_w(S_w)$
  $H_p, A_p \leftarrow E_p(S_p)$
  $H_d, A_d \leftarrow E_d(S_d)$
  for (H, A) *in* $\{(H_w, A_w), (H_p, A_p), (H_d, A_d)\}$ **do**
    Perform GCN computation on each attention matrix to obtain T
    Apply mean pooling on T to obtain Z_c
    Store Z_c in Z_{all}
  end
  Feature Fusion:
  V = Concatenate all Z_c in Z_{all}
  Use V to predict the label $\hat{y}$
  Compute the cross-entropy loss $\mathcal{L}_{ce}$ based on $\hat{y}$ and y
  Update model parameters θ
end
return $\hat{y}$

5. Experiments

5.1. Experimental Setting

Baselines: In this study, we employed two primary categories of baseline models: those based on GCNs[9] and PLMs. For GCN-based classification, we utilized the following models: SHINE[10] (Wang et al., 2021), BERT-GCN[11] (Lin et al., 2021), DGoW-GNN[12] (Abbahaddou et al., 2023), TextFCG[13] (Wang et al., 2023), and InductGCN[14] (Wang et al., 2022). For PLM-based classification, we leveraged the BERT[15] model (Devlin et al., 2019), along with RoBERTa[16] (Liu et al., 2019), and DistilBERT[17] (Sanh et al., 2019).

Datasets: We utilized two datasets for the experiments. The first dataset, MJAR, consists of 18,007 training samples and 2,424 test samples, all consistently annotated by multiple models, with 10% of the training set reserved as the validation set. The second dataset, the publicly available Twitter dataset from Williams and Mahmoud (2017), contains 3,907 manually annotated review samples. In this dataset, the data was split into training, validation, and test sets following Devine et al. (2023), with proportions of 64%, 16%, and 20%, respectively. Both user-generated content datasets were employed to evaluate the proposed approach, MPFR.

Parameters: The classification experiments were conducted on Ubuntu 22.04, using an Intel Xeon E5-2680 v4 CPU, an RTX 4090 24GB GPU, and 256GB of memory. An additional NVIDIA Quadro RTX 8000 48GB GPU was utilized for review annotation. The Adam optimizer was employed with a learning rate of 0.00001, weight decay of 0.001, a dropout rate of 0.2, and 50 epochs. An early stopping mechanism was implemented. Given the distribution of data lengths, the maximum sequence length for the pre-trained model was set to 128, and the batch size was set to 32. Text analysis was performed using SpaCy's en_core_web_lg[18].

Metrics: We employed *Accuracy (Acc)*, *Precision (Prec)*, *Recall (Rec)*, and the F_1 score to measure classification performance. These metrics are defined as follows: True Positive (TP) is the number of correctly predicted positive samples; True Negative (TN) represents the number of correctly predicted negative samples; False Positive (FP) indicates the number of negative samples incorrectly predicted as positive; and False Negative (FN) is the number of positive samples incorrectly predicted as negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

[9] We also incorporated baseline models from the broader class of Graph Neural Networks (GNNs) to enhance the comprehensiveness of the comparative analysis.

[10] https://github.com/tata1661/SHINE-EMNLP21

[11] https://github.com/ZeroRin/BertGCN

[12] https://github.com/abbahaddou/DGOW

[13] https://github.com/style507/Text-FCG

[14] https://github.com/usydnlp/InductTGCN

[15] https://huggingface.co/google-bert/bert-base-uncased

[16] https://huggingface.co/FacebookAI/roberta-base

[17] https://huggingface.co/distilbert/distilbert-base-uncased

[18] https://spacy.io/models/en#en_core_web_lg

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

Macro-averaging is used to calculate the *Precision*, *Recall*, and F_1 scores. Let Z_i represent the metric value for each class. The macro-average is calculated as follows:

$$Z_{\text{macro}} = \frac{1}{n} \sum_{i=1}^{n} Z_i \tag{14}$$

5.2. Few-Shot for Review Classification

We conducted experiments to evaluate the prediction performance of LLMs on the test set, as shown in Fig. 9. Zero-shot prompts were used for these predictions, and the results indicated that, without example samples, the LLMs struggled to effectively classify application reviews, resulting in relatively low accuracy. Among the models, Llama3 exhibited the highest accuracy in the zero-shot scenario, achieving 46.30%. As additional samples were incorporated, the performance of the LLMs improved steadily, with the Gemma model demonstrating the most significant enhancement. In the 10-shot experiments, the Gemma model achieved the highest accuracy at 81.39%, followed by the Mistral model at 79.36%, and Llama3 at 76.73%. Overall, Gemma displayed superior predictive capability on the test set, while Llama3's performance was comparatively weaker.

From the experimental results, we observed that the prediction capabilities of LLMs fluctuated with variations in the number of example samples. For instance, the accuracy of the Gemma model decreased by 7.06% from 7-shot to 9-shot, while the Mistral model experienced a 3.59% drop in accuracy between 5-shot and 7-shot. Controlling the model's output for domain-specific tasks solely by modifying the prompts proved challenging. Additionally, considerable randomness was observed in the prediction of labels for the test set using LLMs. However, increasing the data volume and expanding the semantic coverage in the training set can effectively mitigate the randomness in LLMs' annotations.

5.3. MPFR Performance Evaluation

In this section, we present the experimental results of MPFR in comparison with several existing text classification algorithms on the MJAR and Twitter datasets. The results, summarized in Table 8, show that our proposed method demonstrates varying levels of improvement in both accuracy and F_1 score across these two datasets.

On the MJAR dataset, our method achieved an F_1 score of 80.36% and an accuracy of 81.93%, representing an improvement of 1.67% in F_1 score and 1.32% in accuracy compared to the best-performing baseline model, RoBERTa, which recorded an F_1 score of 78.69% and an accuracy of 80.61%. Additionally, our method outperformed other GCNs-based models, such as SHINE, DGoW-GNN, and Induct-GCN, whose F_1 scores ranged from 67.23% to 75.84%. DistilBERT and BERT achieved F_1 scores of 77.25% and 78.62%, respectively.

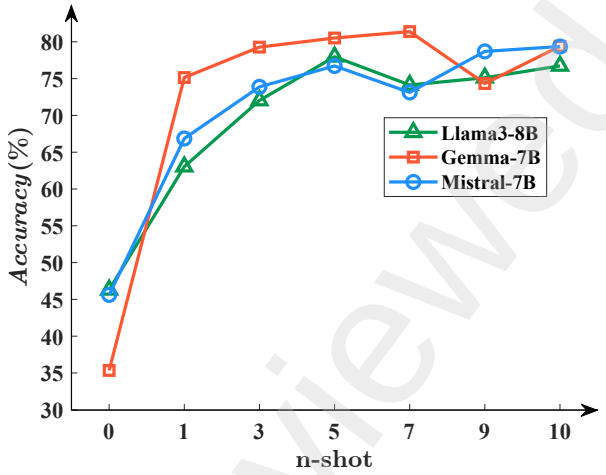


Fig. 9: Accuracy of Llama3, Gemma, and Mistral in predicting labels on the test set, varying from 0 to 10 shots.

On the Twitter dataset, our method achieved the highest F_1 score of 81.97% and an accuracy of 82.6%, outperforming the best baseline model, RoBERTa, by 1.73% in F_1 score, as RoBERTa recorded an F_1 score of 80.24%. Our method also surpassed the best-performing BERT model by 1.77% in accuracy. For the BERT-GCN method, we conducted experiments with two different backbones and found that using RoBERTa as the backbone yielded better performance on both datasets, though the results still fell short of those achieved by the PLMs.

Notably, we observed that across both datasets, GCNs-based classification methods underperformed when compared to PLMs. We attribute this to two primary reasons: First, while GCNs methods can capture word correlations, they rely on static word embeddings (e.g., Pennington et al. (2014)), which offer less effective semantic representation compared to PLMs-based classification algorithms. Second, the non-standardized syntax in application reviews further hampers the classification performance of GCNs.

Across various datasets, it is apparent that GCNs models perform better on the MJAR dataset than on the Twitter dataset, likely due to the larger dataset size, which enables the construction of more effective GCNs architectures. Our analysis suggests that the potential of GCNs is constrained in smaller datasets. Moreover, experiments on both datasets indicate that integrating multi-perspective features significantly enhances classification performance.

5.4. Ablation Study

To evaluate the contribution of different modules to the proposed method, we conducted a series of ablation experiments assessing the impact of attention matrices, POS

Table 8
Experimental Results (%) of MPFR and baseline models on the MJAR and Twitter datasets.

Model	MJAR				Twitter			
	Acc	***Prec***	***Rec***	F_1	***Acc***	***Prec***	***Rec***	F_1
SHINE	79.16	75.50	71.76	73.35	52.92	50.78	51.30	50.98
DGoW-GNN	70.98	73.06	66.71	67.23	59.24	57.47	57.15	57.29
Text-FCG	77.93	79.10	74.52	75.84	72.26	72.51	70.23	70.90
Induct-GCN	75.58	75.85	73.90	73.30	69.74	68.99	71.51	69.37
BERT	80.36	81.89	78.00	78.62	80.83	79.92	80.55	80.16
DistilBERT	79.57	81.41	75.72	77.25	80.70	80.40	79.53	79.93
RoBERTa	80.61	81.99	77.77	78.69	80.70	79.93	80.59	80.24
BERT-GCN	78.30	76.62	78.80	77.70	75.91	74.97	75.69	75.33
RoBERTa-GCN	79.21	79.72	77.32	78.50	76.29	75.36	75.87	75.61
MPFR(Ours)	**81.93**$_{(1.32)}\uparrow$	**82.08**$_{(0.09)}\uparrow$	**80.14**$_{(1.34)}\uparrow$	**80.36**$_{(1.67)}\uparrow$	**82.60**$_{(1.77)}\uparrow$	**82.2**$_{(1.80)}\uparrow$	**81.94**$_{(1.35)}\uparrow$	**81.97**$_{(1.73)}\uparrow$

features, word features, and dependency features on model performance. The notation 'w/o' denotes the exclusion of a specific module; for instance, 'w/o all' refers to the backbone PLMs without the attention matrices, while 'w/o dep' indicates the exclusion of the dependency module. The effectiveness of each module was independently validated, and the experimental results are presented in Table 9.

The experimental results indicate that incorporating attention matrices yielded improvements of varying magnitudes across the datasets, attributed to the modeling of inter-word relationships. Specifically, on the MJAR dataset, the F_1 score showed an enhancement of 0.77%, while on the Twitter dataset, a smaller improvement of 0.14% was observed. Furthermore, additional experiments were conducted utilizing POS tags and dependency relations as standalone input features. The findings demonstrate that POS features provided a meaningful representation of semantic information for text classification tasks, whereas the dependency features were less effective in capturing the relevant semantics.

Through our experiments integrating two modules, we observed notable variations in model performance. In the MJAR dataset, the absence of the word-level feature 'w/o word' yielded a marginal improvement compared to using a single module. However, in the Twitter dataset, the performance declined under similar conditions. When omitting dependency features 'w/o dep', the performance in the Twitter dataset deteriorated substantially compared to the single-module scenario, with neither F_1 score nor accuracy achieving stability. Nonetheless, combining all three features resulted in a marked performance enhancement, highlighting the advantages of multi-perspective feature integration for classification tasks.

5.5. The Effect Analysis of Attention Matrices

To verify the impact of attention matrices in capturing relationships between words, we visualized these matrices. The attention matrices illustrate the weights assigned to individual words within a review, reflecting their relative importance. By applying graph structure-weighted calculations to word vectors via attention matrices, we achieve a more accurate semantic representation by combining weighted word vectors. The pooling methods applied to the attention matrices are depicted in Fig. 10. It can be observed that, compared to max pooling, average pooling accentuates global information captured by multi-head attention through smoothing, while max pooling emphasizes locally significant features. As shown in Table 10, $\text{MPFR}_{\text{mean}}$ represents the use of mean pooling on the attention matrices, and MPFR_{max} represents the use of max pooling. The decline in performance with max pooling suggests it is less effective in capturing relevant features within the attention matrices.

5.6. The Effect of LLMs on Review Annotation

In the model selection process, we prepared a total of five models: Llama3 (AI@Meta, 2024), Gemma (Team et al., 2024), Mistral (Jiang et al., 2023), MPT[19] (Team, 2023), and Falcon[20] (Penedo et al., 2024). While constructing prompt templates and generating outputs, we observed that the MPT and Falcon models, likely due to the relatively smaller size of their training corpora, produced annotated data of significantly lower quality. To further investigate the relationship between training corpus size and the quality of annotated data, we conducted an analysis using publicly available data, as illustrated in Fig. 11. Our findings indicate that an increase in the amount of training data during LLMs training correlates with an improvement in the quality of annotated data for annotation tasks. Furthermore, we focused solely on open-source models with approximately 7B parameters and did not include comparisons with closed-source models like GPT or Claude. We plan to explore these comparisons in future research by leveraging their respective APIs.

5.7. The Effect of GCN Feature Dimensions

To validate the impact of the GCN's output feature dimension on model performance, we conducted a series of

[19] https://huggingface.co/mosaicml/mpt-7b
[20] https://huggingface.co/tiiuae/falcon-7b

Table 9
Ablation study (%) of MPFR on the MJAR and Twitter datasets. 'base' refers to the backbone model, 'word' indicates the use of only the word feature module, 'pos' refers to the POS feature module, and 'dep' refers to the dependency feature module. 'w/o' denotes variants of MPFR excluding the specified component.

	MJAR				Twitter			
	Acc	***Prec***	***Rec***	F_1	***Acc***	***Prec***	***Rec***	F_1
base	80.61	81.99	77.77	78.69	80.70	79.93	80.59	80.24
word	81.31	**83.02**	78.10	79.46	80.71	79.72	**82.34**	80.38
pos	80.36	82.09	77.28	78.48	77.93	79.42	80.22	77.86
dep	64.65	62.54	61.23	61.64	50.32	53.44	46.28	44.55
w/o word	81.72	81.67	**80.42**	80.16	77.68	77.51	80.25	77.62
w/o dep	79.66	81.32	77.02	77.97	73.52	76.38	77.55	73.99
w/o pos	81.15	81.25	78.80	79.34	79.82	79.17	81.80	79.70
MPFR	**81.93**$_{(0.21)}\uparrow$	82.08$_{(0.94)}\downarrow$	80.14$_{(0.28)}\downarrow$	**80.36**$_{(0.2)}\uparrow$	**82.6**$_{(1.89)}\uparrow$	**82.2**$_{(2.27)}\uparrow$	81.94$_{(0.4)}\downarrow$	**81.97**$_{(1.59)}\uparrow$

Table 10
Analysis of pooling operations. Experimental results (%) of various pooling operations on the MJAR and Twitter datasets. 'Max' refers to max pooling, 'k-Max' refers to k-max pooling, and 'Mean' refers to average pooling.

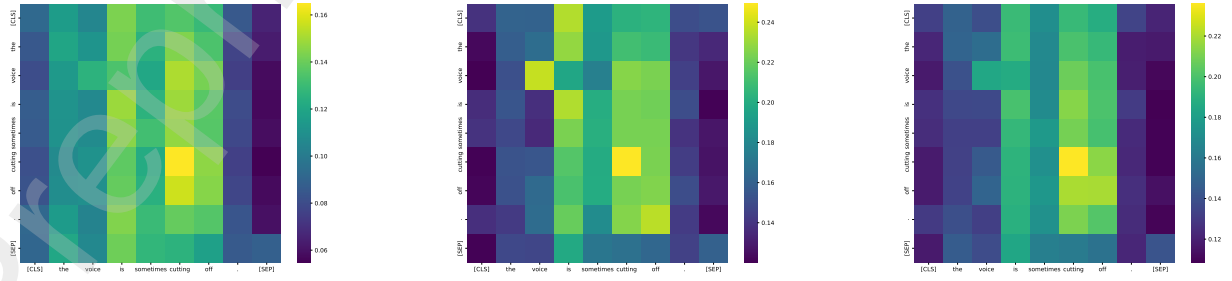
	MJAR				Twitter			
	Acc	***Prec***	***Rec***	F_1	***Acc***	***Prec***	***Rec***	F_1
MPFR_{Max}	76.11	76.87	72.91	73.64	72.64	72.73	70.07	69.81
$\text{MPFR}_{\text{2-Max}}$	79.66	81.07	77.02	77.84	80.58	79.54	81.16	80.09
$\text{MPFR}_{\text{3-Max}}$	80.86	**82.16**	77.97	79.13	82.35	81.84	81.87	81.64
$\text{MPFR}_{\text{Mean}}$	**81.93**$_{(1.07)}\uparrow$	82.08$_{(0.08)}\downarrow$	**80.14**$_{(2.17)}\uparrow$	**80.36**$_{(1.23)}\uparrow$	**82.6**$_{(0.25)}\uparrow$	**82.2**$_{(0.36)}\uparrow$	**81.94**$_{(0.07)}\uparrow$	**81.97**$_{(0.33)}\uparrow$

experiments focusing on this hyper-parameter. The results for the two datasets are illustrated in Fig. 12. We observed that the Twitter dataset exhibits greater sensitivity to this hyper-parameter compared to the MJAR dataset. Optimal performance on both datasets was achieved when the feature dimension was set to 256. Deviations from this value, whether by increasing or decreasing the dimension, led to varying degrees of performance degradation.

5.8. Case Study

To evaluate the impact of joint annotation by multiple LLMs on application review data, we analyzed cases where annotations were either consistent or inconsistent. In consistently annotated data, reviews often contained clearly identifiable keywords, such as "*will not let me log in or sign up. please fix.*" In this instance, the keyword "*fix*" clearly signals a bug report, enabling multiple models to produce consistent annotations. In contrast, ambiguous annotations, such as "*Hard to find groups that you know,*" express dissatisfaction with a feature but lack clear intention-indicating keywords. This ambiguity makes it difficult for models to assign definitive labels. During manual analysis, such vague expressions often require extensive discussion.

Using multiple LLMs for joint annotation allows diverse model interpretations to filter out content with unclear expressions. This approach not only reduces the time required for data annotation but also helps identify and eliminate samples with ambiguous language, thereby enhancing the



(a) Mean pooling (b) Max pooling (c) 3-Max pooling

Fig. 10: Visualization of attention matrices across different pooling operations.

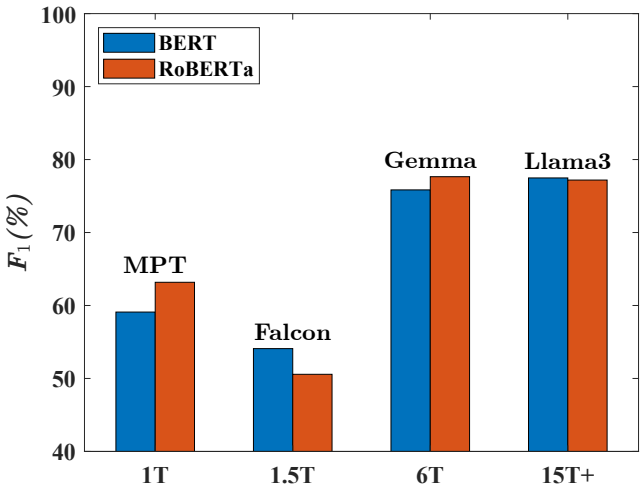


Fig. 11: The relationship between LLMs training corpus size and data annotation performance. (Note: The training corpus size for Mistral-7B has not been publicly disclosed.)

dataset's distinctive features. However, the effectiveness of this method depends on the consistency of the models' annotation capabilities. If one model demonstrates significantly weaker contextual understanding, it may incorrectly label sentences with clear intent, ultimately reducing the overall consistency of the annotated data.

5.9. Visualization of Text Representation

To further investigate the multi-perspective feature representation of the data, we visualized the feature distribution. As illustrated in Fig. 13, after training with the RoBERTa model, the data distribution for the three labels in the test set converged along three distinct directions. However, training with the MPFR model yielded a different pattern. The incorporation of features such as POS and dependency resulted in further convergence of the bug report distribution, while the feature request distribution—due to the smaller data volume—diverged. This divergence can be attributed to the increased number of features, which requires a larger dataset for distribution to stabilize. These visualizations suggest that additional data would improve review classification performance, emphasizing the critical role of dataset size. This insight highlights the importance of expanding the dataset.

6. Conclusion

In this paper, we addressed key challenges in mobile application review analysis, specifically the lack of annotated data and the need for an effective review classification approach. To address these issues, we first constructed a large-scale dataset, MJAR, using joint annotation from multiple LLMs. We then proposed a classification algorithm that integrates multi-perspective features, leveraging attention matrices from the encoder output alongside inter-word correlations. Additionally, semantic parsing was employed to extract POS tags and dependency trees, which were fused

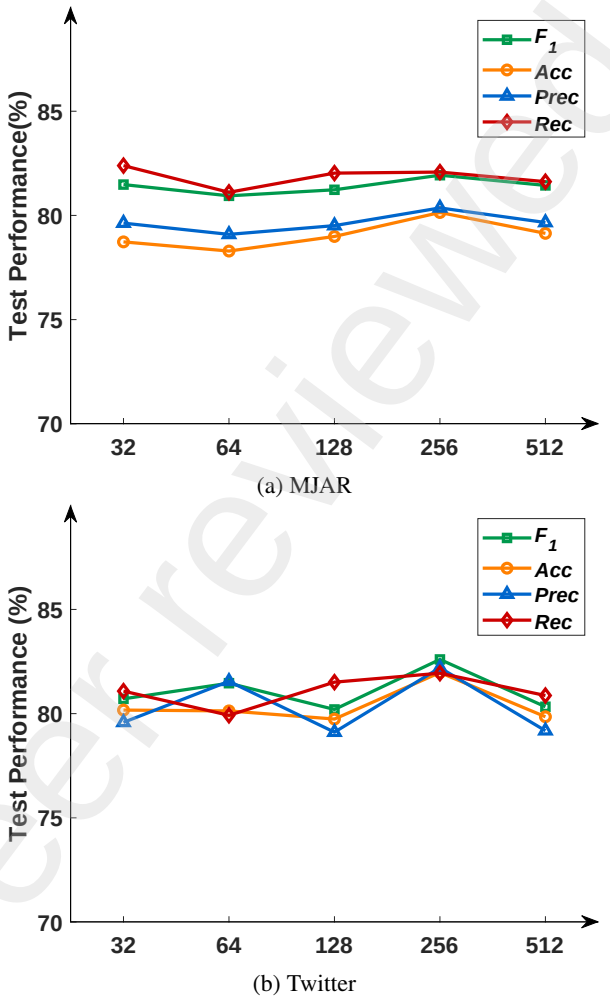
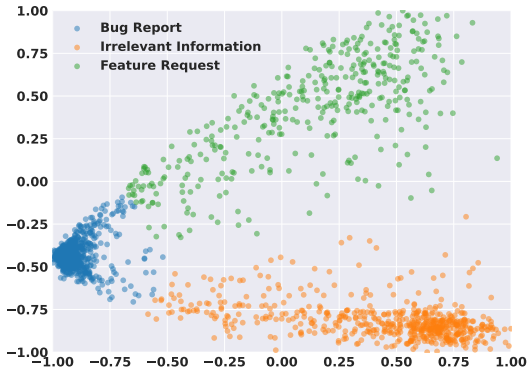


Fig. 12: The relationship between GCN feature dimension and test performance.

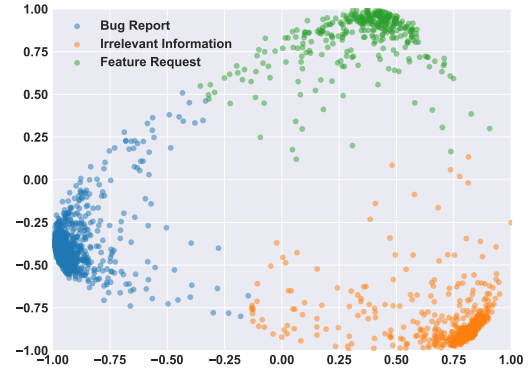
to enhance sentence-level semantic representation. We evaluated the model's performance on two datasets, comparing the results against GCNs and PLMs-based classification methods. The experimental results showed that MPFR outperformed the best-performing RoBERTa model, with improvements of 1.67% and 1.73% on the two datasets, respectively. In future work, we aim to implement more efficient data annotation methods to expand the dataset and further improve review classification performance. Furthermore, we plan to annotate more fine-grained features to guide code generation by LLMs, bridging the gap between user reviews and corresponding code outputs.

CRediT Authorship Contribution Statement

Jiangping Huang: Conceptualization, Methodology, Supervision, Writing & Editing. **Bochen Yi:** Data Curation, Validation, Writing. **Weisong Sun:** Methodology, Formal Analysis, Review. **Bangrui Wan:** Supervision, Validation. **Yang Xu:** Investigation, Resources. **Yebo Feng:** Investigation, Formal Analysis, and Review. **Wenguang Ye:** Data Curation. **Qinjun Qiu:** Formal Analysis.



(a) MPFR



(b) RoBERTa

Fig. 13: Visualization of the test data distribution after training MPFR and RoBERTa on the MJAR dataset.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

The data will be made publicly available to facilitate the reproduction of our work.

Acknowledgments

This study is supported by the Humanities and Social Sciences Fund of the Ministry of Education of China (20YJCZH047) and the Chongqing University of Posts and Telecommunications Japan Research Center (K2020-222).

References

Abbahaddou, Y., Lutzeyer, J., Vazirgiannis, M., 2023. Graph neural networks on discriminative graphs of words, in: NeurIPS New Frontiers in Graph Learning Workshop.

Ahmed, T., Devanbu, P., Treude, C., Pradel, M., 2024. Can llms replace manual annotation of software engineering artifacts? arXiv preprint arXiv:2408.05534 .

AI@Meta, 2024. Llama 3 model card URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Araujo, A.F., Gôlo, M.P., Marcacini, R.M., 2022. Opinion mining for app reviews: an analysis of textual representation and predictive models. Automated Software Engineering 29, 5.

Aslam, N., Ramay, W.Y., Xia, K., Sarwar, N., 2020. Convolutional neural network based classification of app reviews. IEEE Access 8, 185619–185628.

Dąbrowski, J., Letier, E., Perini, A., Susi, A., 2020. Mining user opinions to support requirement engineering: an empirical study, in: International Conference on Advanced Information Systems Engineering, Springer. pp. 401–416.

Dąbrowski, J., Letier, E., Perini, A., Susi, A., 2022. Analysing app reviews for software engineering: a systematic literature review. Empirical Software Engineering 27, 43.

Dai, Y., Shou, L., Gong, M., Xia, X., Kang, Z., Xu, Z., Jiang, D., 2022. Graph fusion network for text classification. Knowledge-based systems 236, 107659.

Deocadez, R., Harrison, R., Rodriguez, D., 2017. Preliminary study on applying semi-supervised learning to app store analysis, in: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, pp. 320–323.

Devine, P., Koh, Y.S., Blincoe, K., 2023. Evaluating software user feedback classifier performance on unseen apps, datasets, and metadata. Empirical Software Engineering 28, 26.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding, in: Burstein, J., Doran, C., Solorio, T. (Eds.), 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), Association for Computational Linguistics, Minneapolis, Minnesota. pp. 4171–4186. doi:10.18653/v1/N19-1423.

Du, M., He, F., Zou, N., Tao, D., Hu, X., 2024. Shortcut learning of large language models in natural language understanding. Communications of the ACM 67, 110–120.

ElSherief, M., Ziems, C., Muchlinski, D., Anupindi, V., Seybolt, J., De Choudhury, M., Yang, D., 2021. Latent hatred: A benchmark for understanding implicit hate speech, in: Moens, M.F., Huang, X., Specia, L., Yih, S.W.t. (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic. pp. 345–363. URL: https://aclanthology.org/2021.emnlp-main.29, doi:10.18653/v1/2021.emnlp-main.29.

Gao, T., Yao, X., Chen, D., 2021. Simcse: Simple contrastive learning of sentence embeddings, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 6894–6910.

Gilardi, F., Alizadeh, M., Kubli, M., 2023. Chatgpt outperforms crowd workers for text-annotation tasks. Proceedings of the National Academy of Sciences 120, e2305016120.

Govindarajan, V., Van Durme, B., White, A.S., 2019. Decomposing generalization: Models of generic, habitual, and episodic statements. Transactions of the Association for Computational Linguistics 7, 501–517.

Guzman, E., El-Haliby, M., Bruegge, B., 2015. Ensemble methods for app review classification: An approach for software evolution (n), in: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE. pp. 771–776.

Hadi, M.A., Fard, F.H., 2023. Evaluating pre-trained models for user feedback analysis in software engineering: A study on classification of app-reviews. Empirical Software Engineering 28, 88.

Hauer, B., Kondrak, G., Luan, Y., Mallik, A., Mou, L., 2021. Semi-supervised and unsupervised sense annotation via translations, in: Mitkov, R., Angelova, G. (Eds.), Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), INCOMA Ltd., Held Online. pp. 504–513. URL: https://

aclanthology.org/2021.ranlp-1.57.

Henao, P.R., Fischbach, J., Spies, D., Frattini, J., Vogelsang, A., 2021. Transfer learning for mining feature requests and bug reports from tweets and app store reviews, in: 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), IEEE. pp. 80–86.

Hey, T., Keim, J., Koziolek, A., Tichy, W.F., 2020. Norbert: Transfer learning for requirements classification, in: 2020 IEEE 28th international requirements engineering conference (RE), IEEE. pp. 169–179.

Hoes, E., Altay, S., Bermeo, J., 2023. Using chatgpt to fight misinformation: Chatgpt nails 72% of 12,000 verified claims. PsyArXiv. April 3.

Hu, G., He, W., Sun, C., Zhu, H., Li, K., Jiang, L., 2023. Hierarchical belief rule-based model for imbalanced multi-classification. Expert Systems with Applications 216, 119451. URL: https://www.sciencedirect.com/science/article/pii/S0957417422024708, doi:https://doi.org/10.1016/j.eswa.2022.119451.

Huang, F., Kwak, H., An, J., 2023. Is chatgpt better than human annotators? potential and limitations of chatgpt in explaining implicit hate speech, in: Companion proceedings of the ACM web conference 2023, pp. 294–297.

Iacob, C., Faily, S., Harrison, R., 2016. Maram: tool support for mobile app review management .

Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D.d.l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al., 2023. Mistral 7b. arXiv preprint arXiv:2310.06825 .

Li, T., Li, Y., Xia, T., Hui, P., 2021. Finding spatiotemporal patterns of mobile application usage. IEEE Transactions on Network Science and Engineering .

Lin, Y., Meng, Y., Sun, X., Han, Q., Kuang, K., Li, J., Wu, F., 2021. Bertgcn: Transductive text classification by combining gnn and bert, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 1456–1462.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 .

Maalej, W., Nabil, H., 2015. Bug report, feature request, or simply praise? on automatically classifying app reviews, in: 2015 IEEE 23rd international requirements engineering conference (RE), IEEE. pp. 116–125.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Alobeidli, H., Cappelli, A., Pannier, B., Almazrouei, E., Launay, J., 2024. The refinedweb dataset for falcon llm: outperforming curated corpora with web data only, in: Proceedings of the 37th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA.

Pennington, J., Socher, R., Manning, C.D., 2014. Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543.

Piao, Y., Lee, S., Lee, D., Kim, S., 2022. Sparse structure learning via graph neural networks for inductive document classification, in: Proceedings of the AAAI conference on artificial intelligence, pp. 11165–11173.

Pustejovsky, J., Stubbs, A., 2012. Natural Language Annotation for Machine Learning: A guide to corpus-building for applications. " O'Reilly Media, Inc.".

Rana, T.A., Cheah, Y.N., 2017. A two-fold rule-based model for aspect extraction. Expert Systems with Applications 89, 273–285. URL: https://www.sciencedirect.com/science/article/pii/S0957417417305249, doi:https://doi.org/10.1016/j.eswa.2017.07.047.

Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 .

Scalabrino, S., Bavota, G., Russo, B., Di Penta, M., Oliveto, R., 2017. Listening to the crowd for the release planning of mobile apps. IEEE Transactions on Software Engineering 45, 68–86.

Shams, R.A., Hussain, W., Oliver, G., Nurwidyantoro, A., Perera, H., Whittle, J., 2020. Society-oriented applications development: Investigating users' values from bangladeshi agriculture mobile applications, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society, pp. 53–62.

Song, J., Ding, H., Wang, Z., Xu, Y., Wang, Y., Zhao, J., 2024. Itake: Interactive unstructured text annotation and knowledge extraction system with llms and modelops, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pp. 326–334.

Sun, W., Miao, Y., Li, Y., Zhang, H., Fang, C., Liu, Y., Deng, G., Liu, Y., Chen, Z., 2024. Source code summarization in the era of large language models. CoRR abs/2407.07959, 1–13.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M.S., Love, J., et al., 2024. Gemma: Open models based on gemini research and technology. arXiv preprint arXiv:2403.08295 .

Team, M.N., 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. URL: www.mosaicml.com/blog/mpt-7b. accessed: 2023-05-05.

Tizard, J., Wang, H., Yohannes, L., Blincoe, K., 2019. Can a conversation paint a picture? mining requirements in software forums, in: 2019 IEEE 27th International Requirements Engineering Conference (RE), IEEE. pp. 17–27.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems 30.

Villarroel, L., Bavota, G., Russo, B., Oliveto, R., Di Penta, M., 2016. Release planning of mobile apps based on user reviews, in: Proceedings of the 38th International Conference on Software Engineering, pp. 14–24.

Wang, K., Han, S.C., Poon, J., 2022. Induct-gcn: Inductive graph convolutional networks for text classification, in: 2022 26th International Conference on Pattern Recognition (ICPR), IEEE. pp. 1243–1249.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., Sui, Z., 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 9426–9439.

Wang, Y., Wang, C., Zhan, J., Ma, W., Jiang, Y., 2023. Text fcg: Fusing contextual information via graph learning for text classification. Expert Systems with Applications 219, 119658.

Wang, Y., Wang, S., Yao, Q., Dou, D., 2021. Hierarchical heterogeneous graph representation learning for short text classification, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 3091–3101.

Wei, J., Courbis, A.L., Lambolais, T., Xu, B., Bernard, P.L., Dray, G., 2022. Towards a data-driven requirements engineering approach: Automatic analysis of user reviews. arXiv preprint arXiv:2206.14669 .

Wei, J., Courbis, A.L., Lambolais, T., Xu, B., Bernard, P.L., Dray, G., 2023. Zero-shot bilingual app reviews mining with large language models, in: 2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE Computer Society, Los Alamitos, CA, USA. pp. 898–904. URL: https://doi.ieeecomputersociety.org/10.1109/ICTAI59109.2023.00135, doi:10.1109/ICTAI59109.2023.00135.

Williams, G., Mahmoud, A., 2017. Mining twitter feeds for software user requirements, in: 2017 IEEE 25th International Requirements Engineering Conference (RE), IEEE. pp. 1–10.

Xie, Q., Huang, J., Du, P., Peng, M., Nie, J.Y., 2021. Inductive topic variational graph auto-encoder for text classification, in: proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies, pp. 4218–4227.

Yang, Y., Miao, R., Wang, Y., Wang, X., 2022. Contrastive graph convolutional networks with adaptive augmentation for text classification. Information Processing & Management 59, 102946.

Yao, L., Mao, C., Luo, Y., 2019. Graph convolutional networks for text classification, in: Proceedings of the AAAI conference on artificial intelligence, pp. 7370–7377.

Zhu, Y., Zhang, P., Haq, E.U., Hui, P., Tyson, G., 2023. Can chatgpt reproduce human-generated labels? a study of social computing tasks. arXiv preprint arXiv:2304.10145 .