國立陽明交通大學
資訊管理研究所
碩士論文

Institute of Information Management
National Yang Ming Chiao Tung University
Master Thesis

基於大型語言模型合成數據訓練的半監督文本分類方法

A semi-supervised text classification method based on large language model synthetic data training

研究生：聶聞華 (Nie, Wen-Hua)
指導教授：古政元 (Ku, Cheng-Yuan)

中華民國 一一三年六月
August 2024

基於大型語言模型合成數據訓練的半監督文本分類方法

A semi-supervised text classification method based on large language model synthetic data training

研 究 生：聶聞華
指導教授：古政元

Student: Nie, Wen-Hua
Advisor: Dr. Ku, Cheng-Yuan

國立陽明交通大學
資訊管理研究所

碩士論文

A Thesis
Submitted to
Institute of Information Management
College of Management
National Yang Ming Chiao Tung University
in Partial Fulfilment of the Requirements
for the Degree of
Master
in
Information Management

August 2024

Taiwan, Republic of China

中華民國 一一三年六月

基於大型語言模型合成數據訓練的半監督文本分類方法

學生：聶聞華　　指導教授：古政元 博士

國立陽明交通大學 資訊管理研究所

摘　　要

隨著大型語言模型（LLMs）的迅速發展，它們在文本生成和問答等任務中展現了卓越的能力，相關研究不斷增多。本研究詳細探討了使用大型語言模型生成資料來增強訓練效果的半監督文本分類方法。該方法通過大型語言模型將已標記資料作爲參考，生成資料集中尚未包含的資料，從而減少所需的標記資料量，並提升模型的訓練效果。在文本分類任務中，使用這種半監督方法，即使在相同標記資料量下，經過四個資料集的驗證，accuracy 可提高 1-2%。這項技術展示了在資料稀缺的情況下，未來可以利用生成模型來合成資料，減少標記資料所需的人力，並優化模型訓練的可能性。

關鍵詞: 大型語言模型、深度學習、機器學習、合成資料、半監督、文本分類

A semi-supervised text classification method based on large language model synthetic data training

Student: Nie, Wen-Hua Advisor: Dr. Ku, Cheng-Yuan

Institute of Information Management
National Yang Ming Chiao Tung University

Abstract

With the rapid development of large language models (LLMs) recently, they have demonstrated outstanding capabilities in generating text, answering questions, and other related areas, leading to an increasing number of studies. This research delves into a semi-supervised text classification method that leverages synthetic data generated by LLMs to enhance training effectiveness. This method uses LLMs to reference labeled data and synthesize new data that is not included in the dataset, thereby reducing the required amount of labeled data and improving model training performance. Using this semi-supervised method, the accuracy in text classification tasks can be improved by 1-2% across four datasets with the same amount of labeled data. This technique successfully demonstrates the potential for using generative models to synthesize data in situations with limited data availability, reducing the need for manual labeling and optimizing model training possibilities.

Keyword: Large language models、Deep Learning、Machine learning、synthetic data、

semi-supervised、text classification

Contents

摘要 i

Abstract ii

Contents iv

List of Figures vi

List of Tables vii

1 Introduction 1

2 Related Work 4

2.1 large language models (LLMs) 4

2.1.1 GPT 4

2.1.2 Llama 3 5

2.2 Data Augmentation 6

2.3 Pseudo-labeling 7

2.4 pre-training model 8

2.4.1 BERT 8

2.4.2 RoBERTa 9

2.4.3 BGE-M3 10

2.5 k-means 11

3 Method 13

3.1 Method Architecture 13

3.2 Large Language Model Data Augmentation Module 15

3.3 Pseudo-Label Optimization Module 16

3.4 loss function 17

4 Experiments 18

4.1 Dataset 18

4.2 Evaluation Metrics 18

4.3 Environment Setup 19
4.4 Hyper-parameter Setup 19
4.5 Experimental Results 20
4.5.1 Impact of L_d Quantity 21
4.5.2 Impact of Pseudo-Label Optimization Module 22
4.5.3 Discussion of results 23
5 Conclusion and Future Prospects 25
5.1 Conclusion 25
5.2 Future Prospects 25
Reference 26

List of Figures

Figure 2.1 Bert fine tuning 8

Figure 3.1 method 14

Figure 4.1 method 20

List of Tables

Table 3.1 Module Samples of Basic Elements for This Project 16

Table 4.1 The statistics of the datasets. 18

Table 4.2 Dataset and Text 18

Table 4.3 pre-trained models transformer format 19

Table 4.4 L_d quantity's impact on LLM 21

Table 4.5 L_d quantity's impact on the pseudo-labeling optimization module 22

Table 4.6 Results of the pseudo-labeling optimization module under different LLM 23

Table 4.7 Comparison of results 23

Chapter 1

Introduction

Text classification is a crucial field in natural language processing (NLP) with numerous practical uses, including spam detection, topic categorization, question answering, and sentiment evaluation. Although notable advancements have been achieved with pre-trained language models and large volumes of labeled data, these models typically require significant amounts of training data. Securing high-quality annotations can be expensive, posing challenges for low-resource relation extraction. In text classification tasks, the presence of unknown categories, diverse types, and intricate text data can complicate both model training and classification. In real-world scenarios, the costs associated with data collection and annotation contribute to data sparsity issues, particularly as the variety of text types expands.

Semi-supervised learning (SSL) is gaining traction as it reduces reliance on extensively annotated datasets, thus facilitating the deployment of deep neural networks in new scenarios. SSL thrives by leveraging a minimal amount of labeled data alongside a larger volume of unlabeled data, which is typically more accessible and economical to collect. To enhance the model's classification accuracy in semi-supervised text classification, researchers aim to minimize discrepancies between the training and inference phases to avoid overfitting. They utilize consistency regularization techniques that focus on maintaining stable predictions for the same sample by introducing various perturbations [1]. These methods often involve perturbing the embedding layer to ensure consistent output distributions between perturbed and original data.

For example, Virtual Adversarial Training (Virtual-AT) [2] introduces minor variations around each data point, thereby defining the robustness of the conditional label distribution. Rev-LSTM [3] integrates losses from Virtual-AT [2], entropy minimization [4], and adversarial examples [5], merging supervised and unsupervised learning while adding perturbations to the neural network's outputs.

During the training of deep neural networks, a notable gap often emerges between the training and inference phases, primarily attributed to the use of dropout, which randomly disables neurons. This gap can lead to overfitting and reduce classification effectiveness, particularly in semi-supervised learning environments where labeled data are scarce. To address this issue, researchers employ pseudo-labeling methods. In this strategy, a model initially trained on a labeled dataset is used to predict labels for unlabeled data; these labels, termed pseudo-labels, are then reincorporated into the training set. This technique is intended to boost the model's performance by effectively enlarging the pool of labeled data. Methods like Self-training [6] autonomously produce pseudo-labels for unlabeled data, enhancing the accuracy and stability of classifiers [7][8]. Pavlinek and Podgorelec[9] investigated the use of topic models for text representation to improve SSL approaches, developing a news text classification strategy that combines self-training and LDA topic models to augment small labeled datasets with unlabeled material. Kumar et al. [10] developed a novel binary classifier system that enhances the efficiency of pseudo-labeling in traditional SSL for text classification by tackling threshold-related problems. Pseudo-labeling is effective in creating text classification models with limited datasets, increasing the amount of training data and improving classification accuracy. Nonetheless, the effectiveness of this approach heavily relies on the reliability of the classifier trained with well-documented data, necessitating a comprehensive and accurately annotated training set.

Synthetic data creation is an approach to augment existing training sets. Through Rule-G [11], this technique is refined by introducing variations into the embedding matrix, which is commonly referred to as data augmentation. An alternate method involves the use of pre-trained models built on vast datasets. Despite the indispensable nature of real-world annotated datasets, they are often scarce, costly, and prone to inherent biases [12], highlighting the importance of methods for generating synthetic data. In particular, producing synthetic question-answer pairs is crucial in these scenarios [13]. Furthermore, creating data via template-driven question generation provides a systematic approach [14].

The recent advancements in large language models (LLMs)[15][16] have introduced a novel

application of the large language model ChatGPT for text annotation tasks, surpassing the performance of crowd-sourced workers in multiple tasks such as relevance determination, stance detection, topic identification, and frame detection. ChatGPT has shown higher zero-shot accuracy in four of the five categories than crowd-sourced workers and has provided more consistent outcomes than both crowd workers and professional annotators across all evaluated tasks. Furthermore, the cost of annotation using ChatGPT is under .003, making it approximately twenty times more cost-effective than using platforms like MTurk. These findings underscore the potential of large language models in text classification, demonstrating that LLMs can produce annotated data, thereby enriching the volume and quality of labeled datasets through synthetic data and improving dataset quality and classification accuracy.

This research introduces a semi-supervised text classification approach that capitalizes on synthetic data produced by large language models to improve training efficacy. Specific prompts are employed to generate synthetic data with large language models. Subsequently, a pre-trained language model and k-means clustering are used to train the labeled data and assign pseudo-labels to the synthetic data. This procedure is reiterated to optimize the quality of synthetic data. Finally, a dataset combining labeled and synthetic data propagates pseudo-labels to unlabeled data, enhancing model training effectiveness. Experimental findings indicate that the proposed semi-supervised approach, which utilizes large language models for data synthesis, yields superior classification performance with the same volume of labeled data compared to alternative methods.

Chapter 2

Related Work

2.1 large language models (LLMs)

2.1.1 GPT

GPT-4 (Generative Pre-trained Transformer 4) is an advanced multimodal language model developed by OpenAI. It possesses the ability to understand and generate multimodal data, including capabilities in visual understanding and visual-text semantic fusion.

In GPT-4, chain-of-thought reasoning is a core capability. This chain-of-thought formation mechanism can be explained as the model constructing an intrinsic representation of language structure and meaning by learning from vast amounts of language data. It completes final outputs through a series of intermediate natural language reasoning steps. This allows GPT-4 to excel in reasoning tasks, mimicking human thought processes by breaking down complex problems into intermediate steps that can be solved individually. GPT-4's chain-of-thought ability endows it with logical analysis capabilities, enabling it to perform well in various NLP tasks such as solving math problems, symbolic operations, and commonsense reasoning.

Multimodal prompt engineering is an essential technique in GPT-4. It refers to designing a series of questions or tasks based on specific goals and contexts to generate coherent and meaningful text on a given subject or domain. Through carefully designed prompts, prompt engineering can enhance the quality and relevance of the generated text. In GPT-4, prompt engineering involves selecting appropriate model architectures and parameters, designing prompt formats and structures, choosing suitable tasks and training data, and fine-tuning the model using the selected prompts and data. This approach significantly improves the operability of the language model, allowing it to alter its behavior based on user requirements.

GPT-4o (where ”o” signifies ”omni”) represents a step forward in enhancing human-computer interactions, providing a more natural experience. This model accepts a diverse range of inputs including text, audio, images, and videos, and is adept at generating outputs across text, audio, and image formats. It boasts the capability to answer audio queries in as little as 232 milliseconds, and maintains an average response duration of 320 milliseconds, aligning closely with human response times. GPT-4o matches the performance levels of GPT-4 Turbo in handling English and coding-related texts, shows notable improvements in processing texts in other languages, and offers both faster service and a cost reduction of 50% on its API.

2.1.2 Llama 3

Llama 3 is an advanced language model developed by Meta, focusing on innovation, scalability, and simplifying model design. The entire Llama 3 project centers around four key areas: model architecture, pre-training data, scaling pre-training, and instruction fine-tuning.

Llama 3 employs a standard decoder-only Transformer architecture with several improvements. Compared to Llama 2, Llama 3 uses a more efficient tokenizer with a vocabulary size of 128K, which enhances language encoding efficiency and significantly boosts model performance. To improve inference efficiency, Llama 3 incorporates grouped query attention (GQA) in both its 8B and 70B parameter models, ensuring that the self-attention mechanism does not cross document boundaries when handling long sequences, thereby enhancing efficiency in processing lengthy texts.

The success of Llama 3 largely depends on a large-scale, high-quality training dataset. The model is pre-trained on over 15 trillion tokens, all from public sources, which is seven times the size of the Llama 2 dataset and includes more than four times the amount of code data. To effectively utilize this training data, Llama 3 adopts detailed scaling laws to optimize the pre-training process. These scaling laws help in selecting the optimal data mixture and making the best training computation decisions. It was found that model performance continues to improve even after training on data two orders of magnitude larger. To train the largest Llama 3 model, a combination of data parallelism, model parallelism, and pipeline parallelism methods was used,

ensuring efficient utilization of vast computational resources.

Llama 3 also introduces innovations in instruction fine-tuning methods, combining supervised fine-tuning (SFT), rejection sampling, proximal policy optimization (PPO), and direct preference optimization (DPO). High-quality prompts and preference rankings significantly impact model performance. By selecting and reviewing these data through multiple rounds of quality assurance, Llama 3 greatly enhances its performance in reasoning and coding tasks. Learning preference rankings enables the model to better choose the correct answers, producing accurate reasoning processes even when faced with difficult reasoning problems, thus improving overall performance.

These advancements benefit the current research in data augmentation by enhancing data quality and optimizing performance.

2.2 Data Augmentation

Data Augmentation involves the artificial creation of new data derived from existing datasets, primarily to enhance the training of new machine learning (ML) models. For ML models to be effectively trained, they require extensive and varied datasets. However, obtaining such diverse real-world datasets is often challenging due to factors like data silos, regulations, and other limitations. Data Augmentation tackles this issue by making minor alterations to the original data, thereby increasing the dataset size artificially.

Kafle et al. [17] introduced an innovative approach to Data Augmentation in the context of visual question answering. Unlike earlier methods that only replaced individual words or a few words, Kafle et al.'s technique generates complete sentences.

The first technique employs template augmentation, using predefined questions to generate answers paired with template questions in a rule-based manner. The second technique uses LSTM to generate questions by providing image features.

In our approach, specific prompts are utilized to allow LLMs to reference labeled data and generate data with corresponding labels.

2.3 Pseudo-labeling

Pseudo-labeling is a technique where an initially trained model on labeled data generates labels for unlabeled data. These labeled pseudo-labels are then integrated into the primary training dataset to enhance the model's precision and overall functionality, typically resulting in better classification outcomes. Within semi-supervised learning, pseudo-labeling research is categorized into three main types: conventional pseudo-labeling, dictionary-based pseudo-labeling, and pseudo-labeling incorporating Data Augmentation. In the conventional pseudo-labeling method, a model trained on labeled data is applied to produce labels for unlabeled data. These generated pseudo-labels are reinserted into the training dataset to continue refining the model. This process often includes strategies to carefully select the most accurate labels, to minimize the risk of incorporating incorrect pseudo-labels that might degrade the model's performance.

Li et al.[18] developed S2TC-BDD, a method that boosts performance by assessing the reliability of pseudo-labels through the variability between labeled and pseudo-labeled texts. Additionally, Mekala et al. introduced LOPS[19], a technique for weakly supervised text classification that determines the probability of incorrect annotations by observing the learning sequence of samples, based on the principle that models typically learn clean-labeled samples prior to those with noisy labels.

2.4 pre-training model

2.4.1 BERT

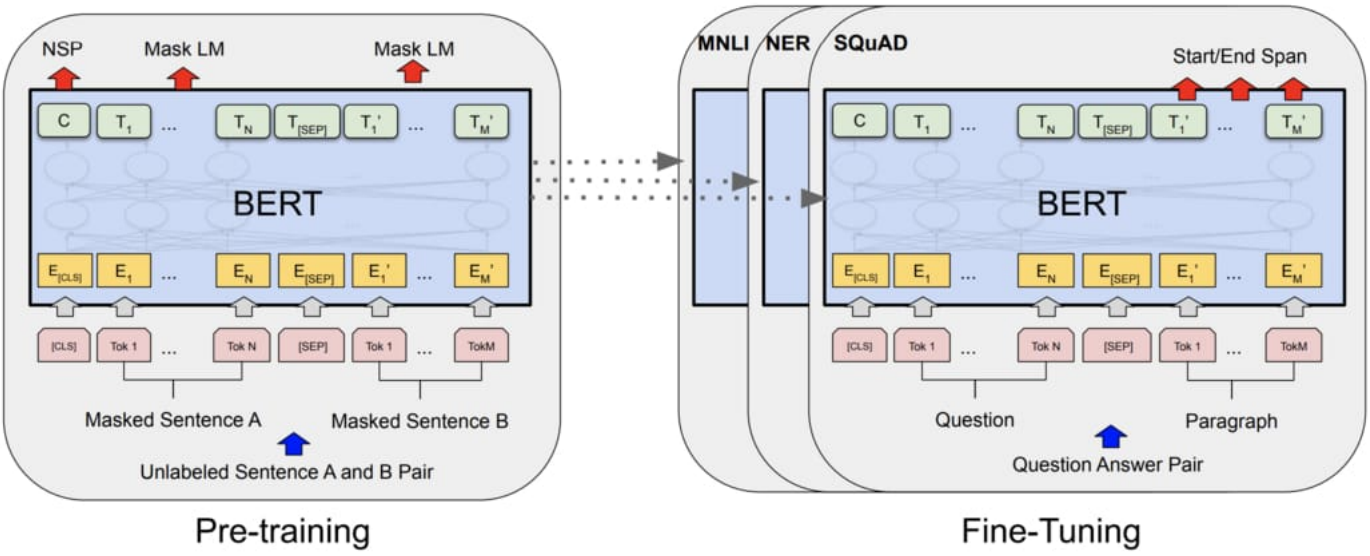

Figure 2.1 Bert fine tuning

BERT (Bidirectional Encoder Representations from Transformers) represents a significant advancement in natural language processing (NLP), introduced by Google AI Language in 2018 [20]. This model utilizes a bidirectional approach to language modeling and incorporates Transformer architectures to effectively grasp the contextual subtleties needed for encoding sequences. BERT's training regimen is structured in two stages: pre-training and fine-tuning. Initially, during the pre-training stage, BERT is educated using a vast array of text data. Subsequently, the fine-tuning stage employs labeled text data to refine the model's predictive accuracy.

During BERT's pre-training stage, the model is engaged in two primary activities: Masked Language Model (MLM) and Next Sentence Prediction (NSP). In the MLM task, BERT obscures certain words within the input sentences (for example, substituting them with the "[MASK]" token) and is trained to deduce these obscured words. This method ensures that BERT relies on

the surrounding textual context rather than the Transformer's direct output positions to predict the masked words, effectively mitigating potential data leakage issues.

For the NSP task, BERT's goal is to ascertain whether two sentences are consecutive in a semantic sense, i.e., whether the second sentence logically follows the first one. During pre-training, BERT randomly pairs sentences and generates an input sequence for each pair. The input sequence comprises two segments: sentence A and sentence B, divided by a special "[SEP]" token. Additionally, BERT includes a special "[CLS]" token at the beginning of the sequence. For NSP, BERT uses a binary label: IsNext (next sentence) and NotNext (not the next sentence). During training, the BERT model processes the entire sequence through the Transformer network, obtains the final hidden layer output, and uses these outputs to predict the labels, determining whether the sentence pairs are sequential.

Through the NSP task, the BERT model learns sentence-level semantic relationships, including contextual information and coherence between sentences. This enables BERT to better understand the semantic connections between sentences and provide more precise and comprehensive semantic representations for various NLP tasks.

In the fine-tuning phase, BERT uses the representation of the "[CLS]" token for classification, as the "[CLS]" token has learned sentence-level knowledge. This allows BERT to perform exceptionally well across a variety of NLP tasks.

2.4.2 RoBERTa

RoBERTa (A Robustly Optimized BERT Pretraining Approach) [21] introduces a series of improvements based on the BERT model:

In BERT's preprocessing, each sample undergoes random word masking, and the same masked words are used for subsequent training, a method known as static masking because the masking remains fixed throughout the training process. In contrast, RoBERTa dynamically selects 15% of tokens to mask each time inputs are provided, making the training process more flexible and enriching, thereby better utilizing the training data.

RoBERTa eliminates the NSP task and instead inputs multiple consecutive sentences at a

time, training on entire texts to learn relationships between sentences. Compared to BERT's NSP task, RoBERTa more directly and effectively learns the connections between sentences.RoBERTa increases the training data from BERT's 16GB to 160GB and uses larger batch sizes and longer training times to enhance model performance and training efficiency.

In BERT, the text is first split into individual characters and then gradually merged using the Byte Pair Encoding (BPE) algorithm to form a vocabulary, such as merging ”u” and ”nited” into the word ”united.” BERT's vocabulary size is about 30,000. This character-level BPE encoding method can handle words not present in the vocabulary but is not very friendly to irregular spelling or polysemous words and does not easily address the Out-of-Vocabulary (OOV) problem. RoBERTa uses byte-level BPE, which first converts each character to its ASCII byte representation and then encodes the bytes using BPE. With a vocabulary size of 50,000, this method is finer-grained compared to BERT and is better suited for handling irregularly spelled text and OOV issues. RoBERTa's byte-level BPE encoding method enriches the vocabulary and better addresses OOV problems.

2.4.3 BGE-M3

BGE-M3 [22] is a highly flexible and versatile embedding model capable of handling multi-language, multi-granularity input data, and unifying various functions of text retrieval. This model aims to provide powerful and flexible embedding representations for natural language processing tasks.

BGE-M3 supports multiple languages and different granularities of input data, and unifies common retrieval functions of text embeddings. Specifically, given a query in any language, the model can retrieve relevant documents in another language or the same language. This makes BGE-M3 applicable to cross-language retrieval and multi-language corpus processing.

In terms of data organization, training BGE-M3 requires large-scale and diverse multilingual datasets. These datasets include unsupervised data, annotated data, and synthetic data, which complement each other and are applied at different stages of the training process. Unsupervised data comes from unannotated corpora and is organized by extracting rich semantic structures

from sources like Wikipedia, S2ORC [23], and xP3 [24], including structures like title-body, title-abstract, and instruction-output. Annotated data comes from annotated corpora and synthetic data, which are used for fine-tuning the model to improve performance and accuracy.

For retrieval functions, BGE-M3 unifies dense retrieval, sparse/lexical retrieval, and multi-vector retrieval. Dense retrieval measures relevance by converting queries and documents into hidden states and calculating their inner product. Sparse retrieval measures relevance by calculating the weight of each word in the query and document and summing the weights of common words. Multi-vector retrieval extends dense retrieval by using the entire output embeddings to represent queries and documents, and calculating fine-grained relevance scores through delayed interaction.

Additionally, BGE-M3 employs self-knowledge distillation to optimize the training process for multiple retrieval functions. By integrating the prediction results of different retrieval methods into more accurate relevance scores, the model can more effectively learn and apply various retrieval functions. Ultimately, BGE-M3 provides accurate and comprehensive embedding representations for multilingual, multi-granularity text retrieval tasks, offering strong support for various natural language processing tasks. In this study, the pre-trained language models mentioned above will be used to generate embeddings and train classifiers to achieve the goal of text classification.

2.5 k-means

The k-means clustering technique, initially developed by MacQueen in 1967 [25], was first applied in vector quantization within the field of signal processing and has subsequently become a fundamental clustering technique. The main purpose of this algorithm is to divide n data points, which might be either samples or instances, into k distinct clusters. In doing so, it assigns each data point to the nearest cluster based on mean distance, which serves as the criterion for clustering. Given a series of observations $(x_1, x_2, \ldots, x_n)$, where each is defined by a d-dimensional real vector, the k-means algorithm seeks to categorize these n observations into k

subsets ($k \leq n$) to minimize the within-cluster sum of squares (WCSS). The objective is to establish clusters S_i that meet the following criteria:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

in which μ_i represents the average of all points within S_i.

In this analysis, k-means clustering is initially employed on labeled data for training. Subsequently, it creates pseudo-labels for synthetic data. These pseudo-labels are then cross-validated with those generated by a pre-trained model, enhancing their precision and thus boosting their overall quality.

Chapter 3

Method

In this chapter, the detailed model architecture and training methods of this study will be introduced. Section 3.1 describes the model architecture. Section 3.2 introduces the setting of prompts in the data augmentation module of the large language model and the data augmentation process. Section 3.3 describes the process of using pre-trained models and clustering to optimize pseudo-labels in the pseudo-label optimization module.

3.1 Method Architecture

The experiments in this study mainly use a unified self-training method to fine-tune the pre-trained models BERT, RoBERTa, and BGE-M3. These methods and models are described in detail in Chapter 2, so they will not be elaborated upon in this section.

This study consists of two modules: data augmentation with large language models and pseudo-label optimization. The specific training process is illustrated in Figure 1. When dealing with different domains or practical problems, it is common to encounter the issue of a lack of labeled datasets. To address this, data augmentation is used to expand the dataset, and optimized pseudo-labels are employed for semi-supervised training.

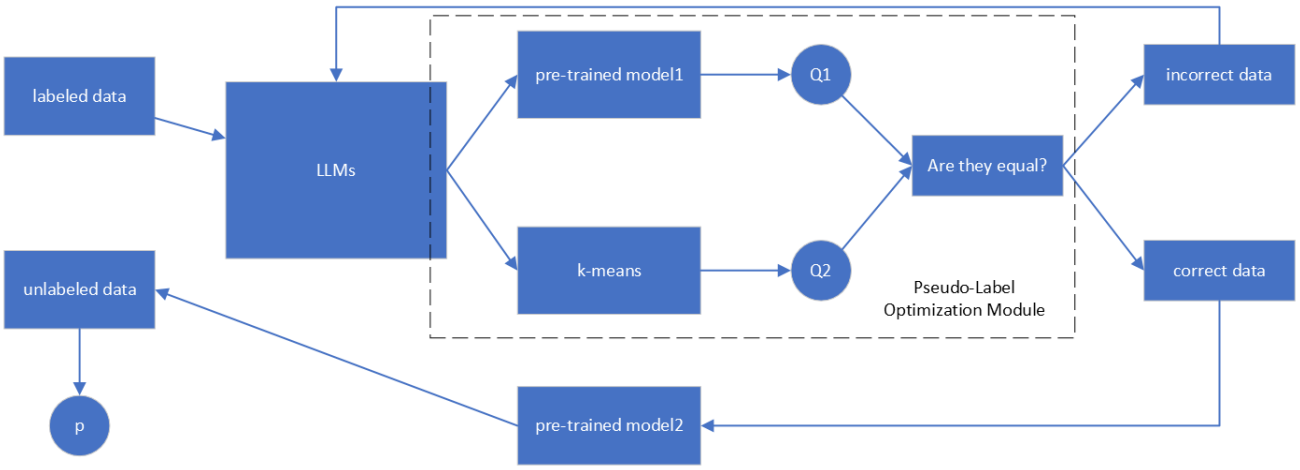

Figure 3.1 method

First, the labeled data (L_d) and unlabeled data (L_u) are prepared. L_u is divided into n parts equal to the number of L_d:

$$L_{u1}, L_{u2}, \ldots, L_{un}$$

L_d is input into the data augmentation module of the large language model, generating synthetic data through specific prompts. Then, in the pseudo-label optimization module, the synthetic data is first converted to embeddings by the pre-trained model. The synthetic data undergoes two rounds of pseudo-labeling using both the pre-trained model classifier trained on L_d and the k-means model, resulting in two sets of pseudo-labels, Q_1 and Q_2. The data points with matching Q_1 and Q_2 labels are considered successfully synthesized data, while the incorrect data points are fed back to the data augmentation module of the large language model to improve the quality of subsequent synthetic data through the context memory of the large model.

Finally, the combined training set of L_d and synthetic data is used to train model2, and predictions are made on the unlabeled data. The predicted labels at this stage are recorded as pseudo-labels P. This process is repeated n times, resulting in pseudo-labels P for all the unlabeled data. The accuracy of these pseudo-labels is then tested.

3.2 Large Language Model Data Augmentation Module

In this research, ensuring the generation quality of large language models is critical. Consequently, we utilized LangGPT [26], a GPT-like language model designed as a programming language for large language models (LLMs). LangGPT integrates the structured, standardized, and reusable features of programming languages with the adaptable and extensible nature of natural language. By analyzing the distinctions between natural and programming languages, we determined the essential attributes that prompt templates should have. LangGPT operates as a dual-layer system, comprising modules and internal components.

LangGPT's modules are categorized into two types: intrinsic modules and extended modules. For intrinsic modules, we carefully crafted the necessary internal components and provided template examples. Based on the LangGPT template, our prompt design is as Table 3.21.

In the "Examples" section, we input data from L_d as examples, but the quantity must be determined based on the actual amount of labeled data and the maximum number of tokens the model can accept. Additionally, since it is necessary for LLMs to maintain context memory to optimize the generation quality, a Reminder command is added to help alleviate the forgetting problem of LLMs. The differences between different LLMs will be compared in detail in the experimental section.

Another point that needs to be explained in detail is the reason for not directly using LLMs to generate labels. As mentioned earlier, the current maximum number of tokens LLMs can accept cannot accommodate all data at once when the amount of labeled data is high, nor can it guarantee the accuracy of LLMs' prior knowledge. More critically, the correct optimization of labels is only retained in a complete conversation memory. If a new round of conversation starts, the optimized label improvements cannot be preserved. Therefore, generating high-quality data augmentation is a better solution.

Module	Samples of Basic Elements
Profile	Author: NIE, WENHUA Version: 1.0 Language: English Description: Data Generator
Skill	Read provided text and determine the theme. Identify detailed sub-themes within the given theme. Generate data based on the given theme.
Rules	The generated data must be original and unique. When using a classifier, the generated data's theme must match the given theme. The one hundred generated sentences must all be completely unique, with no repetitions. In each pair of sentences, the number of identical words cannot exceed three, yet they should remain on the same topic. Sentences do not need to be coherent, just identify the theme. Only provide the generated sentences, without quotes or any additional text. Each generated sentence must be at least 10 words long.
Workflow	User provides the theme. Generate entirely new data based on the provided theme, ensuring complete uniqueness.
Reminder	Description: You will always remind yourself role settings and you output Reminder contents before responding to the user. Reminder: The user language is language (<language>), rules (<rules>). "<output>"
Dataset Description	A subset of AG's news, consisting of news titles across four topic categories.
Initialization	Generate the same number of completely unique sentences as in the examples, with the same theme. The sentences should follow the style of the provided examples. Make sure they are categorized under the same topic.
Examples	cheetahs flourish spanish plain reuters reuters stalking teddy bears towels retiring harass family dog week cheetah cub bunjee blissfully unconcerned cameras trailing heidenreichs house. teen ebay con prompts warning detectives warn internet auction bidders wary pay teenage fraudster conned customers.

Table 3.1 Module Samples of Basic Elements for This Project

3.3 Pseudo-Label Optimization Module

In this study, BGE-M3 was chosen as the pre-trained model for embeddings and classifiers, ensuring consistency in embeddings as all text data is converted by BGE-M3. First, L_d is used to

train model1, based on the pre-trained BGE-M3 model, and model2, using the k-means method. Then, model1 predicts the synthetic data, providing pseudo-labels Q_1 for the synthetic data. Model2 identifies the L_d closest to the center of each cluster and assigns its label to the clustered synthetic data, resulting in pseudo-labels Q_2.

The reason for choosing the k-means clustering method to determine pseudo-labels is twofold: on the one hand, unsupervised clustering tasks group data with similar features into the same cluster, determining whether the data is of the same kind without specific labels. On the other hand, k-means calculates distances effectively, making correct data closer to the true labeled data, mitigating the limitations of LLMs in achieving the quality of manually selected data.

Then, the data with matching Q_1 and Q_2 labels are selected as correct data, while the rest are considered incorrect data. The incorrect data is fed back into the large language model data augmentation module to optimize the quality of the synthetic data.

3.4 loss function

The loss function is set to the cross-entropy function. When using neural networks for classification and prediction, cross-entropy error is generally better than classification error and is more effective in evaluating the quality of the neural network compared to mean squared error. During the training of classification models, cross-entropy is typically used as the loss function to be minimized:

$$CE(p, q) = -\sum_{i=1}^{C} p_i \log(q_i)$$

where C represents the number of classes, p_i is the true value, and q_i is the predicted value.

Chapter 4

Experiments

4.1 Dataset

We perform thorough experiments across eight widely utilized real-world datasets to evaluate the effectiveness and broad applicability of our method. Here are the specifics of each dataset:

AgNews [27]: A segment of the AG's news corpus, as compiled by [28], consists of 8,000 news titles categorized into four thematic classes. Table 4.1 illustrates this distribution, where C stands for the number of classes, N represents the total number of entries in the dataset, A shows the average word count per entry, and L/S depicts the size ratio between the largest and smallest clusters. Table 4.2 presents a textual sample from the AgNews corpus.

Dataset	C	N	A	L/S
AgNews	4	8,000	23	1

Table 4.1 The statistics of the datasets.

Dataset	Text
AgNews	”teen ebay con prompts warning detectives warn internet auction bidders wary pay teenage fraudster conned customers”

Table 4.2 Dataset and Text

4.2 Evaluation Metrics

We employ two prevalent metrics for evaluating text clustering outcomes: accuracy (ACC). ACC is given by:

$$ACC = \frac{\sum_{i=1}^{N} 1\{y_i = \text{map}(\hat{y}_i)\}}{N}, \tag{4.1}$$

where y_i and $\hat{y}_i$ denote the ground truth and the predicted label for the text x_i, respectively.

4.3 Environment Setup

The experiments were conducted in a Python 3.8 and PyTorch 2.0 environment. For hardware, we used a high-performance GPU: NVIDIA RTX 4090.

4.4 Hyper-parameter Setup

In the large language model data augmentation module, we chose three large models for experimentation: GPT-4o, GPT-4, and LLaMA-3-70B. The GPT series used OpenAI's official ChatGPT for data augmentation within the same conversation, while LLaMA-3-70B uses Meta AI..

For the pseudo-label optimization module, we used the pre-trained models BERT, RoBERTa, and BGE-M3 for training. The transformer format used is as follows:

pre-trained models	**transformer format**
BERT	"bert-base-uncased"
RoBERTa	"roberta-base"
BGE-M3	"BAAI/bge-base-en-v1.5"

Table 4.3 pre-trained models transformer format

We adopted the AdamW optimization algorithm. The training process was set to 100 epochs, with a batch size of 16 and an initial learning rate of 5×10^{-6}. The amount of data generated by LLM each time is equal to that of L_d. All experimental results were repeated five times, and the final average value was taken.

4.5 Experimental Results

First, in order to explore the impact of L_d data volume on the two modules, we designed two different experiments. Figure 3 shows an example of data generated within a single chat by GPT4o.

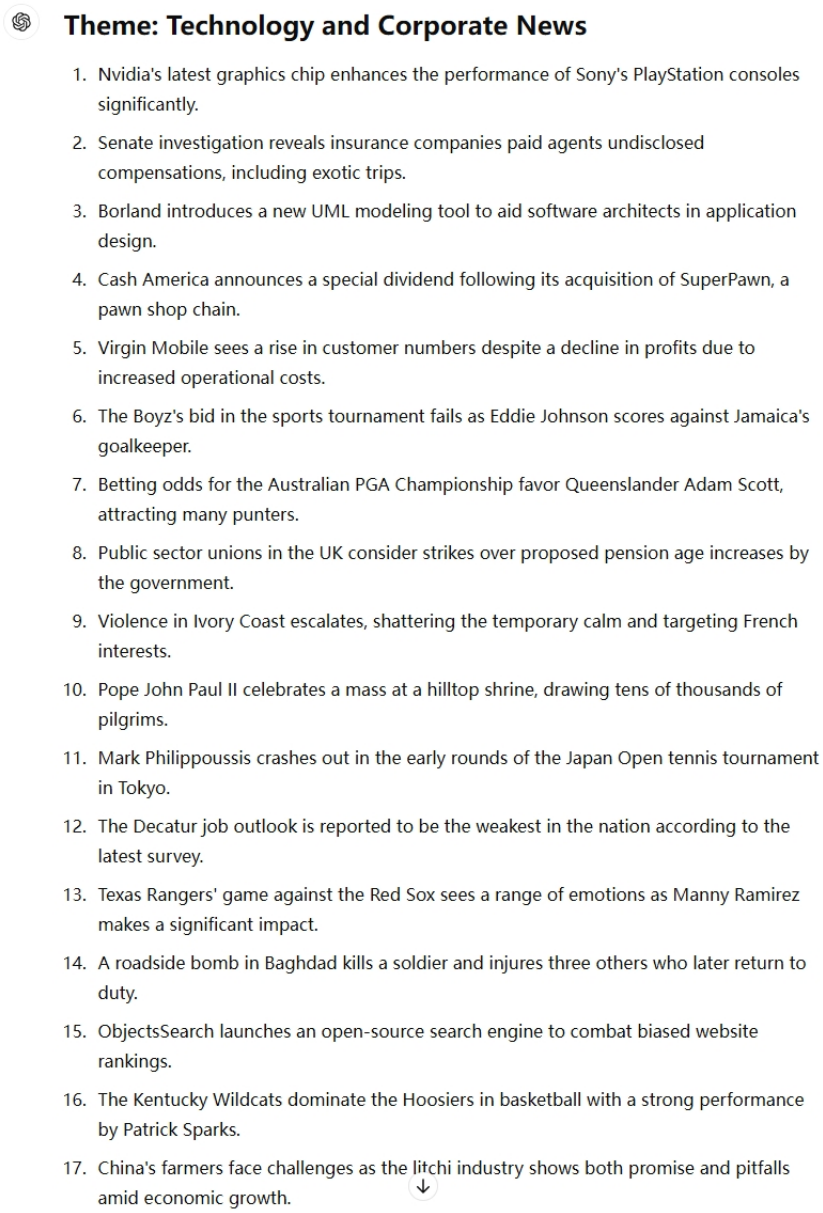

Figure 4.1 method

4.5.1 Impact of L_d Quantity

The quantity of L_d was uniformly set to (10, 25, 50, 100). By only changing the large language model and recording the accuracy (acc) for each L_d, as shown in Table 4.4, it can be observed that the large language model is influenced by the quantity of L_d. Even with different LLMs, the more L_d data there is, the better the quality of the synthetic data tends to be. To further verify, we also recorded the optimization process of the pseudo-labeling model under the condition of using GPT4o entirely.

method	**10**	**25**	**50**	**100**
BERT	72.85	70.67	76.39	83.45
BERT-Llama3	74.28	75.24	79.64	84.06
BERT-GPT4	76.76	73.23	75.76	79.35
BERT-GPT4o	71.73	74.45	78.54	85.11
RoBERTa	71.85	68.15	71.64	82.89
RoBERTa-Llama3	69.21	71.62	72.30	83.13
RoBERTa-GPT4	64.76	64.73	75.21	76.00
RoBERTa-GPT4o	69.80	72.70	72.30	85.48
BGE-M3	65.07	68.01	74.67	83.57
BGE-M3-Llama3	61.78	66.51	73.71	84.29
BGE-M3-GPT4	67.11	38.52	75.67	77.89
BGE-M3-GPT4o	66.93	73.23	77.03	85.37

Table 4.4 L_d quantity's impact on LLM

Under the same large language model conditions, we used three different pre-trained models as the training models for the pseudo-label optimization module. We compared the following scenarios:

Training only with L_d: "model name-LD". Training only with synthetic data: "model name-D". Training only with synthetic data optimized by the pseudo-label optimization module: " model name-DA". Training with a mixture of L_d and synthetic data optimized by the pseudo-label optimization module: "model name-ALL". Through Table 4.5, it can be observed that the pseudo-labeling optimization module is also influenced by the quantity of L_d. The quantity of L_d improves the quality of synthetic data, thereby enhancing the effectiveness of the pseudo-labeling optimization module. In the case of L_d quantities (10, 25, 50), a special situation arose

where the synthetic data performed better than the original data. Analyzing the experimental data, the reason was found to be that L_d was randomly selected. When the quantity of L_d is too small, it may not include all label types. Moreover, since k-means selects the label of L_d closest to the cluster center as the predicted label, insufficient data can result in poor clustering performance, leading to clustering failure. However, during synthesis, the data generated by the large language model tends to be more comprehensive, with the cluster centers being closer to the actual labels, thus resulting in better performance.

method	**10**	**25**	**50**	**100**
BERT-LD	72.85	70.67	76.39	83.45
BERT-D	69.25	73.49	78.01	84.06
BERT-DA	50.48	43.15	80.13	79.35
BERT-ALL	71.73	74.45	78.54	85.11
RoBERTa-LD	71.85	68.15	71.64	82.89
RoBERTa-D	65.18	67.59	71.26	83.13
RoBERTa-DA	53.32	34.27	78.75	76.00
RoBERTa-ALL	69.80	72.70	77.50	85.48
BGE-M3-LD	65.07	68.01	74.67	83.57
BGE-M3-DA	68.21	72.49	76.23	84.29
BGE-M3-D	52.70	38.52	81.46	77.89
BGE-M3-ALL	66.93	73.23	77.03	85.37

Table 4.5 L_d quantity's impact on the pseudo-labeling optimization module

4.5.2 Impact of Pseudo-Label Optimization Module

The quality of data generated by LLM is also a crucial influencing factor. As shown in Table 4.6, when keeping L_d uniformly at 100, different synthetic data under different LLMs can affect the final experimental results. According to inference, BERT-DA should perform better after optimization by the pseudo-labeling optimization module compared to BERT-D, which has not undergone training. However, the table shows a decrease in ACC.

The reason for this is that the pseudo-labeling optimization module primarily eliminates incorrect pseudo-labels, improving the accuracy of the pseudo-labels, but this also reduces the quantity of synthetic data. In the process of this study, incorrect data is returned to the LLM as error examples for re-contextual memory, rather than being directly used for training. During

method	GPT4o	GPT4	Llama3
BERT-LD	83.45	83.45	83.45
BERT-D	84.06	83.35	84.86
BERT-DA	79.35	80.76	76.90
BERT-ALL	85.11	84.28	84.09
RoBERTa-LD	82.89	82.89	82.89
RoBERTa-D	83.13	84.72	83.00
RoBERTa-DA	76.00	76.57	75.58
RoBERTa-ALL	85.48	84.70	83.13
BGE-M3-LD	83.57	83.57	83.57
BGE-M3-DA	84.29	84.63	83.95
BGE-M3-D	77.89	80.25	77.57
BGE-M3-ALL	85.37	85.30	84.05

Table 4.6 Results of the pseudo-labeling optimization module under different LLM

the experiments, the amount of optimized BERT-DA data was only about half of the original synthetic data. Therefore, by referring to the comparison in Table 5, it can be found that, under the same quantity, the ACC of optimized labels is superior to that of entirely unoptimized labels.

Similarly, after integrating L_d, the ACC was successfully improved across the board, further proving the effectiveness of the pseudo-labeling optimization module in this study.

4.5.3 Discussion of results

method	GPT4o	GPT4	Llama3
BERT-LD	83.45	83.45	83.45
BERT-ALL	85.11	84.28	84.09
RoBERTa-LD	82.89	82.89	82.89
RoBERTa-ALL	85.48	84.70	83.13
BGE-M3-LD	83.57	83.57	83.57
BGE-M3-ALL	85.37	85.30	84.05

Table 4.7 Comparison of results

As shown in Table 4.7, using three different LLMs and only L_d, and applying the method of this study, there is a 1-2% improvement in ACC to varying degrees under the same pre-training model and the same number of labels. The results also show that the quality of synthetic data currently varies according to the performance of the LLM. Consistent with various test rankings,

data synthesized by GPT4o achieved the best results. While GPT4 has a lower maximum token acceptance than GPT4o, its performance is quite close, and when the quantity of L_d is not large, it might even outperform GPT4o. On the other hand, LLama3 lags behind the GPT series in both token acceptance and synthetic quality.

In summary, the method proposed in this study effectively improves text classification performance in few-shot scenarios.

Chapter 5

Conclusion and Future Prospects

5.1 Conclusion

The experimental findings reveal that our proposed strategy, which combines data augmentation with pseudo-labeling using large language models, is highly effective for classification tasks and the efficient training of multi-label classification models. Remarkably, our approach demonstrates improved performance even when reference labels are scarce. This confirms that, in comparison to baseline and state-of-the-art methods, one-shot and few-shot Chat-GPT annotation techniques yield better outcomes. To minimize the usage of Chat-GPT tokens and lower annotation expenses, we advise adopting the few-shot learning approach. Conversely, zero-shot pseudo-labeling is not recommended due to its poorer performance compared to manual annotation.

5.2 Future Prospects

The lack of datasets remains a challenging issue in the field of deep learning. The high cost of manual annotation and the difficulty of obtaining some datasets make it hard to quickly deploy deep learning models for certain tasks. Although large language models use an unlabelled training approach, the uncertainty and quality of generated data remain significant challenges that need to be addressed in the future.

RAG (Retrieval-Augmented Generation) is a recently proposed approach that uses specific datasets in LLMs to help build knowledge bases, thereby finding more accurate answers and optimizing the quality of generated responses. Future work will focus on the research of RAG and further reducing the need for labeled data.

Reference

[1] S. Garg and G. Ramakrishnan, "BAE: BERT-based adversarial examples for text classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6174–6181. [Online]. Available: https://aclanthology.org/2020.emnlp-main.498

[2] T. Miyato, S. ichi Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," 2018.

[3] D. S. Sachan, M. Zaheer, and R. Salakhutdinov, "Revisiting lstm networks for semi-supervised text classification via mixed objective function," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, p. 6940─6948, Jul. 2019. [Online]. Available: http://dx.doi.org/10.1609/aaai.v33i01.33016940

[4] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2004. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2004/file/96f2b50b5d3613adf9c27049b2a888c7-Paper.pdf

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[6] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, Massachusetts, USA: Association for Computational Linguistics, Jun. 1995, pp. 189–196. [Online]. Available: https://aclanthology.org/P95-1026

[7] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, “Self-training with noisy student improves imagenet classification,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 684–10 695.

[8] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang, “Unlabeled data improves adversarial robustness,” *Advances in neural information processing systems*, vol. 32, 2019.

[9] M. Pavlinek and V. Podgorelec, “Text classification method based on self-training and lda topic models,” *Expert Systems with Applications*, vol. 80, pp. 83–93, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417301665

[10] T. Kumar, J. Park, M. S. Ali, A. S. Uddin, J. H. Ko, and S.-H. Bae, “Binary-classifiers-enabled filters for semi-supervised learning,” *IEEE Access*, vol. 9, pp. 167 663–167 673, 2021.

[11] N. Zhou, N. Yao, J. Zhao, and Y. Zhang, “Rule-based adversarial sample generation for text classification,” *Neural Computing and Applications*, vol. 34, no. 13, pp. 10 575–10 586, 2022.

[12] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” *arXiv preprint arXiv:1508.05326*, 2015.

[13] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” *arXiv preprint arXiv:1606.05250*, 2016.

[14] D. Chen, J. Bolton, and C. D. Manning, “A thorough examination of the cnn/daily mail reading comprehension task,” *arXiv preprint arXiv:1606.02858*, 2016.

[15] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[16] F. Gilardi, M. Alizadeh, and M. Kubli, "Chatgpt outperforms crowd workers for text-annotation tasks," *Proceedings of the National Academy of Sciences*, vol. 120, no. 30, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1073/pnas.2305016120

[17] K. Kafle, M. Yousefhussien, and C. Kanan, "Data augmentation for visual question answering," in *Proceedings of the 10th International Conference on Natural Language Generation*, J. M. Alonso, A. Bugarín, and E. Reiter, Eds. Santiago de Compostela, Spain: Association for Computational Linguistics, Sep. 2017, pp. 198–202. [Online]. Available: https://aclanthology.org/W17-3529

[18] C. Li, X. Li, and J. Ouyang, "Semi-supervised text classification with balanced deep representation distributions," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 5044–5053. [Online]. Available: https://aclanthology.org/2021.acl-long.391

[19] D. Mekala, C. Dong, and J. Shang, "Lops: Learning order inspired pseudo-label selection for weakly supervised text classification," 2022.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[22] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, "Bge m3-embedding: Multilingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," 2024.

[23] K. Lo, L. L. Wang, M. Neumann, R. Kinney, and D. Weld, "S2ORC: The semantic scholar open research corpus," in *Proceedings of the 58th Annual Meeting of the Association*

for Computational Linguistics, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 4969–4983. [Online]. Available: https://aclanthology.org/2020.acl-main.447

[24] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf, X. Tang, D. Radev, A. F. Aji, K. Almubarak, S. Albanie, Z. Alyafeai, A. Webson, E. Raff, and C. Raffel, "Crosslingual generalization through multitask finetuning," 2023.

[25] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[26] M. Wang, Y. Liu, X. Zhang, S. Li, Y. Huang, C. Zhang, D. Wang, S. Feng, and J. Li, "Langgpt: Rethinking structured reusable prompt design framework for llms from the programming language," 2024.

[27] M. R. H. Rakib, N. Zeh, M. Jankowska, and E. Milios, "Enhancement of short text clustering by iterative classification," 2020.

[28] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf