FPT: Feature Prompt Tuning for Few-shot Readability Assessment

Anonymous ACL submission

Abstract

Prompt-based methods have achieved promising results in most few-shot text classification tasks. However, for readability assessment tasks, traditional prompt methods lack crucial linguistic knowledge, which has already been proven to be essential. Moreover, previous studies on utilizing linguistic features have shown non-robust performance in few-shot settings and may even impair model performance. To address these issues, we propose a novel prompt-based tuning framework that incorporates rich linguistic knowledge, called **F**eature **P**rompt **T**uning (FPT). Specifically, we extract linguistic features from the text and embed them into trainable soft prompts. Further, we devise a new loss function to calibrate the similarity ranking order between categories. Experimental results demonstrate that our proposed method FTP not only exhibits a significant performance improvement over the prior best prompt-based tuning approaches, but also surpasses the previous leading methods that incorporate linguistic features. Also, our proposed model significantly outperforms the large language model gpt-3.5-turbo-16k in most cases. Our proposed method establishes a new architecture for prompt tuning that sheds light on how linguistic features can be easily adapted to linguistic-related tasks.

1 Introduction

Readability assessment (RA) is the task of evaluating the reading difficulty of a given piece of text (Vajjala, 2022). It has wide applications, such as choosing appropriate reading materials for language teaching (Collins-Thompson and Callan, 2004), supporting readers with learning disabilities (Rello et al., 2012), and ranking search results by their reading levels (Kim et al., 2012).

Early works in RA mainly focused on designing handcrafted linguistic features such as word length (in characters/syllables), sentence length, and usage

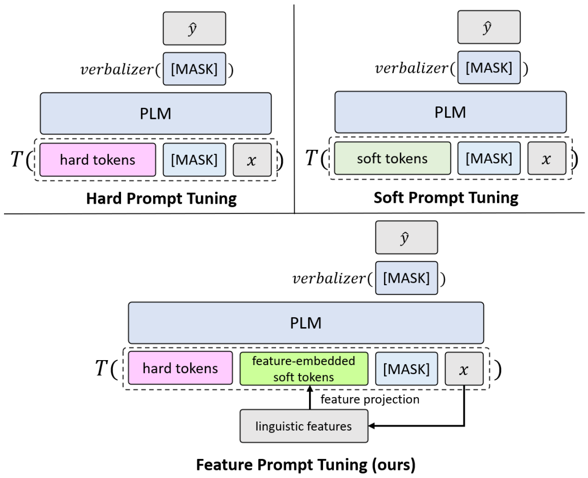


Figure 1: Comparison of previous prompt tuning frameworks and our proposed Feature Prompt Tuning (FPT). $T(\cdot)$ and $verbalizer(\cdot)$ denote the template and verbalizer function, respectively. FPT utilizes both hard and soft tokens which are projected from the linguistic features extracted from the input x.

of different difficulty-level words. In recent years, RA has been dominated by neural network-based architectures (Meng et al., 2021; Azpiazu and Pera, 2019). The key challenge of these methods is to learn a better text representation that can capture deep semantic features. Current research has also explored different ways of combining linguistic features with pretrained language models (PLMs), achieving state-of-the-art results on numerous RA datasets (Li et al., 2022; Lee et al., 2021). However, these studies have mainly focused on fine-tuning with a large amount of labelled data while only a few studies have explored few-shot settings.

Prompt-based tuning, shown to be a powerful method for the classification task in the few-shot setting, makes full use of the information in PLMs by reformulating classification tasks as cloze questions. Different prompt-based tuning strategies are illustrated in Figure 1. The hard prompt tuning applies a template with [MASK] token to the original input and maps the predicted label word to the corresponding class (Han et al., 2022; Shin et al., 2020). The performance is sensitive to the quality of template, which introduces time-consuming and labor-intensive prompt design and optimization. To address this problem, researchers propose soft prompt strategies, where continuous embeddings of trainable tokens replaces the hard template and are optimized by training (Liu et al., 2021; Lester et al., 2021).

Despite the success in a range of text classification tasks, existing prompt-based tuning methods still suffer from inferior performance in RA. This might be attributed to the lack of linguistic knowledge which has been demonstrated to play a crucial role in RA (Vajjala, 2022; Qiu et al., 2021; Li et al., 2022). Meanwhile, RA differs from general classification tasks in that there exists a notion of ranking order between classes. Our intuition behind the utilization of linguistic knowledge is that the learned representations of different levels should preserve the similarity relationship analogous to that of original linguistic features of different levels.

Motivated by the above insights, in this paper, we propose a novel prompt-based tuning method that incorporates rich linguistic knowledge, called **F**eature **P**rompt **T**uning (FPT), as shown in the bottom of Figure 1. Specifically, our methodology begins with extracting linguistic features from the text. These extracted features are subsequently embedded into feature prompts, functioning as trainable soft prompts. Contrary to the conventional prompt tuning frameworks, our model can explicitly benefit from linguistic knowledge. Furthermore, we devise a new loss function to calibrate the similarity relationships between the embedded features across different categories. Our approach is straightforward and effective, offering wide applicability to other tasks where the importance of handcrafted features is emphasized.

To verify the effectiveness of our proposed methods, we conduct extensive experiments on three RA datasets, including one Chinese data (Li et al., 2022) and two English datasets, WeeBit (Vajjala and Meurers, 2012) and Cambridge (Xia et al., 2019). By incorporating linguistic knowledge, our proposed model FPT improves significantly over other prompt-based methods. For instance, in the 2-shot setting, FPT brings a relative performance gain of 43.9% over the traditional soft prompt method on the Chinese dataset and 5.50% on English Weebit. Moreover, compared to other feature fusion methods, FPT outperforms the previous best method Projecting Feature (PF) (Li et al., 2022) by 43.19% on Chinese data and 11.55% on English Weebit data. Also, we experiment on the Large Language Model (LLM), demonstrating the superiority of our approach on RA.

We summarize our contributions as follows:

- We propose a novel prompt-based tuning framework, Feature Prompt Tuning (FPT), which incorporates rich linguistic knowledge for RA.

- We design a new calibration loss to ensure the linguistic features retain their original similarity information during optimization.

- Our experimental results show that our method outperforms other prompt-based tuning methods and effectively leverages linguistic features, leading to better and more stable performance improvements than previous approaches.

2 Related Works

2.1 Readability Assessment

Early works have explored a wide range of linguistic features as measurements for readability. Flesch (1948) performed regression over features such as average word length in syllable; Schwarm and Ostendorf (2005) trained an SVM over features including LM perplexity and syntactic tree height; Pitler and Nenkova (2008) illustrated that discourse relations can be a good predictor of readability.

Recent works largely employ deep learning approaches for RA. Several deep architectures, including BERT (Devlin et al., 2018), HAN (Yang et al., 2016), and multi-attentive RNN were applied to achieve strong performance without feature engineering (Martinc et al., 2021; Azpiazu and Pera, 2019). However, the performance of neural models tends to fluctuate a lot across different RA datasets (Deutsch et al., 2020), suggesting that relying only on neural networks might not be a robust solution for RA. Meanwhile, later works has shown that a hybrid approach combining transformer-based encoders with linguistic features can achieve even higher performance (Lee et al., 2021; Lee and Vajjala, 2022; Li et al., 2022). Lee and Lee (2023) applied a prompt-based learning based on seq2seq model, casting RA as a text-to-text task. Different from the above deep learning approach, we explicitly incorporate linguistic knowledge into our framework to boost performance.

2.2 Prompt-based Tuning

Fine-tuning PLMs have shown their prevalence in various NLP tasks. PLMs, such as BERT (Devlin et al., 2018), GPT (Radford et al., 2018), XLNet (Xia et al., 2019), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2020), have been proposed with varied self-supervised learning architectures. It has been demonstrated that larger models tend to perform better in many learning scenarios (Brown et al., 2020), which stimulated PLMs with billions of parameters to emerge.

Fine-tuning large PLMs may be prohibitive, and there exist a significant gap between pretraining tasks and downstream tasks. Prompt tuning addresses this challenge by reformulating downstream tasks as a language modeling problem and optimizing the prompt. Prompts are used to probe PLM's intrinsic knowledge to perform a task (Min et al., 2022), and various techniques of prompting have been explored to aid PLM better: hard prompt (Shin et al., 2020; Schick and Schütze, 2021), soft prompt (Lester et al., 2021; Li and Liang, 2021), verbalizer (Cui et al., 2022) and pretrained prompt tuning (Gu et al., 2021).

The effectiveness of prompt tuning has been validated in various NLP tasks, including sentiment analysis (Wu and Shi, 2022), named entity recognition (Ma et al., 2022), relation extraction (Chen et al., 2022) and semantic parsing (Schucher et al., 2021). However, the potential of prompt tuning is less explored in RA. In this work, we focus on the effectiveness of linguistic features for modeling readability, and utilize linguistic features to guide prompt tuning.

3 Background

We model RA as a text classification task. Formally, a RA dataset can be denoted as $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ is the text set and $\mathcal{Y}$ is the class set. Each instance $x \in \mathcal{X}$ consists of several tokens, $x = \{w_1, w_2, ..., w_{|x|}\}$, and is annotated with a label $y \in \mathcal{Y}$, indicating the reading difficulty.

3.1 Fine-tuning PLMs for RA

Given a PLM $\mathcal{M}$ for RA, fine-tuning methods first convert a text $x = (w_1, w_2, ..., w_{|x|})$ into an input sequence $([\text{CLS}], w_1, w_2, ..., w_{|x|}, [\text{SEP}])$. The PLM encodes this sequence into the hidden vectors $h = (h_{[\text{CLS}]}, h_1, h_2, ..., h_{|x|}, h_{[\text{SEP}]})$.

In the conventional fine-tuning, an additional classifier FC is trained on top of the [CLS] embedding $h_{[\text{CLS}]}$. This classifier produces a probability distribution over the class set $\mathcal{Y}$ through a softmax function, which can be formulated as:

$$P(\cdot|x) = \text{Softmax}(FC(h_{[\text{CLS}]})),$$

The objective of fine-tuning is to minimize the cross-entropy loss between the predicted probability distribution $P(\cdot|x)$ and the ground-truth label y:

$$\mathcal{L}_{classfication} = -\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \log P(y|x).$$

3.2 Prompt-based Tuning

Prompt-based tuning aims to bridge the gap between pretraining tasks and downstream tasks, as illustrated in Figure 1.

Hard Prompt. It typically consists of a template $T(\cdot)$, which transforms the input x into a prompt input x_{prompt}, and a set of label words V that are connected to the label space through a mapping function $\Phi : V \rightarrow \mathcal{Y}$, often referred to as the verbalizer. The prompt input contains at least one [MASK] token for the model to fill with label words.

Taking an example in RA, x_{prompt} could take the form of

$$x_{prompt} = T(x) = \text{"It is [MASK] to read: } x\text{"}.$$

In this case, the input embedding sequence of x_{prompt} is denoted as

$$(e(\text{"It is"}), e([\text{MASK}]), e(\text{"to read: "}), e(x)).$$

Soft Prompt. It replaces hard tokens in the template with trainable soft tokens $[h_1, ..., h_l]$, yielding an input embedding sequence of

$$(h_1, ..., h_l, e([\text{MASK}]), e(x)).$$

Hybrid Prompt. It combines soft tokens with hard prompt tokens T to form the input embedding sequence:

$$(h_1, ..., h_l, e(T), e([\text{MASK}]), e(x)).$$

By feeding the input embedding sequence of x_{prompt} into $\mathcal{M}$, the probability distribution over the class set $\mathcal{Y}$ is modeled by:

$$P_{\mathcal{M}}(y|x) = P_{\mathcal{M}}([\text{MASK}] = \Phi(y)|x_{prompt})$$

The learning objective of prompt-based tuning is to minimize the cross entropy loss:

$$\mathcal{L}_{classification} = -\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \log P_{\mathcal{M}}(y|x)$$

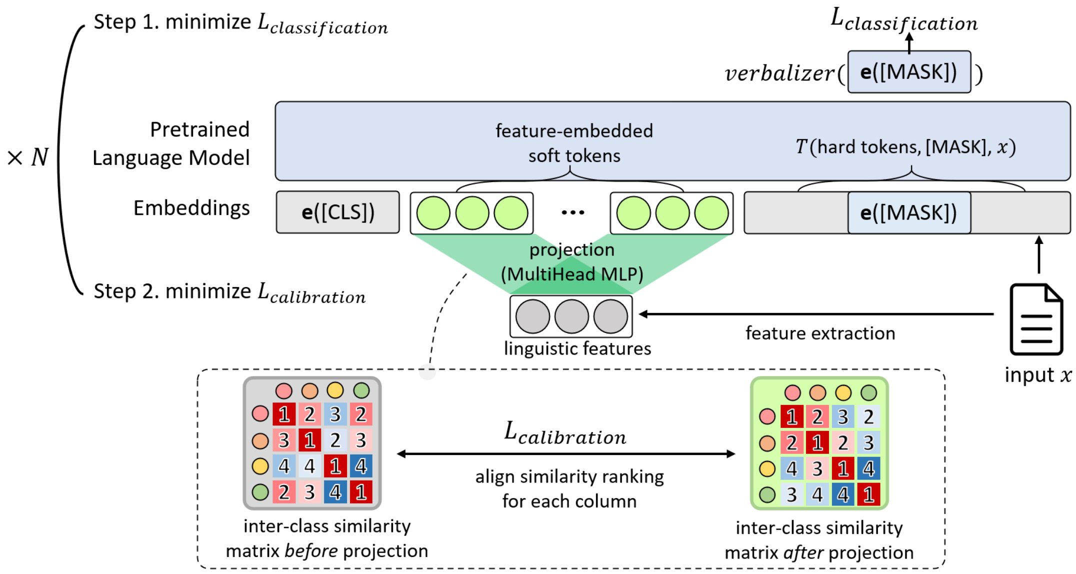


Figure 2: The architecture of the proposed Feature Prompt Tuning. Column-wise ranking orders of similarity matrices are denoted with numbers.

4 Feature Prompt Tuning

In this section, we propose a novel method for RA with prompt-based tuning, named Feature Prompt Tuning (FPT). The architecture of our model is illustrated in Figure 2. Specifically, we extract linguistic features from the texts and embed them into soft prompts. Then, we employ a loss function to calibrate the similarity relationship between embedded features of different classes. We adopt an alternating procedure to optimize the model with respect to the classification loss and calibration loss.

4.1 Feature Prompt Construction

Feature Extraction Our approach for extracting linguistic features from text is consistent with previous works (Li et al., 2022; Lee et al., 2021). For English texts, the linguistic features are extracted using the *lingfeat* toolkit (Lee et al., 2021), which includes discourse, syntactic, lexical, and shallow features. In terms of Chinese linguistic features, we directly utilize the *zhfeat* toolkit (Li et al., 2022) to extract character, word, sentence, and paragraph-level features. Specific details are provided in Appendix A. For an input text x, we denote the extracted features as f_x, which is a α-dimensional vector with α representing the number of extracted features.

Feature Embedding To incorporate linguistic knowledge into prompt-based tuning, we transform linguistic feature f_x into l distinct vectors $\{v_1, ..., v_l\}$ which function as the embeddings of soft tokens, as follows:

$$\{v_1, ..., v_l\} = \text{MultiHeadMLP}(f_x).$$

Here, MultiHeadMLP is a multi-head MLP with l output heads. Each head consists of a series of fully connected layers followed by non-linear activation functions.

The purpose of using a multi-head MLP is to allow the model to map f_x into separate vector spaces and learn multiple aspects of the linguistic features. This enables the model to better capture the relationships between different features and their contribution to RA.

Ultimately, we formulate the input embedding sequence of x_{prompt} as follows:

$$(v_1, ..., v_l, e(T), e([\text{MASK}]), e(x)).$$

This input sequence is passed through the PLM $\mathcal{M}$ to calculate $\mathcal{L}_{classification}$ loss as described in Section 3.2.

4.2 Inter-class Similarity Calibration

We denote $\mathcal{F} = \{F_{c_1}, \cdots, F_{c_n}\}$ as the collection of linguistic features for the dataset $\mathcal{D}$, which consists of n classes. Here, $F_{c_i} = \{f_{x_{i1}}, \cdots, f_{x_{ik}}\}$ signifies the extracted features of k samples which belong to i-th class. We apply average pooling to the feature embeddings of each sample in $\mathcal{F}$, resulting in a set of embedded linguistic features, denoted as $\mathcal{F}' = \{F'_{c_1}, \cdots, F'_{c_n}\}$. To gauge the similarity between any two classes F_{c_m} and F_{c_n}, we employ a pairwise approach based on cosine similarity, expressed as:

$$s_{mn} = \frac{1}{k^2} \sum_{i=1}^{k} \sum_{j=1}^{k} \cos(f_{x_{mi}}, f_{x_{nj}})$$

With the feature representation and similarity function in place, we can define our calibration objective. The fundamental intuition is that the distribution of extracted linguistic features should be preserved as much as possible. Namely, if the similarity between F_{c_m} and F_{c_n} is relatively low, the similarity between F'_{c_m} and F'_{c_n} should also be proportionately low, and vice versa. Therefore, during the training process, we devise an objective function based on a list-wise ranking loss function ListMLE (Xia et al., 2008), to maintain this initial ranking relationship.

More specifically, we compute the similarity between each pair of classes within $\mathcal{F}$ to generate the similarity matrix:

$$M = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{bmatrix}$$

Likewise, we can derive the similarity matrix M' for $\mathcal{F}'$.

We then use $\Pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$ to denote the ranking order of the columns in M, where π_i represents the ranking order of the i-th column. We obtain $\hat{M}'$ by rearranging the columns of M' according to Π:

$$\hat{M}' = \begin{bmatrix} s'_{\pi_{11}} & s'_{\pi_{12}} & \cdots & s'_{\pi_{1n}} \\ s'_{\pi_{21}} & s'_{\pi_{22}} & \cdots & s'_{\pi_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ s'_{\pi_{n1}} & s'_{\pi_{n2}} & \cdots & s'_{\pi_{nn}} \end{bmatrix}$$

Finally, we aim to minimize the following loss function for similarity calibration:

$$L_{calibration} = -\sum_{k=1}^{n} log \prod_{i=1}^{n} \frac{\exp(s'_{\pi_{ik}})}{\sum_{j=i}^{n} \exp(s'_{\pi_{jk}})}$$

4.3 Training Procedure

Training Objectives Given the dataset $\mathcal{D}$ and the linguistic feature set $\mathcal{F}$, we establish two training objectives. The primary objective is to minimize the classification loss $L_{classification}$, which is computed based on the difference between the predicted and actual class labels. The secondary objective is to calibrate the inter-class similarity of the mapped features by minimizing the loss function $L_{calibration}$ defined in Section 4.2.

Algorithm 1: Alternating Training Procedure for Feature Prompt Learning

Initialize model parameters M and feature embeddings f
for each epoch **do**
  Shuffle dataset D
  for each batch b in D **do**
    Compute $L_{classification}$ for b using M and f
    Update M by minimizing $L_{classification}$
    Compute $L_{calibration}$ for b using f
    Update f by minimizing $L_{calibration}$
  end for
end for

Alternating Training Procedure For training both loss functions, we adopt an alternating training procedure, as encapsulated in Algorithm 1. This procedure iteratively updates the model parameters and feature embeddings by minimizing the classification loss $L_{classification}$ and the similarity calibration loss $L_{calibration}$, respectively.

In each epoch, the dataset D is shuffled to ensure the model is not biased towards any particular ordering of the data. For each batch b in D, the classification loss $L_{classification}$ is computed using the current model parameters M and feature embeddings f. The model parameters M are then updated by minimizing this loss. Subsequently, the similarity calibration loss $L_{calibration}$ is computed using the updated feature embeddings f, and the feature embeddings are updated by minimizing this loss. This process is repeated for each batch in the dataset, and for each epoch. The alternating training procedure ensures that the model learns to classify the data accurately while maintaining the inter-class similarity structure of the feature space.

5 Experimental Setting

5.1 Datasets

To evaluate the effectiveness of our proposed method, we conduct experiments on one Chinese dataset and two English datasets, following the same data split as Li et al. (2022). The statistic distribution of datasets can be found in Appendix B.

ChineseLR (Li et al., 2022) is a Chinese dataset collected from textbooks of the middle and primary schools of more than ten publishers. Following the standards specified in the *Chinese Curriculum Standards for Compulsory Education*, all texts are divided into five difficulty levels.

WeeBit (Vajjala and Meurers, 2012) is often considered as the benchmark data for English RA. It was initially created as an extension of the well-known Weekly Reader corpus.

Cambridge (Xia et al., 2019) consists of reading passages from the five main suite Cambridge English Exams (KET, PET, FCE, CAE, CPE).

5.2 Baselines 1: Prompt-based Methods

For prompt-based methods, we compare with hard, soft, and hybrid prompts. To avoid the influence of verbalizers on experimental results, we adopt a soft verbalizer (Hambardzumyan et al., 2021) that employs a linear layer classifier across all prompt-based methods.

Hard Prompt (HP): We implement four manually defined templates for prompt tuning and select a template with the best performance on the development set. Details of the templates can be found in Appendix C.

Soft Prompt (SP): It replaces manually defined prompts with trainable continuous prompts. We follow the same implementation as Lester et al. (2021) and use randomly sampled vocabulary to initialize the prompts.

Hybrid Prompt (HBP): It concatenates trainable continuous prompts to the wrapped input embeddings. We adopt the implementation from Gu et al. (2022).

P-tuning: A hybrid prompt method, which replaces some tokens in manually designed prompts with soft prompts and only retains task-relevant anchor words. The soft prompts are embedded with a bidirectional LSTM and a MLP (Liu et al., 2021).

5.3 Baselines 2: Fusion Methods

We also compare with the methods fusing linguistic features and PLMs from previous studies.

SVM: Use the single numerical output of a neural model (BERT) as a feature itself, joined with linguistic features, and then fed them into SVM (Lee et al., 2021; Deutsch et al., 2020).

FT: Standard fine-tuning method without linguistic features, where the hidden representation of [CLS] token is used for classification. This baseline is for validating whether the linguistic features are indeed having a positive effect.

Concatenation (Con): Fine-tune with linguistic features, in which the linguistic features are directly concatenated to the hidden representation of the [CLS] token (Meng et al., 2021; Qiu et al., 2021).

PF: Fuse linguistic features with hidden representations of [CLS] through projection filtering (Li et al., 2022).

5.4 Implementation Details

Under the few-shot setting, we randomly sample $k = 1, 2, 4, 8, 16$ instances in each class from the training and development set. For each k-shot experiment, we sample 4 different training and dev sets and repeat experiments on each training set for 4 times. We select the best model checkpoint based on the performance on the development set, and evaluate the models on the entire test set. As for the evaluation metric, we use $accuracy$ in all experiments and take the mean values as the final results.

All our models and baselines are implemented with the PyTorch (Paszke et al., 2019) framework and Huggingface transformers (Wolf et al., 2020). We use BERT (Devlin et al., 2018) as our PLM backbone. During training, we employ the AdamW optimizer (Loshchilov and Hutter, 2019) with a weight decay of 0.01 and a warm-up ratio of 0.05. We tune the model with the batch size of 8 for 30 epochs, and the learning rate is 1e-5. All experiments are conducted with four NVIDIA GeForce RTX 3090s.

6 Results and Analysis

6.1 Comparison with Prompt-based Methods

Table 1 shows the results of our proposed method FPT and prompt-based baselines under the few-shot setting. (1) Our method FPT significantly outperforms nearly all baseline methods across all three datasets under different shots, demonstrating that our method exhibits greater robustness and adaptability to variations in data sizes and languages. (2) FTP particularly excels on the ChineseLR dataset, and it outperforms the soft prompt (SP) method by 8.41, 14.1, 10.15, 9.94 and 7.9 points under 1, 2, 4, 8, 16 shots, respectively. (3) In the task of RA, the soft prompt method generally outperforms the hard prompt. Interestingly, the hybrid prompt, a combination of both, does

k	Methods	ChineseLR	Weebit	Cambridge
1	HP	29.49(5.21)	41.83(4.72)	36.25(8.49)
	SP	31.22(4.70)	**46.61**(3.63)	41.73(8.45)
	HBP	33.51(5.19)	44.46(5.02)	42.04(9.12)
	P-tuning	33.36(4.12)	41.23(4.11)	40.36(7.15)
	FPT(ours)	**39.63**(6.38)	43.61(4.50)	**44.17**(7.12)
2	HP	28.38(8.14)	49.23(2.85)	46.88(9.31)
	SP	32.14(5.54)	52.22(4.35)	49.13(8.38)
	HBP	33.38(7.02)	52.52(2.66)	49.56(7.12)
	P-tuning	35.12(4.20)	50.71(3.87)	48.97(8.47)
	FPT(ours)	**46.24**(5.62)	**55.10**(4.04)	**59.79**(10.2)
4	HP	36.56(5.18)	53.41(4.50)	48.75(8.49)
	SP	38.78(2.83)	54.96(3.89)	49.36(9.14)
	HBP	39.81(2.67)	56.88(3.52)	50.13(8.77)
	P-tuning	38.45(3.09)	54.35(3.21)	48.85(9.64)
	FPT(ours)	**48.93**(3.21)	**57.70**(4.63)	**53.54**(7.21)
8	HP	41.21(4.83)	61.31(3.13)	55.42(6.86)
	SP	42.72(2.82)	62.02(2.67)	56.75(6.89)
	HBP	41.93(4.12)	63.37(2.02)	57.34(9.28)
	P-tuning	42.81(4.04)	61.81(3.28)	56.90(7.23)
	FPT(ours)	**52.66**(5.00)	**64.92**(2.75)	**59.38**(6.58)
16	HP	47.35(3.69)	63.75(5.41)	61.67(8.98)
	SP	47.44(2.09)	67.54(4.56)	63.77(7.43)
	HBP	47.08(3.11)	67.30(4.69)	63.98(7.34)
	P-tuning	46.26(3.19)	65.52(3.84)	62.03(9.62)
	FPT(ours)	**55.25**(2.93)	**68.19**(4.21)	**65.00**(4.25)

Table 1: Experimental results comparing with prompt-based methods. We report mean performance and the standard deviation in brackets. The best results are in bold, and the best results of previous prompt-based methods are underlined.

k	Methods	ChineseLR	Weebit	Cambridge
1	FT	28.59(4.88)	45.99(2.94)	34.17(4.33)
	SVM	25.34(3.87)	44.82(3.14)	35.31(5.23)
	Con	28.53(4.68)	43.81(3.88)	33.33(10.1)
	PF	30.13(3.99)	44.01(2.91)	35.11(9.12)
	FPT(ours)	**33.29**(4.80)	**46.67**(3.50)	**43.96**(7.09)
2	FT	22.87(7.19)	48.79(3.49)	44.17(10.4)
	SVM	23.95(9.28)	49.55(3.78)	43.99(11.0)
	Con	25.61(8.21)	49.29(2.88)	41.67(8.16)
	PF	26.12(7.21)	50.23(2.81)	41.52(7.34)
	FPT(ours)	**37.40**(4.77)	**56.03**(3.48)	**55.83**(6.72)
4	FT	36.64(5.37)	52.46(4.28)	47.50(6.29)
	SVM	37.11(6.88)	53.03(5.65)	47.58(8.67)
	Con	36.64(5.37)	52.46(4.28)	47.50(6.29)
	PF	37.13(5.11)	53.18(2.99)	48.46(4.79)
	FPT(ours)	**44.88**(3.27)	**56.17**(3.84)	**55.00**(4.86)
8	FT	40.45(2.91)	61.11(3.15)	61.46(7.81)
	SVM	40.52(3.67)	60.98(5.78)	61.55(9.10)
	Con	41.65(2.98)	58.41(3.31)	58.96(7.43)
	PF	44.00(2.86)	59.32(2.97)	55.62(10.9)
	FPT(ours)	**47.60**(3.66)	**62.40**(3.30)	**64.17**(5.95)
16	FT	45.73(4.11)	65.93(5.50)	71.04(7.97)
	SVM	46.85(3.72)	63.72(4.98)	71.22(8.15)
	Con	48.33(3.99)	64.52(4.73)	**71.46**(6.12)
	PF	48.66(3.20)	65.08(4.60)	69.38(6.79)
	FPT(ours)	**53.94**(3.16)	**68.10**(3.25)	69.17(7.77)

Table 2: Experimental results comparing with the feature fusion methods. Con means Concatenation. For a fair comparison, here FPT concatenates the feature embeddings to the original input embedding and outputs the classification logits over [CLS] embedding instead of [MASK].

not always yield better results than the standalone soft prompt. This could be attributed to the inherent challenge in designing and selecting effective hard prompts for RA. Nevertheless, as a hybrid prompt approach that integrates linguistic knowledge, our proposed method continues to exhibit robust performance, demonstrating its adaptability and effectiveness.

6.2 Comparison with Fusion Methods

Table 2 reports the experimental results comparing with fusion methods under the few-shot setting. (1) Our proposed method FPT shows a stable and significant improvement compared to the previous feature fusion methods. For instance, in the 2-shot setting, FPT outperforms the best previous fusion methods by 11.28, 5.8 and 11.66 points on ChineseLR, Weebit and Cambridge, respectively. This demonstrates our method's effectiveness in integrating linguistic features for RA. (2) Methods with linguistic features perform better than standard fine-tuning on Chinese datasets. However, it may not necessarily lead to improvement on English datasets, especially when k is increased to a sufficient amount, which indicates that simply applying linguistic features to aid in English RA is not consistently effective.

Dataset	Methods	k=2	k=4	k=8
ChineseLR	FPT	**46.24**	**48.93**	**52.66**
	-SC	40.97	46.03	50.48
	-SC and FP	25.45	36.56	40.57
Weebit	FPT	**55.10**	**57.70**	**64.92**
	-SC	52.68	56.92	63.63
	-SC and FP	48.65	53.41	61.31

Table 3: Ablation study of FPT on ChineseLR and Weebit datasets. SC represents the similarity calibration and FP means utilizing linguistic features as soft prompts.

6.3 Ablation Study

To validate the effectiveness of each component in our proposed model, we conduct ablation experiments on both English Weebit and ChineseLR datastes. Table 3 lists the results.

Our full model yields the best performance on

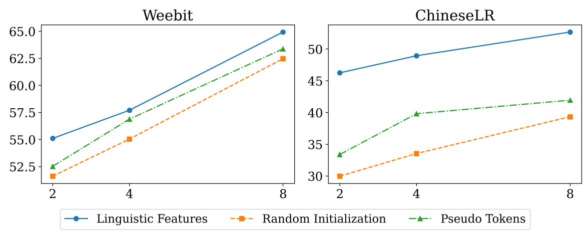


Figure 3: The comparison results of linguistic features, randomly initialized vectors and pseudo tokens.

both datasets. When removing the similarity calibration (SC) module, the performance is markedly decreased, demonstrating the necessity of retaining the linguistic features' original similarity information during optimization. We have also investigated the impact of SC by visualising the similarity difference matrix before and after applying SC, the results of which are presented in Appendix D. Moreover, further removal of the feature prompt (FP) shows a steep drop in performance (12.37 points on ChieseLR and 4.29 points on Weebit when $k = 4$), validating the effectiveness of incorporating linguistic features as soft prompts. We note that the improvement of SC and FP is more significant on the Chinese dataset compared to the English dataset, indicating that the Chinese RA task is more dependent on linguistic features.

6.4 The Significance of Linguistic Features

To further analyze whether linguistic features improve performance, in our model structure, we replace the linguistic feature vectors with randomly initialized vectors. On the other hand, we reimplement the Hybrid Prompt Tuning by utilizing pseudo tokens as soft prompts. We conduct experiments on WeeBit and ChineseLR datasets, and the comparison results are shown in Figure 3.

The performance on both datasets significantly decreases when the linguistic features are replaced with random vectors, especially on the ChineseLR dataset, where the decrease is up to 16.27%. The fewer the samples, the more severe the decline caused by the replacement, further indicating the beneficial role of linguistic features when data is insufficient. Moreover, compared to pseudo tokens, using vector-form embeddings as soft prompts requires the integration of linguistic knowledge to achieve better performance.

k	Dataset	FPT	LLM
0	Weebit	-	30.79
	Cambridge	-	43.33
	ChineseLR	-	21.67
1	Weebit	**43.61**	31.75
	Cambridge	44.17	**48.33**
	ChineseLR	39.63	-
2	Weebit	**55.10**	33.17
	Cambridge	**59.79**	54.16
	ChineseLR	46.24	-

Table 4: Comparison between our model and LLM (gpt-3.5-turbo-16k) on three datasets. k represents the number of in-context examples. Due to the limitation of context length, the experiments on Chinese dataset cannot be carried out.

6.5 Comparison with the LLM

Large language model (LLM) excels at various downstream tasks without the need for parameter adjustment. We conduct experiments on LLM utilizing the gpt-3.5-turbo-16k API, and the accuracy results are presented in Table 4. Our model with 110M parameters significantly outperforms the LLM model on the English dataset (except one sample on Cambridge). Moreover, gpt-3.5-turbo-16k is unable to perform 1-shot or 2-shot experiments on ChineseLR due to its limited context length. This underscores the necessity for research on tasks related to longer textual information in RA (Reading Comprehension and Answering).

7 Conclusion

Inspired by the solid performance of prompt tuning on classification tasks and the importance of linguistic features in the RA task, we empirically investigated the effectiveness of incorporating linguistic features into prompt tuning for RA. We convert linguistic features of the input into soft tokens and utilize the similarity calibration loss to preserve similarity relationship between classes before and after the transformation. The results show noticeable improvements over previous fusion methods and prompt-based approaches in the few-shot learning setting. The ablation study further illustrated that the proposed model benefits from linguistic features and additional similarity calibration.

Limitations

Our proposed method, which leverages the masked language model (MLM) backbone such as BERT, has demonstrated its efficacy across a variety of natural language processing tasks. Despite its strengths, we acknowledge several limitations that warrant further investigation.

Firstly, our approach exhibits constraints in processing long texts, a scenario frequently encountered in Chinese readability evaluation datasets. The inherent architecture of MLMs like BERT is optimized for shorter sequences, leading to potential performance degradation when dealing with extensive text inputs.

Secondly, while MLM-based methods are proficient in classification tasks, they often fall short in terms of interpretability of the classification outcomes. The black-box nature of these models makes it challenging to trace and understand the decision-making process, which is crucial for applications where justification of results is required.

Lastly, the success of our method is significantly contingent upon the quality of linguistic features extracted from the text. However, the extraction of high-quality linguistic features is not always guaranteed, especially in languages with rich morphology or poor data resources.

In conclusion, while our method stands as a robust approach for several NLP tasks, addressing these limitations is imperative for advancing the field and extending the applicability of MLM-based models to a broader spectrum of text analysis challenges.

Ethics Statement

Potential Risks Firstly, as a neural network-based method, the predictive outcomes of our approach should not be applied in practical applications without the involvement of human experts. This is a responsible practice for the actual beneficiaries, the learners. Secondly, as mentioned earlier, low-quality or even incorrect linguistic features can negatively impact our method. Therefore, evaluating the quality of linguistic features is essential for the efficacy of our approach.

About Computational Budget For each k-shot experiment, we conducted a total of 16 repetitions (refer to Section 5.4) for all baselines and FPT. The duration of a single experiment varies according to the size of k (approximately 20 seconds to 200 seconds), but the time consumed by different methods is almost identical.

Use of Scientific Artifacts We utilize the *lingfeat* toolkit (Lee et al., 2021) to extract linguistic features from English texts; this toolkit is publicly accessible under the CC-BY-SA-4.0 license. For extracting Chinese linguistic features, we employ the *zhfeat* toolkit (Li et al., 2022).

Use of AI Assistants We have employed ChatGPT as a writing assistant, primarily for polishing the text after the initial composition.

References

Ion Madrazo Azpiazu and Maria Soledad Pera. 2019. Multiattentive recurrent neural network architecture for multilingual readability assessment. *Transactions of the Association for Computational Linguistics*, 7:421–436.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xiang Chen, Lei Li, Ningyu Zhang, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Relation extraction as open-book examination: Retrieval-enhanced prompt tuning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2443–2448.

Kevyn Collins-Thompson and Jamie Callan. 2004. Information retrieval for language tutoring: An overview of the reap project. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, page 544–545, New York, NY, USA. Association for Computing Machinery.

Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. Prototypical verbalizer for prompt-based few-shot tuning. *arXiv preprint arXiv:2203.09770*.

Tovly Deutsch, Masoud Jasbi, and Stuart Shieber. 2020. Linguistic features for readability assessment. *arXiv preprint arXiv:2006.00377*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland. Association for Computational Linguistics.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. 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)*, pages 4921–4933, Online. Association for Computational Linguistics.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

Jin Young Kim, Kevyn Collins-Thompson, Paul N. Bennett, and Susan T. Dumais. 2012. Characterizing web content, user interests, and search behavior by reading level and topic. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, page 213–222, New York, NY, USA. Association for Computing Machinery.

Bruce W Lee, Yoo Sung Jang, and Jason Lee. 2021. Pushing on text readability assessment: A transformer meets handcrafted linguistic features. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10669–10686.

Bruce W. Lee and Jason Lee. 2023. Prompt-based learning for text readability assessment. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1819–1824, Dubrovnik, Croatia. Association for Computational Linguistics.

Justin Lee and Sowmya Vajjala. 2022. A neural pairwise ranking model for readability assessment. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3802–3813, Dublin, Ireland. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wenbiao Li, Wang Ziyang, and Yunfang Wu. 2022. A unified neural network model for readability assessment with feature projection and length-balanced loss. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7446–7457, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Linyang Li, Qi Zhang, and Xuanjing Huang. 2022. Template-free prompt tuning for few-shot NER. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5721–5732, Seattle, United States. Association for Computational Linguistics.

Matej Martinc, Senja Pollak, and Marko Robnik-Šikonja. 2021. Supervised and unsupervised neural approaches to text readability. *Computational Linguistics*, 47(1):141–179.

Changping Meng, Muhao Chen, Jie Mao, and Jennifer Neville. 2021. Readnet: A hierarchical transformer framework for web article readability analysis. *CoRR*, abs/2103.04083.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pages 186–195.

Xinying Qiu, Yuan Chen, Hanwu Chen, Jian-Yun Nie, Yuming Shen, and Dawei Lu. 2021. Learning syntactic dense embedding with correlation graph for automatic readability assessment. 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)*, pages 3013–3025, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Luz Rello, Horacio Saggion, Ricardo Baeza-Yates, and Eduardo Graells. 2012. Graphical schemes may improve readability but not understandability for people with dyslexia. In *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations*, pages 25–32, Montréal, Canada. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269.

Nathan Schucher, Siva Reddy, and Harm de Vries. 2021. The power of prompt tuning for low-resource semantic parsing. *arXiv preprint arXiv:2110.08525*.

Sarah E Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05)*, pages 523–530.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Sowmya Vajjala. 2022. Trends, limitations and open challenges in automatic readability assessment research. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5366–5377, Marseille, France. European Language Resources Association.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the seventh workshop on building educational applications using NLP*, pages 163–173.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Hui Wu and Xiaodong Shi. 2022. Adversarial soft prompt tuning for cross-domain sentiment analysis. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2438–2447.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. *Listwise Approach to Learning to Rank: Theory and Algorithm*, page 1192–1199. Association for Computing Machinery, New York, NY, USA.

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2019. Text readability assessment for second language learners. *arXiv preprint arXiv:1906.07580*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

A Details of Linguistic Features

A.1 Chinese Linguistic Features

Idx	Dim	Feature description
1	1	Total number of characters
2	1	Number of character types
3	1	Type Token Ratio (TTR)
4	1	Average number of strokes
5	1	Weighted average number of strokes
6	25	Number of characters with different strokes
7	25	Proportion of characters with different strokes
8	1	Average character frequency
9	1	Weighted average character frequency
10	1	Number of single characters
11	1	Proportion of single characters
12	1	Number of common characters
13	1	Proportion of common characters
14	1	Number of unregistered characters
15	1	Proportion of unregistered characters
16	1	Number of first-level characters
17	1	Proportion of first-level characters
18	1	Number of second-level characters
19	1	Proportion of second-level characters
20	1	Number of third-level characters
21	1	Proportion of third-level characters
22	1	Number of fourth-level characters
23	1	Proportion of fourth-level characters
24	1	Average character level

Table 5: Character features description.

Idx	Dim	Feature description
1	1	Total number of words
2	1	Number of word types
3	1	Type Token Ratio (TTR)
4	1	Average word length
5	1	Weighted average word length
6	1	Average word frequency
7	1	Weighted average word frequency

Idx	Dim	Feature description
8	1	Number of single-character words
9	1	Proportion of single-character words
10	1	Number of two-character words
11	1	Proportion of two-character words
12	1	Number of three-character words
13	1	Proportion of three-character words
14	1	Number of four-character words
15	1	Proportion of four-character words
16	1	Number of multi-character words
17	1	Proportion of multi-character words
18	1	Number of idioms
19	1	Number of single words
20	1	Proportion of single words
21	1	Number of unregistered words
22	1	Proportion of unregistered words
23	1	Number of first-level words
24	1	Proportion of first-level words
25	1	Number of second-level words
26	1	Proportion of second-level words
27	1	Number of third-level words
28	1	Proportion of third-level words
29	1	Number of fourth-level words
30	1	Proportion of fourth-level words
31	1	Average word level
32	57	Number of words with different POS
33	57	Proportion of words with different POS

Table 6: Word features description.

Idx	Dim	Feature description
1	1	Total number of sentences
2	1	Average characters in a sentence
3	1	Average words in a sentence
4	1	Maximum characters in a sentence
5	1	Maximum words in a sentence
6	1	Number of clauses
7	1	Average characters in a clause
8	1	Average words in a clause
9	1	Maximum characters in a clause
10	1	Maximum words in a clause
11	30	Sentence length distribution
12	1	Average syntax tree height
13	1	Maximum syntax tree height
14	1	Syntax tree height <= 5 ratio
15	1	Syntax tree height <= 10 ratio
16	1	Syntax tree height <= 15 ratio
17	1	Syntax tree height >= 16 ratio
18	14	Dependency distribution

Table 7: Sentence features description.

Idx	Dim	Feature description
1	1	Total number of paragraphs
2	1	Average characters in a paragraph
3	1	Average words in a paragraph
4	1	Maximum characters in a paragraph
5	1	Maximum words in a paragraph

Table 8: Paragraph features description.

A.2 English Linguistic Features

Idx	Dim	Feature description
1	1	Total number of Entities Mentions counts
2	1	Average number of Entities Mentions counts per sentence
3	1	Average number of Entities Mentions counts per token (word)
4	1	Total number of unique Entities
5	1	Average number of unique Entities per sentence
6	1	Average number of Entities Mentions counts per token (word)s
7	1	Total number of unique Entities
8	1	Ratio of ss transitions to total
9	1	Ratio of so transitions to total
10	1	Ratio of sx transitions to total
11	1	Ratio of sn transitions to total
12	1	Ratio of os transitions to total
13	1	Ratio of oo transitions to total
14	1	Ratio of ox transitions to total
15	1	Ratio of on transitions to total
16	1	Ratio of xs transitions to total
17	1	Ratio of xo transitions to total
18	1	Ratio of xx transitions to total
19	1	Ratio of xn transitions to total
20	1	Ratio of ns transitions to total
21	1	Ratio of no transitions to total
22	1	Ratio of nx transitions to total
23	1	Ratio of nn transitions to total
24	1	Local Coherence for PA score
25	1	Local Coherence for PW score
26	1	Local Coherence for PU score
27	1	Local Coherence distance for PA score
28	1	Local Coherence distance for PW score
29	1	Local Coherence distance for PU score

Table 9: Discourse features description.

Idx	Dim	Feature description
1	1	Total count of Noun phrases
2	1	Average count of Noun phrases per sentence
3	1	Average count of Noun phrases per token
4	1	Ratio of Noun phrases count to Verb phrases count
5	1	Ratio of Noun phrases count to Subordinate Clauses count
6	1	Ratio of Noun phrases count to Prep phrases count
7	1	Ratio of Noun phrases count to Adj phrases count
8	1	Ratio of Noun phrases count to Adv phrases count
9	1	Total count of Verb phrases
10	1	Average count of Verb phrases per sentence
11	1	Average count of Verb phrases per token
12	1	Ratio of Verb phrases count to Noun phrases count
13	1	Ratio of Verb phrases count to Subordinate Clauses count
14	1	Ratio of Verb phrases count to Prep phrases count
15	1	Ratio of Verb phrases count to Adj phrases count
16	1	Ratio of Verb phrases count to Adv phrases count
17	1	Total count of Subordinate Clauses
18	1	Average count of Subordinate Clauses per sentence

Idx	Dim	Feature description
19	1	Average count of Subordinate Clauses per token
20	1	Ratio of Subordinate Clauses count to Noun phrases count
21	1	Ratio of Subordinate Clauses count to Verb phrases count
22	1	Ratio of Subordinate Clauses count to Prep phrases count
23	1	Ratio of Subordinate Clauses count to Adj phrases count
24	1	Ratio of Subordinate Clauses count to Adv phrases count
25	1	Total count of prepositional phrases
26	1	Average count of prepositional phrases per sentence
27	1	Average count of prepositional phrases per token
28	1	Ratio of Prep phrases count to Noun phrases count
29	1	Ratio of Prep phrases count to Verb phrases count
30	1	Ratio of Prep phrases count to Subordinate Clauses count
31	1	Ratio of Prep phrases count to Adj phrases count
32	1	Ratio of Prep phrases count to Adv phrases count
33	1	Total count of Adjective phrases
34	1	Average count of Adjective phrases per sentence
35	1	Average count of Adjective phrases per token
36	1	Ratio of Adj phrases count to Noun phrases count
37	1	Ratio of Adj phrases count to Verb phrases count
38	1	Ratio of Adj phrases count to Subordinate Clauses count
39	1	Ratio of Adj phrases count to Prep phrases count
40	1	Ratio of Adj phrases count to Adv phrases count
41	1	Total count of Adverb phrases
42	1	Average count of Adverb phrases per sentence
43	1	Average count of Adverb phrases per token
44	1	Ratio of Adv phrases count to Noun phrases count
45	1	Ratio of Adv phrases count to Verb phrases count
46	1	Ratio of Adv phrases count to Subordinate Clauses count
47	1	Ratio of Adv phrases count to Prep phrases count
48	1	Ratio of Adv phrases count to Adj phrases count
49	1	Total Tree height of all sentences
50	1	Average Tree height per sentence
51	1	Average Tree height per token (word)
52	1	Total length of flattened Trees
53	1	Average length of flattened Trees per sentence
54	1	Average length of flattened Trees per token (word)
55	1	Total count of Noun POS tags
56	1	Average count of Noun POS tags per sentence
57	1	Average count of Noun POS tags per token
58	1	Ratio of Noun POS count to Adjective POS count
59	1	Ratio of Noun POS count to Verb POS count
60	1	Ratio of Noun POS count to Adverb POS count
61	1	Ratio of Noun POS count to Subordinating Conjunction count
62	1	Ratio of Noun POS count to Coordinating Conjunction count
63	1	Total count of Verb POS tags
64	1	Average count of Verb POS tags per sentence
65	1	Average count of Verb POS tags per token

Idx	Dim	Feature description
66	1	Ratio of Verb POS count to Adjective POS count
67	1	Ratio of Verb POS count to Noun POS count
68	1	Ratio of Verb POS count to Adverb POS count
69	1	Ratio of Verb POS count to Subordinating Conjunction count
70	1	Ratio of Verb POS count to Coordinating Conjunction count
71	1	Total count of Adjective POS tags
72	1	Average count of Adjective POS tags per sentence
73	1	Average count of Adjective POS tags per token
74	1	Ratio of Adjective POS count to Noun POS count
75	1	Ratio of Adjective POS count to Verb POS count
76	1	Ratio of Adjective POS count to Adverb POS count
77	1	Ratio of Adjective POS count to Subordinating Conjunction count
78	1	Ratio of Adjective POS count to Coordinating Conjunction count
79	1	Total count of Adverb POS tags
80	1	Average count of Adverb POS tags per sentence
81	1	Average count of Adverb POS tags per token
82	1	Ratio of Adverb POS count to Adjective POS count
83	1	Ratio of Adverb POS count to Noun POS count
84	1	Ratio of Adverb POS count to Verb POS count
85	1	Ratio of Adverb POS count to Subordinating Conjunction count
86	1	Ratio of Adverb POS count to Coordinating Conjunction count
87	1	Total count of Subordinating Conjunction POS tags
88	1	Average count of Subordinating Conjunction POS tags per sentence
89	1	Average count of Subordinating Conjunction POS tags per token
90	1	Ratio of Subordinating Conjunction POS count to Adjective POS count
91	1	Ratio of Subordinating Conjunction POS count to Noun POS count
92	1	Ratio of Subordinating Conjunction POS count to Verb POS count
93	1	Ratio of Subordinating Conjunction POS count to Adverb POS count
94	1	Ratio of Subordinating Conjunction POS count to Coordinating Conjunction count
95	1	Total count of Coordinating Conjunction POS tags
96	1	Average count of Coordinating Conjunction POS tags per sentence
97	1	Average count of Coordinating Conjunction POS tags per token
98	1	Ratio of Coordinating Conjunction POS count to Adjective POS count
99	1	Ratio of Coordinating Conjunction POS count to Noun POS count
100	1	Ratio of Coordinating Conjunction POS count to Verb POS count
101	1	Ratio of Coordinating Conjunction POS count to Adverb POS count
102	1	Ratio of Coordinating Conjunction POS count to Subordinating Conjunction count
103	1	Total count of Content words
104	1	Average count of Content words per sentence

Idx	Dim	Feature description
105	1	Average count of Content words per token
106	1	Total count of Function words
107	1	Average count of Function words per sentence
108	1	Average count of Function words per token
109	1	Ratio of Content words to Function words

Table 10: Syntactic features description.

Idx	Dim	Feature description
1	1	Unique Nouns/total Nouns (Noun Variation-1)
2	1	(Unique Nouns**2)/total Nouns (Squared Noun Variation-1)
3	1	Unique Nouns/sqrt(2*total Nouns) (Corrected Noun Variation-1)
4	1	Unique Verbs/total Verbs (Verb Variation-1)
5	1	(Unique Verbs**2)/total Verbs (Squared Verb Variation-1)
6	1	Unique Verbs/sqrt(2*total Verbs) (Corrected Verb Variation-1)
7	1	Unique Adjectives/total Adjectives (Adjective Variation-1)
8	1	(Unique Adjectives**2)/total Adjectives (Squared Adjective Variation-1)
9	1	Unique Adjectives/sqrt(2*total Adjectives) (Corrected Adjective Variation-1)
10	1	Unique Adverbs/total Adverbs (AdVerb Variation-1)
11	1	(Unique Adverbs**2)/total Adverbs (Squared AdVerb Variation-1)
12	1	Unique Adverbs/sqrt(2*total Adverbs) (Corrected AdVerb Variation-1)
13	1	Unique tokens/total tokens (TTR)
14	1	Unique tokens/sqrt(2*total tokens) (Corrected TTR)
15	1	Log(unique tokens)/log(total tokens) (Bi-Logarithmic TTR)
16	1	(Log(unique tokens))**2/log(total tokens/unique tokens) (Uber Index)
17	1	Measure of Textual Lexical Diversity (default TTR = 0.72)
18	1	Total AoA (Age of Acquisition) of words
19	1	Average AoA of words per sentence
20	1	Average AoA of words per token
21	1	Total lemmas AoA of lemmas
22	1	Average lemmas AoA of lemmas per sentence
23	1	Average lemmas AoA of lemmas per token
24	1	Total lemmas AoA of lemmas, Bird norm
25	1	Average lemmas AoA of lemmas, Bird norm per sentence
26	1	Average lemmas AoA of lemmas, Bird norm per token
27	1	Total lemmas AoA of lemmas, Bristol norm
28	1	Average lemmas AoA of lemmas, Bristol norm per sentence
29	1	Average lemmas AoA of lemmas, Bristol norm per token
30	1	Total AoA of lemmas, Cortese and Khanna norm
31	1	Average AoA of lemmas, Cortese and Khanna norm per sentence
32	1	Average AoA of lemmas, Cortese and Khanna norm per token
33	1	Total SubtlexUS FREQcount value
34	1	Average SubtlexUS FREQcount value per sentenc
35	1	Average SubtlexUS FREQcount value per token
36	1	Total SubtlexUS CDcount value
37	1	Average SubtlexUS CDcount value per sentence
38	1	Average SubtlexUS CDcount value per token
39	1	Total SubtlexUS FREQlow value
40	1	Average SubtlexUS FREQlow value per sentence
41	1	Average SubtlexUS FREQlow value per token
42	1	Total SubtlexUS CDlow value
43	1	Average SubtlexUS CDlow value per sentence
44	1	Average SubtlexUS CDlow value per token
45	1	Total SubtlexUS SUBTLWF value
46	1	Average SubtlexUS SUBTLWF value per sentence
47	1	Average SubtlexUS SUBTLWF value per token
48	1	Total SubtlexUS Lg10WF value
49	1	Average SubtlexUS Lg10WF value per sentence
50	1	Average SubtlexUS Lg10WF value per token
51	1	Total SubtlexUS SUBTLCD value
52	1	Average SubtlexUS SUBTLCD value per sentence
53	1	Average SubtlexUS SUBTLCD value per token
54	1	Total SubtlexUS Lg10CD value
55	1	Average SubtlexUS Lg10CD value per sentence
56	1	Average SubtlexUS Lg10CD value per token

Table 11: Lexico Semantic features description.

Idx	Dim	Feature description
1	1	Total count of tokens x total count of sentence
2	1	Sqrt(total count of tokens x total count of sentence)
3	1	Log(total count of tokens)/log(total count of sentence)
4	1	Average count of tokens per sentence
5	1	Average count of syllables per sentence
6	1	Average count of syllables per token
7	1	Average count of characters per sentence
8	1	Average count of characters per token
9	1	Smog Index
10	1	Coleman Liau Readability Score
11	1	Gunning Fog Count Score
12	1	New Automated Readability Index
13	1	Flesch Kincaid Grade Level
14	1	Linsear Write Formula Score

Table 12: Shallow Traditional features description.

B The statistic distribution of datasets

Dataset	WeeBit		Cambridge		ChineseLR	
Level	Passages	Avg.Length	Passages	Avg.Length	Passages	Avg.Length
1	625	152	60	141	814	266
2	625	189	60	271	1063	679
3	625	295	60	617	1104	1140
4	625	242	60	763	762	2165
5	625	347	60	751	417	3299
All	3125	245	300	509	4160	1255

Table 13: Statistics of datasets for readability assessment. Avg.Length means the average tokens per passage.

C Templates

Chinese Dataset Based on the *Chinese Curriculum Standards for Compulsory Education*, we devise the following templates:

- $T_1(\cdot)$ = 一篇第[MASK]学段的文章:
- $T_2(\cdot)$ = 这是一篇第[MASK]学段的课文:
- $T_3(\cdot)$ = 一篇第[MASK]学段的课文:
- $T_4(\cdot)$ = 一篇阅读难度为[MASK]的课文:

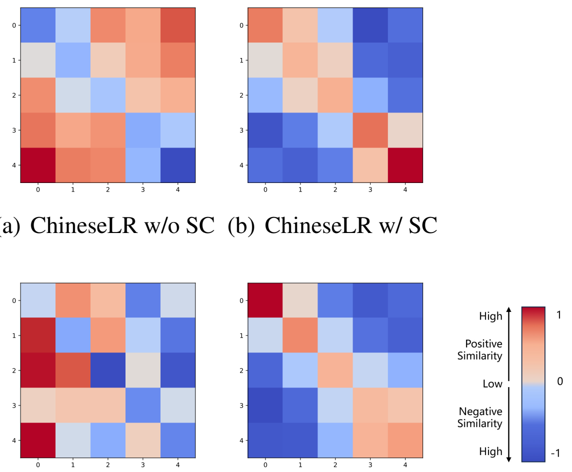
English Dataset Based on (Vajjala and Meurers, 2012), we use the following templates:

- $T_1(\cdot)$ = A [MASK] article to understand:
- $T_2(\cdot)$ = A [MASK] text to understand:
- $T_3(\cdot)$ = This is a [MASK] article to understand:
- $T_4(\cdot)$ = A [MASK] article to read:

D The Impact of Similarity Calibration

To investigate the impact of Similarity Calibration (SC), we plot the similarity difference matrices before and after linguistic feature embedding on two datasets, both with and without SC. Specifically, we calculate the similarity of linguistic features between each category before and after embedding to obtain two similarity matrices. Then we subtract the former from the latter to obtain the difference matrix. The results are shown in Figure 4, where the diagonal of the matrix represents the similarity of the linguistic features from the same category.

On both datasets, SC can effectively increase the similarity between the same and analogous categories (represented by warm colors), while reducing the similarity between distance categories (represented by cool colors). This can provide effective assistance for classification tasks.



(a) ChineseLR w/o SC (b) ChineseLR w/ SC

(c) Weebit w/o SC (d) Weebit w/ SC

Figure 4: Similarity difference matrices. We plot the difference matrices of similarity before and after linguistic feature embedding, both with and without SC. The horizontal and vertical coordinates represent the level of linguistic features. By comparing the diagonal of the matrix before and after the similarity calibration (that is, the similarity between linguistic features of the same level), the similarity between analogous categories is drawn closer.