mathematics

MDPI

Article

CLG: Contrastive Label Generation with Knowledge for Few-Shot Learning

Han Ma, Baoyu Fan, Benjamin K. Ng * and Chan-Tong Lam

Faculty of Applied Sciences, Macao Polytechnic University, Macao 999078, China; han.ma@mpu.edu.mo (H.M.); baoyu.fan@mpu.edu.mo (B.F.); ctlam@mpu.edu.mo (C.-T.L.)
* Correspondence: bng@mpu.edu.mo

Abstract: Training large-scale models needs big data. However, the few-shot problem is difficult to resolve due to inadequate training data. It is valuable to use only a few training samples to perform the task, such as using big data for application scenarios due to cost and resource problems. So, to tackle this problem, we present a simple and efficient method, contrastive label generation with knowledge for few-shot learning (CLG). Specifically, we: (1) Propose contrastive label generation to align the label with data input and enhance feature representations; (2) Propose a label knowledge filter to avoid noise during injection of the explicit knowledge into the data and label; (3) Employ label logits mask to simplify the task; (4) Employ multi-task fusion loss to learn different perspectives from the training set. The experiments demonstrate that CLG achieves an accuracy of 59.237%, which is more than about 3% in comparison with the best baseline. It shows that CLG obtains better features and gives the model more information about the input sentences to improve the classification ability.

Keywords: few-shot learning; contrastive learning; knowledge graph; natural language processing; transfer learning

MSC: 68T50

Citation: Ma, H.; Fan, B.; Ng, B.K.; Lam, C.-T. CLG: Contrastive Label Generation with Knowledge for Few-Shot Learning. *Mathematics* **2024**, *12*, 472. https://doi.org/10.3390/math12030472

Academic Editor: Faheim Sufi

Received: 5 January 2024
Revised: 24 January 2024
Accepted: 30 January 2024
Published: 1 February 2024

1. Introduction

Big data have become a considerable treasure due to their scale effect with the rapid development of storage technology in the past few years. The model achieves a huge improvement based on the big data in natural language processing (NLP) (BERT [1], RoBERTa [2], ALBERT [3], XLNet [4], BART [5], GPT-1 [6], GPT-2 [7], GPT-3 [8], T5 [9], ERNIE 1.0 [10], ERNIE 2.0 [11], and ERNIE 3.0 [12]) and computer vision (ViT [13], DALL-E [14], DALL-E 2 [15]), and Flamingo [16]). However, a few training samples cannot provide enough information as big data to train a model. Therefore, the performance of this problem is not satisfactory.

Based on [17], we would expect to find a function in the few-shot problem; when we input x, then the function can return y. x and y are the corresponding data and label. In this case, the model parameters shall ideally reach the optimal point. Training a model to reach the optimal point is similar to looking for treasure. When someone reaches the optimal point, he or she can obtain the treasure. However, in the process of searching for treasure, there are two traps: one is approximation error, and the other is estimation error. The model architecture determines the search space. When the search space does not contain the optimal point, the nearest point to the optimal point in the space is the best point that can be reached. The distance between the optimal point and the best point is the approximation error. When the amount of data is insufficient, the distance between the best point and the actual point is the estimation error. The estimation error is more prominent in the few-shot problem. That is the reason few-shot learning may produce inferior results.

In this work, we advance contrastive label generation with knowledge for few-shot learning (CLG), as shown in Figure 1. We inject the text and label with the context prompt and knowledge triplets to enhance their semantic information. Then, we input them into the CLG model. We shrink the output logits candidate space to simplify the task. Finally, we train the model with three objects.

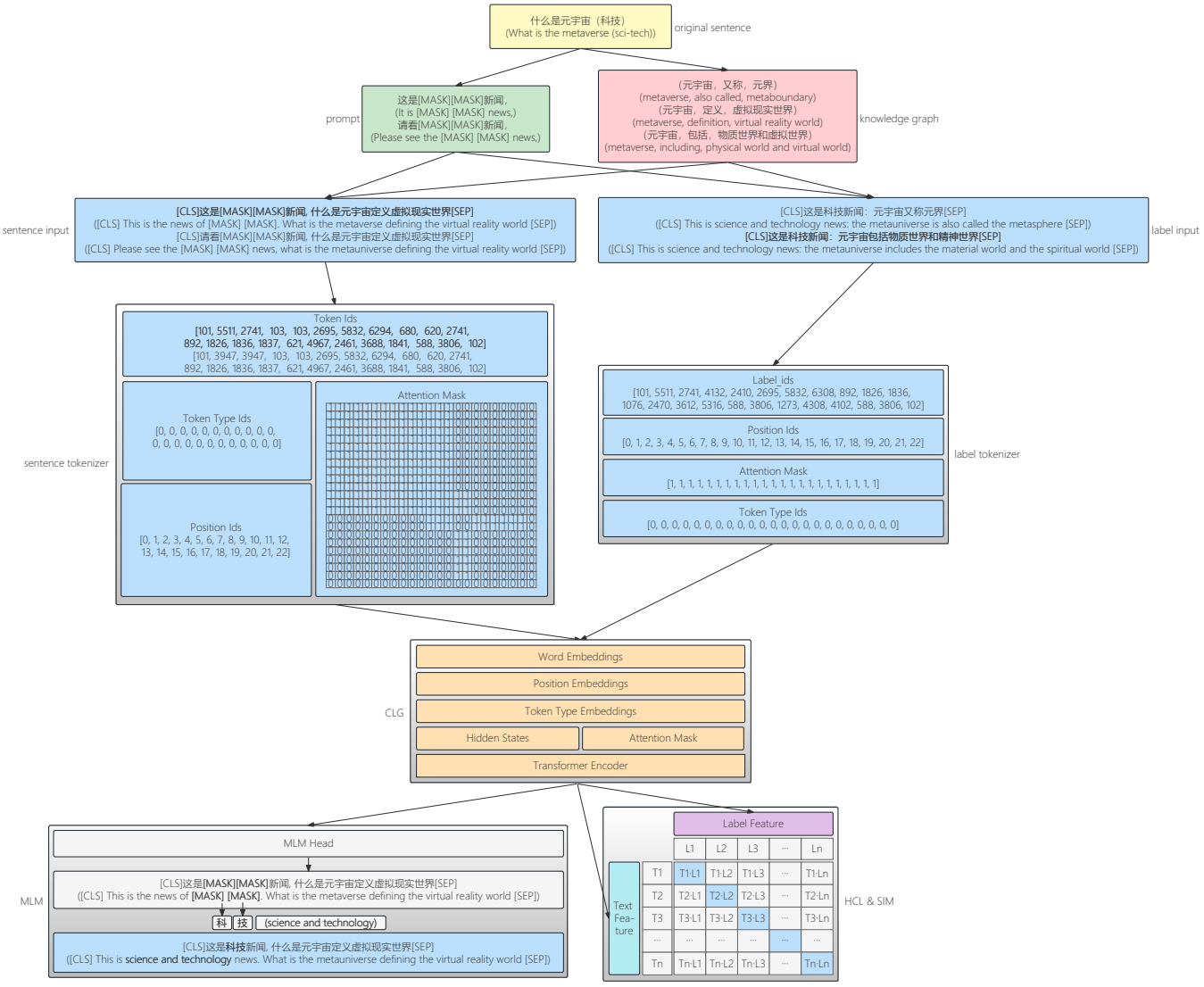

Figure 1. Overview of the structure of CLG. The text and label are augmented by the prompt and knowledge and then input into the model. In the model, there are three embedding layers to map the text and label to the vector space. Then, the encoder is used to extract the features. In the end, the features of the text and label are used to perform the downstream tasks.

The external knowledge base contains some content unrelated to the training data. In order to avoid the noise of external knowledge, we propose the label knowledge filter, which can skip the unrelated entity and relation to avoid the knowledge noise.

In the contrastive prompt and knowledge context, the text has been much enhanced, which creates a gap between the enhanced text with the unenhanced label. Therefore, the similarities will be far less. To solve this problem, we propose the label alignment method, which aligns the label with the text by introducing the prompt and knowledge.

Masked Language Modeling (MLM) searches the answers in the whole vocabulary list. There are many noise candidates in the search space. To solve this problem, we propose the MLM mask strategy, which can mask the noise candidates to narrow the search space into the label candidates.

We adopt multi-task fusion loss, HCL (Contrastive Learning with Hard negative samples), SIM (SIMilarity), and MLM. In order to obtain the common benefits of the three objective functions, all of the objective functions are weighted and summed to obtain the final loss.

The contributions of this work can be summarized as:

- Contrastive label generation (CLG). We propose contrastive label generation to align labels with the data. The input text has undergone semantic enhancement when associated with the label, but the label did not perform these operations, resulting in a lack of semantic information in the label. Therefore, we align the labels with the data and perform the same processing on the labels to reduce the difficulty of the model learning task;
- Label knowledge filter. We propose a label knowledge filter to avoid injecting noise into the text during the injection of knowledge. The traditional injection method matches the entities in the sentence with the head entities of the knowledge graph and injects the matched relations and tail entities after the center head entity. This kind of matching introduces knowledge that is not related to the sentence, so we eliminate the irrelevant noise by constructing a label knowledge filter to improve the effectiveness of the knowledge injection;
- Label logits mask. We employ the label logits mask to simplify the task. When the model outputs the logits, its candidate space is 8021 in the entire domain, while our task only needs to predict the first word of each label, which is a total of 15 words. Therefore, the first word in all of the labels is different. Hence, we constrain the output space to the task-related word range by masking the probability of other irrelevant candidate spaces so that the model can focus on outputting task-related words and improve its focus and performance;
- Multi-task fusion loss. We adopt the multi-task fusion loss. Different tasks could help the model to learn the different perspectives of the training set. Based on this, we adopt three different objects to help the model learn different perspectives of the training set to improve the final test task performance.

Extensive analysis verifies that CLG improves accuracy for the few-shot classification task.

This paper is organized as follows. Section 2 presents the literature review of data augmentation, model augmentation, and algorithm augmentation. Section 3 introduces the problem setup to clarify the problem to be resolved and presents the specific methodology of CLG concerning data, model, and algorithm. Section 4 features the experimental results and the conclusion about CLG. Section 5 includes the conclusion about CLG.

2. Related Work

Data augmentation uses some strategies to create more data. It is divided into homologous data augmentation and heterologous data augmentation. Homologous data augmentation is a method that creates data from the same source as the data set. It mainly increases new data from the training set, weakly labeled set, and unlabeled set in the same source. For the training set, it can translate [18–20], flip [20,21], shear [20], scale [19,22], reflect [23,24], crop [22,25], rotate [26], and so on to enrich the training set. For the weakly labeled or unlabeled set, there are some noisy or rare labeled data. Firstly, we can use these data to train a model and then use this model to obtain the pseudolabels for the weakly or

unlabeled data. It only gives the label to the data with high confidence until the accuracy does not increase. Refs. [27,28] automatically annotate unlabeled data with pseudolabels to improve the accuracy and robustness of the model. Ref. [29] decodes the unlabeled data feature vectors via a large language model to generate the pseudolabels. The tail data of the long-tailed problem are also similar to those of the few-shot problem. Ref. [30] proposes a data-balanced review to keep the model learning all classifications. This method certifies the effect of multi-stage knowledge transfer to improve the classification accuracy of a few training samples of the tail. Heterologous data augmentation uses similar domain data sets to train the model. The more similar the used domain data sets are, the better the result.

Prompt creates a prompting function that results in the most effective performance on the downstream task. Based on [31], there are four paradigms of modern NLP technology development. Paradigm I is fully supervised learning without a non-neural network. It mainly extracts features of data by hand-crafting. Paradigm II is fully supervised learning with a neural network. It mainly designs the architecture of models to obtain better results. Paradigm III is pre-training and fine-tuning. It mainly designs the objective function to fit the downstream task. Paradigm IV is pre-training, prompt, and prediction. It mainly uses the prompt as context to close the distance between downstream tasks and pre-training tasks. These paradigms are shown in the development of NLP. According to [31], the prompt in the pre-trained language model is seen as the parameter. So, there are two methods using the prompt in few-shot learning. On the one hand, the prompt is dynamic in fine-tuning. For instance, in prefix-tuning [32] and warp [33], they freeze all of the pre-trained language model parameters while only updating the prompt parameters. AutoPrompt [34] dynamically obtains the optimal prompt template through model training. This method can alleviate the problems of difficult construction of the manual templates and unstable effects. The dynamic prompt is common and easy to carry out. It can keep the knowledge in the model of the pre-trained task. However, its performance is limited in terms of data scale because more data usually lead to better results. On the other hand, the prompt is static in fine-tuning, such as PET-TC [35], PET-Gen [36], and LM-BFF [37]. The static prompt only trains model parameters and sees the prompt as a part of the input. It can update the parameters of the large language model with the downstream task without introducing new parameters, but it needs to design the prompt template, and its robustness is not strong. The prompt is a useful method to extend the training set and convert the downstream task form to fit the pre-training task.

Explainability is important to certify the answer reasonably. Ref. [38] proposes an unsupervised approach to generate explanations for Machine Reading Comprehension (MRC) tasks. Ref. [39] proposes a modularized pre-training model framework to enable the model to train two sub-modules independently. It also proposes a knowledge retriever and a knowledge retrieval task (KRT) to train the ability of the model to retrieve correct knowledge entities from the knowledge base (KB) to improve the ability to generate dialogue. The KG provides external information to improve the explainability of the model. There are two main methods of applying a KG: explicit knowledge representation and implicit knowledge representation. Explicit knowledge representation injects the KG triplet text into sentences. KnowBERT [40] proposes the embedding of multiple knowledge bases into large-scale models to enhance the representation. SentiLR [41] adds emotional polarity to each word in the sentence and designs a new pre-trained task, which can establish the relationship between sentence-level emotional tags and word-level emotional words. KEPLER [42] proposes a unified model for knowledge embedding and language representation. It uses a large model to jointly learn knowledge embedding (KE) and MLM objective functions to align the actual knowledge and language representation into the same semantic space. CoLAKE [43] proposes word–knowledge graphs, which are undirected and heterogeneous. These graphs can realize the representation of language and knowledge and then improve the performance of the model in downstream tasks through pre-trained tasks. K-BERT [44] uses three embedding layers to represent the input sentences. The token embedding obtains the token information. The soft-position embedding obtains the relative

position information. The segment embedding obtains the sentence pair information. K-BERT constructs the sentence tree, including the sentence and KG triplet. The triplet injects into the positions that are after the matched head entity, which enriches the information of the input to obtain better results. PK-BERT [45] attempts to resolve the few-shot problem. Firstly, it adds a prompt before the original sentence. Secondly, it injects the KG triplet (head, relation, tail) after sentences. Thirdly, in the attention matrix, the sentence tokens can see each other, but the head token can only see its relation token. The relation token can see both its head and its tail token. The tail token can only see its relation token. Finally, PK-BERT constructs the relative position ID to provide the token index information for the word and KG triplet in a sentence. The results show that PK-BERT can dig into the potential of the model, provide the knowledge, and guide the model to better complete the downstream tasks. Implicit knowledge representation embeds the KG triplets into vector space and then injects them into sentence vectors. ERNIE (THU) [46] uses TransE, which is a method of mapping the knowledge to a vector in the feature space to express and learn the KG. It first obtains the features of the token and entity, respectively; then codes and fuses them; and, finally, separates the features. This method can make the features of the token and entity contain each other's information to achieve the goal of text and knowledge fusion. For MLM tasks, ERNIE (Baidu) 1.0 [10] uses three different fine-grained mask strategies, namely, the random word-level mask strategy, entity-level mask strategy, and phrase-level mask strategy, so that the model can understand the language from different levels. ERNIE (Baidu) 2.0 [11] uses continual multi-task learning and introduces a series of tasks to let the model pay attention to words, sentences, and semantics, which can make models more knowledgeable. In addition to the advantages of the first generation and the second generation, ERNIE (Baidu) 3.0 [12] adds the universal knowledge–text prediction task to enable model learning, knowledge memory, and reasoning ability. In Ref. [47], the authors propose leveraging negative enhanced knowledge, using augmented knowledge pieces as hard negatives to improve knowledge representations.

To deal with the few-shot problem, Ref. [48] uses triplet information for data augmentation in relationship extraction, Ref. [49] uses clustering algorithms to construct pseudo-labeled data to alleviate the problem of insufficient sample size, and Ref. [50] obtains more comprehensive and accurate interaction information from other prototype networks to train the prototype network.

3. Methodology

In this section, we first introduce the problem setting and the challenges in few-shot classification. We address the challenges by proposing contrastive label generation, a knowledge graph with a block list, a logits–labels mask, multiple-task training, and some other methods.

3.1. Problem Setting

For a few-shot single-classification task, there are some sentences, S, corresponding to labels, L, in a small data set D as shown in Equations (1)–(3), where the sentences list S includes N sentences, the label list L includes C labels, and the data set D includes I pairs of a sentence and a label (S, L). For the few-shot setting, the value of I is only 240 in our experiments.

$$S = \{S_1, \ldots, S_N\} \tag{1}$$

$$L = \{L_1, \ldots, L_C\} \tag{2}$$

$$D = \{(S_1, L_1), \ldots, (S_I, L_I)\} \tag{3}$$

For a pair of one sentence S_I and one label L_I as shown in Equations (4) and (5), where n sentence tokens, t, and m label tokens, l, are included.

$$S_I = \{t_1, \ldots, t_n\} \tag{4}$$

$$L_I = \{l_1, \ldots, l_m\} \tag{5}$$

To solve this problem, we propose the method CLG that can generate label tokens to solve the classification task as shown in Equation (6), where m are sentence tokens and x and i are label tokens.

$$\{y_1, \ldots, y_i\} = CLG(\{x_1, \ldots, x_m\}) \tag{6}$$

3.2. Contrastive Label Generation

The traditional classification task converts the labels as the indexes from 0 to the classification number C, which loses the semantic meanings of the labels and limits the output space of the language model. So, we first adopt the original label phrases instead of indexes as the labels to keep the semantic information of the labels and then construct the context sentences for the labels to improve the label semantic information. In addition, we set the masked language modeling object to predict the label words, which activates the language ability of the model, aligns the downstream task with the pre-training task for transfer performance, and enlarges the output space of the task for other similar tasks. However, this makes the search space for the output larger than the classification number. To solve this problem, we adopt a logits mask, which masks all token probability, except for the label tokens, to shorten the search space.

To solve these two problems and improve the performance of the model, we propose contrastive label generation (CLG), as shown in Figure 1, which activates the language ability of the model, enlarges the output space of the model, and improves the label representation. Specifically, to make the model learn the semantic meanings of the labels and align the downstream task with the pre-training task, which is the language modeling (LM), we propose label generation to let the model output the language tokens of the label instead of the index of the classification. To improve the label feature representation, we consider that the task has its candidate classification list; it would be easy to perform a classification task via the elimination method. So, we present the contrastive label to make both the label tokens obtain better representations in the feature space and compare the candidates to make the task easy to perform.

CLG makes the model output the language tokens of the label instead of the index of the classification and makes the model learn the semantic meanings of the labels and enhance the output space of the model.

The classification task is similar to answering the choice question. When we make choices, we always have some alternative answers. It could be easier to answer the question when we consider the difference between choices. So, in this case, we compare the similarity of each input sentence with the labels and select the most similar label as the prediction answer.

Inspired by CLIP [51], we use the text and label pairs to calculate the similarity. In the training phase, the form is almost the same as the common pre-training phase as shown in Table 1. t_i means the index of the text. l_i means the index of the label. Each line has a number 1, indicating the correct label for the text. The only difference in the common pre-training phase is the use of sentences with two different prompts; however, in the training phase, labels with prompt prefixes are used.

Table 1. Training inputs matrix. The key to contrastive learning is to construct positive samples and negative samples. In this task, positive samples are correct text–label pairs that represent the value 1, and negative samples are incorrect text–label pairs that represent the value 0.

	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	l_{11}	l_{12}	l_{13}	l_{14}	l_{15}
t_1	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
t_2	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
t_3	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
t_4	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
t_5	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
t_6	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
t_7	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
t_8	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0

3.3. Logits Mask and Label Mask

MLM is a head that is used to output raw hidden states. It uses a special token [MASK] to let the model predict the word that is masked. This method can shorten the distance between the downstream task and the pre-trained task to improve the effect of transfer learning.

When carrying out the MLM task, the original method randomly masks some single words, but there are many phrases in sentences. Masking only one word cannot provide the whole meaning. So, the Whole Word Masking (WWM) strategy is proposed to solve this question. If one word in a phrase has been masked, then it also masks the whole phrase.

The search space of the MLM object is the vocabulary size, which, in our experiment, is 8021, but the label words are much fewer than the vocabulary words. So, a natural idea is to shorten the search space. In our task, there are 15 phrases of the labels in the data set. In the pre-training stage, without the MLM logits mask, the task needs to search in the whole vocabulary space. However, we found that the first words of all labels are different. Therefore, it only needs to predict the first word, and then the subsequent words of the labels can be auto-completed. Hence, with the MLM logits mask, the task only needs to search the 15 words that are the first words of the 15 label phrases. So, this method can effectively shorten the search space and complete the task more easily.

MLM labels mask is also used to avoid unrelated words in the vocabulary. This can shrink the choices space of the task. It can be viewed as solving multiple choice questions by elimination. For example, when we perform a generative task, the standard method will choose some words from the whole vocabulary list. The labels mask will also select some words from part of the vocabulary list that only includes the first words of the 15 label phrases. This could greatly reduce the number of alternative words and simplify the task.

MLM mask indexes are the token ID 103 of the special token [MASK]. It can avoid unuseful information and let the model focus on the importance of the task.

3.4. Knowledge-Enhanced Label

The KG can provide semantic information for the model. It is a huge semantic network with many head entities, relation edges, and tail entities about people, things, and objects. The KG contains explicit knowledge, which is comprised the relations between entities and the attributes of entities. This can provide the knowledge for sentences. Therefore, it is logical, reasonable, and interpretable. The effectiveness is knowledge-driven and depends on the quantity and quality of the KG.

The KG usually represents the head entities, relation edges, and tail entities as triplets [h,r,t]. It can identify the phrases of sentences in the KG, and then inject the related relations and tails of the phrases into the sentence. The KG may not exactly match the domain of the data set, so the KG block list is used to avoid injecting the knowledge noise into the input sentence. If there are many triplets injected into the sentences, the algorithm will randomly sample the candidates to make sure that it will not obtain too much knowledge to influence the semantic information of the original sentences. Inspired by K-BERT [44]

and CoLAKE [43], we use the KG and divide the sentence include two parts. One is the word graph; the other is the KG. The word graph is a fully connected word semantic network. In such a network, each word can be visible to others. They can be calculated by the model to map the word token to the embedding vector and then obtain the semantic representation. The KG is a knowledge semantic network composed of nodes and edges. The nodes mean the entities, and the edges mean the relations between the entities. If a phrase in its sentence can match a head entity in the KG, then the corresponding relation and tail entity of the head entity will be injected into the sentence. After that, the model will map the word tokens in the KG to the vectors in the feature representation space. In contrastive learning, we inject the KG triplets into the labels and add the prompt prefix before the labels to align with the text. Then, we input them into the model to obtain the feature vectors of the labels.

3.5. Data Preprocessing

Due to the few-shot problem setup, we can only use a few training samples to train the model. The task is about news title classification. The training set includes 240 news titles and the corresponding 15 classifications. So, effectively using it is a challenge. Given that the challenge is the data scale, the natural idea is to obtain more data to train. Considering the gap between the downstream and the pre-training task, we use the prompt to convert the downstream task form into the pre-training task form. This allows the model to better understand the task and improve the performance. Based on the prompt, the sentences differ in prefixes but convey the same semantics. So, they all have the same label. It is an easy but effective method for achieving data augmentation as shown in the following Equation (7).

$$D_{fsl} = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{n=1}^{N} (p_i s_j t_n) \tag{7}$$

The D_{fsl} means the few-shot learning data set, P_i means the i-th prompt, S_j means the j-th symbol, T_n means the n-th sentence, I is the total number of prompts, J is the total number of symbols, and N is the total amount of training data. This way can create $I * J * N$ data for training, which can significantly enrich the training set.

3.6. Prompt Prefix

The prompt consists of templates and symbols, which can shorten the search space of the answers and prompt their retrieval. Since the downstream task may not be the same as the pre-trained task, the prompt can convert downstream data into pre-trained data, which helps to improve the effect of transfer learning. It can stimulate the potential power of the large language models and shorten the distance between the downstream objective and the pre-trained objective, as shown in the following example.

It is [MASK][MASK] news.

Inspired by the GPT series [10–12], we use the prompt as the prefix template to augment data. Specifically, we use different prefix templates before sentences. Each prefix template can create a set of input sentences. Through this method, we can obtain more data to train the model.

The prompt gives models more insights to let models generate output that better align with the task scenario. We manually craft and compare some prompts to obtain the results.

As shown in Table 2, the token IDs are the results of the tokenizer tokenizing the Token Word. Some special tokens need to be explained. Token [CLS] means the head of the sentence. It can represent the whole sentence. Token [MASK] means the word that is masked. It masks the sentence label that the model needs for prediction. Token [SEP] means the end of the sentence.

Table 2. Input sentence. The input words are tokenized to the token IDs.

Type	Inputs
Token Word	ZH: [CLS]这是[MASK][MASK]新闻，什么是元宇宙定义虚拟现实世界[SEP] (EN: [CLS]It is [MASK][MASK] news, What is metaverse define virtual reality world [SEP])
Token IDs	[101, 5511, 2741, 103, 103, 2695, 5832, 119, 680, 620, 2741, 892, 1826, 1836, 1837, 621, 4967, 2461, 3688, 1841, 588, 3806, 102]

When labels are determined, the main difference between templates is how to reasonably build prefix templates for two-character Chinese labels through natural language. Finally, we manage to pick some prompt templates that positively impact our results.

3.7. Multi-Task Fusion Loss

HCL loss [52] is concerned with contrastive learning with hard negative samples. It calculates the similarity of the samples to determine the label of the data. This can close the distance between positive samples and, at the same time, push away the distance between the negative samples.

$$\mathcal{L}_h = -\sum_{i=1}^{N} log \frac{e^{[f(x_i)f(y_i^+)^T]/\tau}}{e^{[f(x_i)f(y_i^+)^T]/\tau} + \sum_{j=1}^{M} e^{[f(x_i)f(y_j^-)^T]/\tau}} \tag{8}$$

In Equation (8), x means the input sentence, y^+ means the positive label of the input sentence, y^- means the negative label of the input sentence, i means the i-th of the batch, j means the j-th of the negative sentences, the function $f()$ maps the text to the vector, τ means the dynamic temperature, N means the batch size, and M means the number of negative classes.

The SIM loss calculates the similarity between the text feature vectors and the label feature vectors. It uses cross-entropy to calculate loss and updates the model weights.

$$\mathcal{L}_{s1} = -\sum_{i=1}^{N} log \frac{e^{[f(x_i)f(y_i^+)^T]/\tau}}{\sum_{j=0}^{M} e^{[f(x_i)f(y_j)^T]/\tau}} \tag{9}$$

$$\mathcal{L}_{s2} = -\sum_{i=1}^{N} log \frac{e^{[f(y_i^+)f(x_i)^T]/\tau}}{\sum_{j=0}^{M} e^{[f(y_j)f(x_i)^T]/\tau}} \tag{10}$$

$$\mathcal{L}_s = (\mathcal{L}_{s1} + \mathcal{L}_{s2})/2.0 \tag{11}$$

In Equations (9)–(11), y means the sentence label, and the other symbols are the same as above.

The MLM loss function converts the downstream task into the pre-trained task, which can improve the effectiveness of the performance of transfer learning.

$$\mathcal{L}_m = -\sum_{i=1}^{N} log \frac{e^{f(x_i)_{[MASK]} \cdot g(y_i)}}{\sum_{j=0}^{M} e^{f(x_i)_{[MASK]} \cdot g(y_j)}} \tag{12}$$

In Equation (12), the function $g()$ maps the label to the vector, $[MASK]$ means the index of the mask in the sentence, and the other symbols are the same as above.

$$\mathcal{L} = w_h * \mathcal{L}_h + w_s * \mathcal{L}_s + w_m * \mathcal{L}_m \tag{13}$$

The total loss calculates the weighted sum of the above three loss functions. In Equation (13), $\mathcal{L}_h$, $\mathcal{L}_s$, and $\mathcal{L}_m$ are the losses of the HCL, SIM, and MLM. The w_h, w_s, and w_m, which are set manually, are the weights of the HCL, SIM, and MLM objective functions.

3.8. Training Algorithm

The algorithm is the strategy used to search for the best parameters in the search space. Algorithm augmentation can provide a better direction for reaching the optimal point in each iteration.

The warmup is a method that lets the learning rate begin from a small value in the initial stage and grow to normal after the warmup period. It needs to set the warmup epochs. In the warmup epochs period, the parameters of the model will move more slowly. In this case, the gradient descent direction is at a small step length in the warmup stage, and it is more likely to avoid falling into the local optimal solution.

The learning scheduler is used to operate and process the learning rate. It can calculate the loss of the evaluation set. When the loss cannot decrease, the learning rate scheduler will decrease the learning rate to try more directions for updating.

Inspired by physical experiments, the algorithm simulates the process of observing the cooling of solids and gradually reduces the internal energy through annealing, so that the internal state reaches the ground state from the stable state. The simulated annealing (SA) algorithm is a method that can weaken the difference in each gradient at the initial training stage so that the model has more exploration directions and the overfitting of model training and model parameters falling into the local optimal solution are avoided. The simulated annealing algorithm keeps the temperature slowly down at some rate. It is possible to jump out from the local optimum solution.

The early stop is another trick that can end the training in advance under certain conditions. Such conditions are generally key indicators of the experiment, such as preparation rate, recall rate, F1, and loss. The first three are that the larger the numerical value, the better the model effect. Therefore, in this case, the condition for leaving early is that when the numerical value is still less than or equal to the highest value in the past within a certain number of times, the training ends. Correspondingly, the smaller the loss value, the better the effect of the model. Therefore, in this case, the condition for the early stop is that when the value is still greater than or equal to the lowest value in the past within a certain number of times, the training ends. Here, a certain number of times is a hyper-parameter, which can be given manually.

3.9. Pseudocode

The pseudocode for the implementation of CLG is shown in the Algorithm 1, and it can be summarized as follows. First, load the text and label from the data set. Second, add the prompt and inject the KG into the text and label. Third, tokenize the text and label from natural language to token ID. Fourth, input the tokens to embeddings and encoder layers and obtain the feature vectors of text and label. Fifth, calculate the similarity of the feature vectors of the text and label. Sixth, predict the token IDs of the position $[MASK]$ in sentences by the MLM head. Finally, calculate the final loss.

Algorithm 1 CLG Pseudocode

```
text, label ← load_data(dataset)
for t, l in zip(text, label) do
    t_p ← add_prompt(t)
    l_p ← add_prompt(l)
    t_pk ← inject_kg(t_p)
    l_pk ← inject_kg(l_p)
end for
for t_input, l_input in zip(t_pk, l_pk) do
    text_feature ← get_feature(t_input)
    label_feature ← get_feature(l_input)
    hcl_logits ← text_feature_vector@label_feature_vector.T/hcl_temperature
    sim_logits ← text_feature_vector@label_feature_vector.T/sim_temperature
    mlm_logits ← mlm_head(text_sequence_output)
    index_label ← index of the label_ids in each batch
    onehot_label ← index_label in each batch
    pos_logits ← hcl_logits * onehot_label
    neg_logits ← hcl_logits * (1 − onehot_label)
    hcl_loss ← −log(pos_logits/(pos_logits + neg_logits)).mean()
    text_loss ← cross_entropy(sim_logits, onehot_label).mean()
    label_loss ← cross_entropy(sim_logits.T, onehot_label.T).mean()
    sim_loss ← (text_loss + label_loss)/2.0
    mlm_loss ← cross_entropy(mlm_logits, label_ids)
    loss ← w_hcl * hcl_loss + w_sim * sim_loss + w_mlm * mlm_loss # w_hcl, w_sim, w_mlm are the
    weights of the losses
end for
```

4. Experiment

4.1. Data Set

The TNEWS data set is a Chinese short text classification data set about news titles. It is divided into 15 categories, including technology, entertainment, automobiles, tourism, finance, education, international, real estate, e-sports, military, stories, culture, sports, agriculture, and stocks. The whole data set contains 73,360 pieces of data, and each piece of data contains a label, label description, sentence, keywords, and ID.

In FewCLUE [53], only a few training samples are used to achieve the classification task. There are 240 pieces of data in the training set, 240 pieces of data in the evaluation set, and 20,000 pieces of data in the test set.

In this work, we use the same data set as FewCLUE [53]. So, there are 15 labels, and each label has 16 pieces of sentences in the training set (15-way–16-shot), which are used for the few-shot learning. The training set size is 240, which is for the few-shot setting. The validation set size is 1200, which is for the feedback of the training phase and saves the model weights when the accuracy is better. The test set size is 2010, which is used to show the task performance of the model.

4.2. Inputs

Our experiments are on 15 categories of the text classification task. Note that the experiment does not use full data. In the training phase, each category has only 16 pieces of data, which is the few-shot setting.

4.2.1. Input Token IDs

This method converts the input token words to token IDs by the tokenizer. The input includes the prompt that transfers the current task to the pre-training task for training models, and the KG triplet enhances the sentence information. We choose the prompt with the best accuracy as the label prompt, as shown in Table 3.

Table 3. Label prompts experiment. The bold number is the best accuracy in the experiment. In this task, we select the best prompt as the label prompt to perform the continued experiments.

No.	Label Prompt	Acc. (%)
1	zh: 这是[MASK][MASK]新闻 en: This is [MASK] news	19.167
2	zh: 看看[MASK][MASK]新闻 en: See [MASK] News	21.563
3	zh: [MASK][MASK]新闻即将播放 en: [MASK] News coming soon	21.458
4	zh: [MASK][MASK]新闻来啦 en: Here comes the [MASK] news	**25.677**
5	zh: 这条新闻的类别是[MASK][MASK] en: The category of this news is [MASK]	14.896
6	zh: 这条新闻是[MASK][MASK]类别的 en: This news is in the category of [MASK]	25.625
7	zh: 我认为这是[MASK][MASK]新闻 en: I think this is [MASK] news	23.854
8	zh: 这是[MASK][MASK]新闻吧 en: Is this [MASK] news	22.604
9	zh: 这是[MASK][MASK]新闻吗 en: Is this [MASK] news	19.74
10	zh: 这是什么新闻？[MASK][MASK]新闻 en: What news is this? [MASK] News	23.229

There are 10 prompts and 10 symbols with their accuracies, as shown in Tables 4 and 5. In this experiment, we select, respectively, the top 5 prompts and the top 2 symbols of the group to expand the training set by 10 times.

Table 4. Text prompts experiment. The bold number is the best accuracy in the experiment. In this task, we select the top 5 prompts as the text prompts to perform the continued experiments.

No.	Text Prompt	Acc. (%)
1	zh: 这是[MASK][MASK]新闻 en: This is [MASK] news	23.698
2	zh: 看看[MASK][MASK]新闻 en: See [MASK] News	**27.604**
3	zh: [MASK][MASK]新闻即将播放 en: [MASK] News coming soon	**28.333**
4	zh: [MASK][MASK]新闻来啦 en: Here comes the [MASK] news	**26.198**
5	zh: 这条新闻的类别是[MASK][MASK] en: The category of this news is [MASK]	**31.25**
6	zh: 这条新闻是[MASK][MASK]类别的 en: This news is in the category of [MASK]	**29.74**
7	zh: 我认为这是[MASK][MASK]新闻 en: I think this is [MASK] news	24.74
8	zh: 这是[MASK][MASK]新闻吧 en: Is this [MASK] news	18.854
9	zh: 这是[MASK][MASK]新闻吗 en: Is this [MASK] news	24.948
10	zh: 这是什么新闻？[MASK][MASK]新闻 en: What news is this? [MASK] News	25.521

Table 5. Text symbols experiment. The bold number is the best accuracy in the experiment. In this task, we select the top 2 symbols as the text symbols to perform the continued experiments.

No.	Prompt	Acc. (%)
1	,	25.052
2	。	**27.24**
3	?	22.187
4	:	25.938
5	;	24.427
6	、	25.729
7	!	**30.26**
8	-	25.365
9	——	25.417
10	—>	27.031

4.2.2. Knowledge Graph

CN-DBpedia [54] is a large-scale general-domain structured encyclopedia KG that includes 5,168,865 pairs of head entities, relation edges, and tail entities.

Due to fewer pieces of information in the short text of the classification task, we use KG triplets to enhance the sentence head entity. This gives the model more information about the sentences and improves the explainability. For example, there is a sentence and a KG triplet, as shown in Table 6. For token type IDs, it is a sentence usually with two types of tokens, [0: the first sentence, 1: the second sentence]. Type 0 represents the original sentence tokens, including the head entity tokens. Type 1 represents the relation edge tokens and tail entity tokens in the KG triplet. In our task, we construct only one sentence that includes the original sentence and KG triplets to perform the continuous experiments. So, the final sequence is an all-zero sequence.

For position IDs, it is numbered in the order of the tokens in the sentence, but we need to note that it first tokenizes the original sentence from 0 to the length of the sentence minus 1. Then, it adopts the relative position code to tokenize KG triplets. Specifically, we add the KG positions, which are the indexes after the end word of the head entities, and the relation edge indexes are the next index at the end of the token of the head entities. The tail entity indexes are the next index of the end of the token of the relation edges. Finally, we combine all of them as the position IDs' input.

For the label, as the same text, we also add the prompt and inject KG relations and tails into the label words to construct label sentences. The label sentences have the label token IDs, label attention mask, and label type IDs.

We convert the input text from words to IDs. Those are unfriendly to humans but friendly to models, which means we can allow the model to understand what we want to say in natural language.

4.2.3. Attention Mask

The input IDs consist of original sentences, entities, and relations of the KG. To avoid the knowledge token's interference in the sentence meaning, the relation edges can only be seen by their corresponding head and tail entities, and the tail entities can only be seen by the corresponding relation edges. The relation edges and tail entities cannot be seen by other original sentence tokens and other head entities.

The attention mask is also called the attention matrix, which can determine whether the tokens in a sentence can be seen by each other. The standard is that all tokens in the original sentence can see each other. If the first entity is identified in the original sentence, the relational edge and tail entity in the KG triplet will be injected into the end of the sentence as the second sentence and input into the model together. Based on [44], the relational edges and tail entities injected into the sentence are independent of the tokens of the non-head entities in the original sentences, so we use the attention mask attention matrix to achieve it. When two tokens see each other, their intersection position in the

two-dimensional table is 1. On the contrary, when two tokens do not see each other, their intersection position in the two-dimensional table is 0, as shown in Table 7.

Table 6. The inputs converted from words to IDs.

Inputs	Data
Text	什么是元宇宙 (What is metaverse)
Label	科技 (sci-tech)
prompt	这是[MASK][MASK]新闻 (It is [MASK] news)
symbol	， (,)
KG	（元宇宙，又称，元界）（元宇宙，定义，虚拟现实世界） (metaverse, define, virtual reality world)
Text Token	[CLS]这是[MASK][MASK]新闻，什么是元宇宙定义虚拟现实世界[SEP] ([CLS] It is [MASK] news, what is metaverse define virtual reality world [SEP])
Text Token IDs	[101, 5511, 2741, 103, 103, 2695, 5832, 6294, 680, 620, 2741, 892, 1826, 1836, 1837, 621, 4967, 2461, 3688, 1841, 588, 3806, 102]
Text Type IDs	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Text Position IDs	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]
Label Token	[CLS]这是科技新闻：元宇宙或称元界[SEP] ([CLS] This is technology news: metaverse is also called meta world [SEP])
Label Token IDs	[101, 5511, 2741, 4132, 2410, 2695, 5832, 6308, 892, 1826, 1836, 2360, 4143, 892, 3806, 102]
Label Token Type IDs	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Label Position Ids	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

4.3. Training Strategy

There are some strategies for models to update the parameters effectively.

The warmup can increase the learning rate from a certain small value to the set value so that the model can explore more gradient update directions at the initial training stage and prevent the model from falling into local optimization. The learning rate scheduler decreases the learning rate until certain epochs, where a designated metric, such as loss not decreasing or accuracy not increasing, indicates a halt in improvement. A simulated annealing gradient can decrease the learning rate. We use simulated annealing to let the model explore more gradient update directions in the initial training stage to prevent the model from falling into local optimization. When the performance of models cannot improve or match certain conditions, then the models will stop learning. Dropout can be used as a feature augmentation method for training the model. In the training process, the weight of each neuron in the model has a certain probability p to set it to zero, so that the neuron cannot have an impact on the neural network. This method can reduce the dependence between neurons in the network; that is, it does not rely too much on some local features, thus enhancing the generalization ability of the model.

4.4. Benchmark

In the FewCLUE [53] benchmark test, the data set was randomly disrupted, and the entire data set was divided into smaller data sets, including the training set, evaluation set, and test set. The training set consists of 15 categories and 16 samples of each category, also known as 15-way–16-shot.

The model is trained by using the corpus, which is the same source as the task data. So, using such a model can help it understand the data and improve its performance The inference ability is better than the models with other sources of the data set. In the model,

the most important parts are the embeddings and encoder. The embedding layers are used to map the token IDs to the vectors, and the encoder layer is used to extract the features.

Table 7. The attention mask matrix shows that the relation tokens can only be seen by their own head entity tokens and tail entity tokens. The value 1 means the two words can be seen by each other and the value 0 means the words can not be seen by each other. The tail entity tokens can only be seen by the relation tokens. The two hashes ## indicate that it is a suffix following some other words.

Attention Mask	[CLS]	这 (It)	是 (is)	[MASK]	[MASK]	新 (news)	##闻	,	什 (What)	##么	是 (is)	元 (metaverse)	##宇	##宙	定 (definition)	义 (definition)	虚 (virtual)	##拟	现 (reality)	##实	世 (world)	##界	[SEP]
[CLS]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
这 (It)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
是 (is)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
[MASK]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
[MASK]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
新 (news)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
##闻	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
,	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
什 (What)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
##么	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
是 (is)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
元 (metaverse)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1
##宇	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1
##宙	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1
定 (definition)	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	1	1	1
义 (definition)	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	1	1	1
虚 (virtual)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
##拟	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
现 (reality)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
##实	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
世 (world)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
##界	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
[SEP]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

4.5. Baseline Line

BERT [1] is the first large-scale model that is pre-trained in big data. BERT has a good ability for generalization and can reuse the parameters of models to obtain a better result in the downstream task. RoBERTa [2] is a model that uses bigger data, more parameters, and some other tricks to pre-train a model based on BERT. GPT [8] is a big model that first uses the generative task to pre-train the model and obtains a better result in few-shot learning and zero-shot learning. Fine-tune is a method that first pre-trains in a big data set and then trains in a small data set to transfer the knowledge from big data to small data by model parameters. Few-shot uses a few training samples to train the model. Zero-shot uses no training data and just directly makes inferences. PET [55] uses prompt and MLM head to convert the task from a classification to generative task. Then, the downstream task form is similar to the pre-training task, which can improve the effect of transfer learning and the performance of the model. LM-BFF [37] automatically creates the prompt prefix. In the training phase, it constructs demonstrations to perform few-shot learning. P-tuning [56] transforms the problem of constructing the prompt into a continuous parameter optimization problem. The non-natural language is used to construct the prompt, which makes the space of the hypothesis of the prompt larger and more likely to find the best prompt. This method is simple and effective, giving GPT the ability of natural language understanding. EFL [57] transforms the downstream task into an implication task and a binary classification problem. It constructs sentence pairs and labels to judge whether the sentence pair is the implication. This method reduces the difficulty of the task. At the same time, the method transforms the downstream task into the pre-training task, which improves the effect of transfer learning. ERNIE 1.0 [10] is pre-trained by masking the knowledge entity, which gives the model more knowledge of words and phrases. In these results, our method CLG reaches the top of the list. It uses a smaller model but obtains better accuracy than other methods. especially when using ERNIE methods. This proves the effectiveness of our method.

4.6. Evaluation Metric

The confusion matrix is an important and common metric for measuring the classification performance of the models. The confusion matrix divides the results into four situations, as shown in Table 8.

Table 8. Calculate the evaluation metric through the confusion matrix.

Confusion Matrix		Ground Truth	
		True	**False**
Prediction	Positive	TP	FP
	Negative	FN	TN

They are True Positive (*TP*), False Positive (*FP*), False Negative (*FN*), and True Negative (*TN*) in the confusion matrix. The accuracy is shown in Equation (14). Similar to other studies, accuracy is the metric of our experiments.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{14}$$

4.7. Results and Discussion

There are some methods performing the same task that can compared to CLG, as shown in Table 9. For intuitive comparison, we put all of the readiness rates in one bar chart, as shown in Figure 2.

Our method, CLG, compares 10 methods, including zero-shot, which is without any training data and just evaluates the model, and fine-tuning models, which use training data to provide the models with more information about the task and domain to help the model improve its ability. In addition, the bigger the base model, the more data are used, and the better the performance of the model. CLG reaches better accuracy than other methods. This shows that our method can improve the understanding and generation ability of the model by contrastive label generation with a knowledge graph for few-shot learning.

Table 9. The experimental result shows our method CLG achieves the best accuracy with other baselines. The bold accuracy represents the best accuracy in the experiment.

No.	Methods	Acc. (%)
1	Zero-Shot RoBERTa [2]	25.3
2	Zero-Shot GPT [8]	37.0
3	Fine-Tuning RoBERTa [2]	49.0
4	Fine-Tuning ERIEN 1.0 [10]	51.6
5	EFL [57]	52.1
6	LM-BFF [37]	53.0
7	P-tuning RoBERTa [56]	54.2
8	PET [55]	54.5
9	P-Tuning ERIEN 1.0 [10]	55.9
10	EFL ERIEN 1.0 [10]	56.3
11	PET ERIEN 1.0 [10]	56.4
12	CLG (Ours)	**59.2**

A heatmap is a visual tool to show the performance of the models. The diagonal line represents the ground truth, and the block color from white to red represents the accuracy from low to high.

The results of model classification are generally similar to the ground truth as shown in Figure 3, but some have high classification accuracies, and some have low classification accuracies. This may be due to the difference between the data of the different labels and the data of the other labels. The accuracy rate of label classification that is easy to distinguish is high, and the accuracy rate of label classification that is easy to confuse is low. To solve this problem, we can enhance the data of some labels with low accuracies, so that the model can better learn the features of such data.

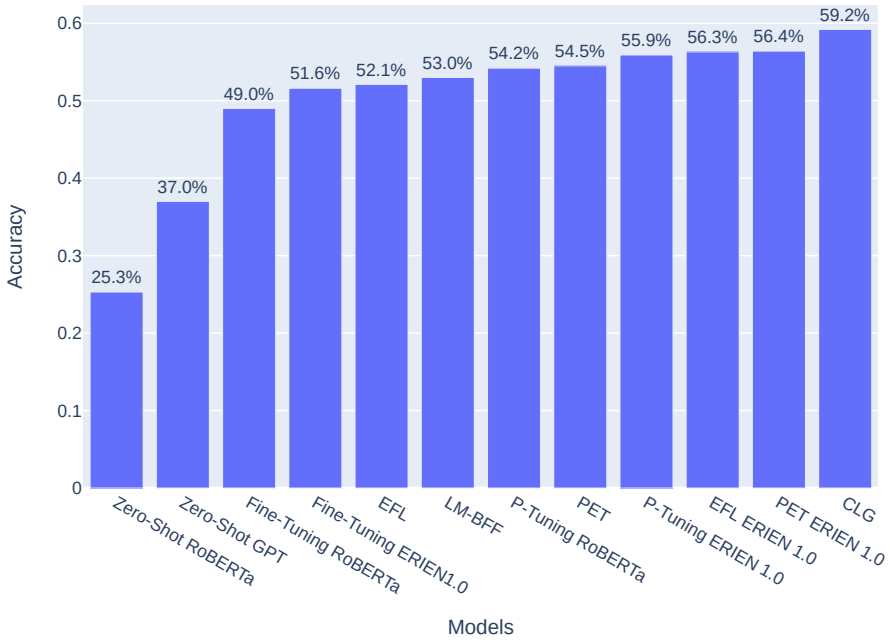

Figure 2. The accuracies of the models.

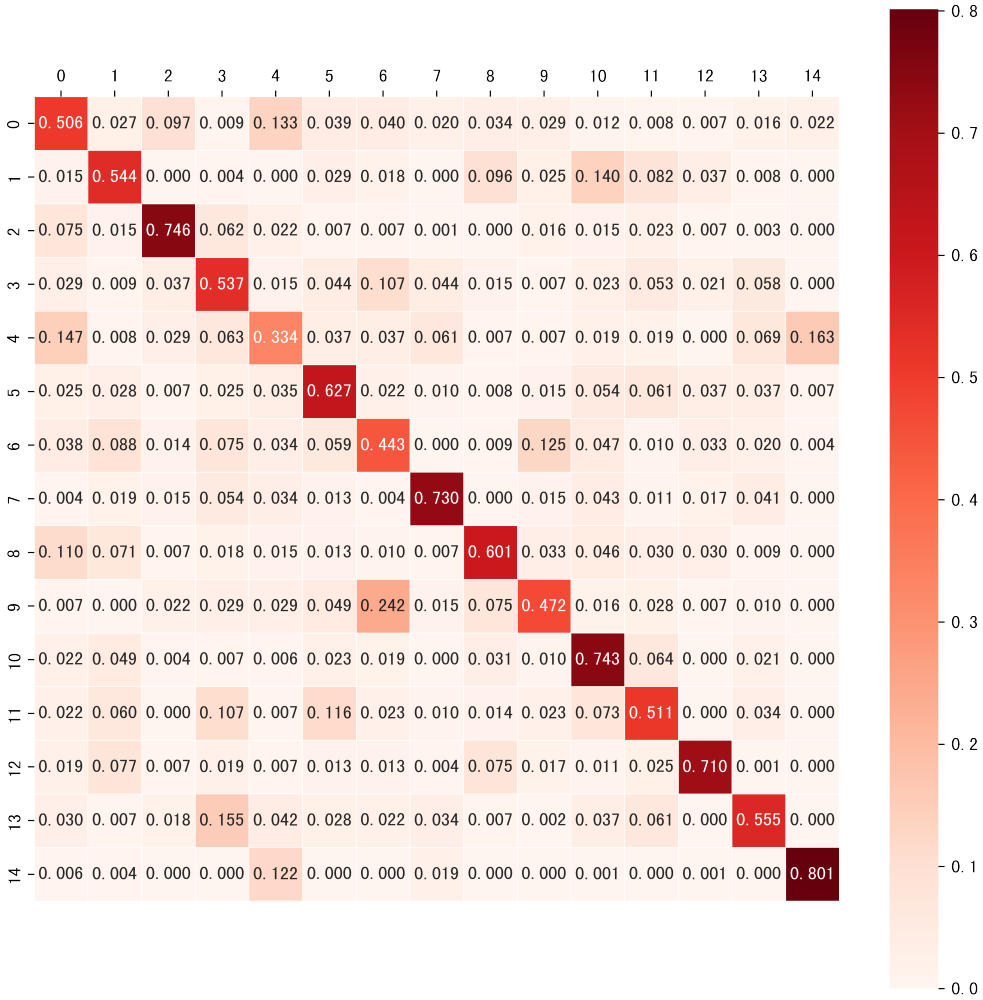

Figure 3. The heatmap shows the classification ability of each class.

4.8. Ablation Study

As the CLG consists of several key techniques, we probe and compare their effectiveness in this section. We use the control parameters method to remove each key part of the experiment, as shown in Table 10.

Table 10. The ablation experiment shows the contributions of each module in CLG. The bold accuracy represents the best accuracy in the experiment.

No.	Function	Acc. (%)
1	Prompts and Symbols	52.344
2	Block List	58.513
3	Text KG	58.029
4	Label KG	58.638
5	Logits and Labels Mask	58.663
6	Warmup	55.978
7	Learning Rate Scheduler	58.967
8	HCL SA	58.219
9	SIM SA	58.847
10	Early Stop	58.992
11	Dropout	59.032
12	Loss Weight	57.610
13	CLG	**59.237**

For the ablation experiment, if the accuracy of a removed function is lower than the CLG accuracy, then it means that the removed relative function is useful to yield better results in CLG. In contrast, if the accuracy of a removed function is higher than the CLG accuracy that means the removed relative function is negative for the performance of the model. Thus, we should remove it. As shown in Table 10, we can see each function has a positive effect on the result and they are needed. The prompts and symbols convert the input task form, which improves the transfer learning effect. The block list avoids the unrelated triplet, which can prevent the noise. The label KG makes the label sentences align with the text sentences form, which reduces the difference between the text and label and improves the contrast learning effect. The logits and labels mask strategy avoids unimportant tokens, which narrows the search scope. The warmup trains the model with a small learning rate at the initial stage of training and then gradually restores the predetermined value so that the model can be fully explored in the early stage of training to find the appropriate gradient descent direction. The learning rate scheduler reduces the learning rate when the training slows down so that the model can approach the optimal point. The HCL SA and SIM SA are simulated annealing algorithms that help the model obtain a better update direction at the initial training stage. The early stop ends the training in advance under certain conditions, which avoids overfitting. Dropout inputs the same sentences twice to obtain additional different training features. The loss weight is the weight of the three losses, which makes the effective loss obtain more weight and improves the model's effect.

5. Conclusions

In this work, we propose CLG, a simple and efficient method that uses contrastive label generation with knowledge for few-shot learning.

CLG adds the prompt and injects knowledge into text and labels. Then, it tokenizes them to input token IDs. After that, it inputs the text and label to the model, including the embedding layers and encoder layer. In this phase, the text and label become feature vectors from token IDs. Then, it calculates the similarity between text features and label features in contrastive learning and predicts the masked word to obtain the logits and loss. Finally, it calculates the evaluation metric by logits and updates the model weights by loss.

CLG can obtain better feature vectors and not only augment data but also simplify the task. It can improve explainability and avoid knowledge noise. It can align the label with the text and shrink the search space. Apart from that, it can learn from the different

perspectives of the training data. The experimental results demonstrate that the CLG is effective and can help to resolve the few-shot problem.

Although it is effective in resolving the few-shot problem, some situations need to be considered. Contrastive learning calculates the similarity scores between the feature vectors. It is worth finding a reasonable similarity function to achieve better features. Constructing the prompt manually is an easy and explicit method for creating the prompt and symbol. It is easy and effective to achieve, but the search area is small and limited, which may make it difficult to obtain the optimal solution. Matching the phrase in sentences with the head of triplets is an effective method for injecting the KG into sentences. However, there are some unrelated triplets that sentences should not inject into sentences. Although we set a block list to avoid matching the unrelated triplets, the performance also depends on the quality of the KG. The more homologous sources of the data and KG are, the better the models are.

In the future, we can explore further in these directions. Although our method performed well in many models at the time, the accuracy is still below 60%. We analyze that this is because the knowledge learned by the model from small samples is very limited, and the diversity of the samples in the test set is due to the training set. Therefore, we can consider exploring in further research how large the data volume needs to be, based on the model, to achieve good indicator values. In addition, we are also considering further improving the algorithm to enhance the efficiency of the model in learning samples.

Author Contributions: Conceptualization, H.M.; methodology, H.M.; software, H.M.; validation, H.M. and B.F.; formal analysis, H.M.; investigation, H.M. and B.F.; resources, B.K.N.; data curation, H.M. and B.F.; writing—original draft preparation, H.M.; writing—review and editing, H.M., B.F., B.K.N. and C.-T.L.; visualization, H.M. and B.F.; supervision, B.K.N. and C.-T.L.; project administration, B.K.N.; funding acquisition, B.K.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Macao Polytechnic University via grant number RP/ESCA-02/2021.

Data Availability Statement: Publicly available data sets were analyzed in this study. These data can be found here: https://github.com/CLUEbenchmark/FewCLUE/tree/main/datasets/tnews, accessed on 30 April 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
2. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
3. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv* **2019**, arXiv:1909.11942.
4. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. Xlnet: Generalized autoregressive pretraining for language understanding. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 670–681.
5. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv* **2019**, arXiv:1910.13461.
6. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. *Improving Language Understanding by Generative Pre-Training*; OpenAI: San Francisco, CA, USA, 2018.
7. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
8. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.
9. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551.
10. Sun, Y.; Wang, S.; Li, Y.; Feng, S.; Chen, X.; Zhang, H.; Tian, X.; Zhu, D.; Tian, H.; Wu, H. Ernie: Enhanced representation through knowledge integration. *arXiv* **2019**, arXiv:1904.09223.

11. Sun, Y.; Wang, S.; Li, Y.; Feng, S.; Tian, H.; Wu, H.; Wang, H. Ernie 2.0: A continual pre-training framework for language understanding. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 8968–8975.
12. Sun, Y.; Wang, S.; Feng, S.; Ding, S.; Pang, C.; Shang, J.; Liu, J.; Chen, X.; Zhao, Y.; Lu, Y.; et al. ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation. *arXiv* **2021**, arXiv:2107.02137.
13. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
14. Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; Sutskever, I. Zero-shot text-to-image generation. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 8821–8831.
15. Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv* **2022**, arXiv:2204.06125.
16. Alayrac, J.B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; et al. Flamingo: A visual language model for few-shot learning. *arXiv* **2022**, arXiv:2204.14198.
17. Wang, Y.; Yao, Q.; Kwok, J.T.; Ni, L.M. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.* **2020**, *53*, 1–34. [CrossRef]
18. Benaim, S.; Wolf, L. One-Shot Unsupervised Cross Domain Translation. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 2104–2114.
19. Lake, B.M.; Salakhutdinov, R.; Tenenbaum, J.B. Human-level concept learning through probabilistic program induction. *Science* **2015**, *350*, 1332–1338. [CrossRef] [PubMed]
20. Shyam, P.; Gupta, S.; Dukkipati, A. Attentive Recurrent Comparators. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
21. Qi, H.; Brown, M.; Lowe, D.G. Learning with Imprinted Weights. *arXiv* **2017**, arXiv:1712.07136.
22. Zhang, Y.; Tang, H.; Jia, K. Fine-Grained Visual Categorization using Meta-Learning Optimization with Sample Selection of Auxiliary Data. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
23. Edwards, H.; Storkey, A. Towards a Neural Statistician. *arXiv* **2016**, arXiv:1606.02185.
24. Kozerawski, J.; Turk, M. CLEAR: Cumulative LEARning for One-Shot One-Class Image Recognition. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
25. Qi, H.; Brown, M.; Lowe, D.G. Low-Shot Learning with Imprinted Weights. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
26. Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; Wierstra, D. Matching Networks for One Shot Learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3637–3645.
27. Xie, Q.; Luong, M.T.; Hovy, E.; Le, Q.V. Self-Training With Noisy Student Improves ImageNet Classification. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 10684–10695. [CrossRef]
28. Sun, Z.; Fan, C.; Sun, X.; Meng, Y.; Wu, F.; Li, J. Neural semi-supervised learning for text classification under large-scale pretraining. *arXiv* **2020**, arXiv:2011.08626.
29. Kahn, J.; Lee, A.; Hannun, A. Self-training for end-to-end speech recognition. In Proceedings of the ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 7084–7088.
30. Fan, B.; Liu, Y.; Cuthbert, L. Improvement of DGA Long Tail Problem Based on Transfer Learning. In *Computer and Information Science*; Lee, R., Ed.; Springer International Publishing: Cham, Switzerland, 2023; pp. 139–152. [CrossRef]
31. Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv* **2021**, arXiv:2107.13586.
32. Li, X.L.; Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv* **2021**, arXiv:2101.00190.
33. Hambardzumyan, K.; Khachatrian, H.; May, J. WARP: Word-level Adversarial ReProgramming. *arXiv* **2021**, arXiv:2101.00121.
34. Shin, T.; Razeghi, Y.; Logan IV, R.L.; Wallace, E.; Singh, S. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv* **2020**, arXiv:2010.15980.
35. Schick, T.; Schütze, H. 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, online, 19–23 April 2021; pp. 255–269. [CrossRef]
36. Schick, T.; Schütze, H. Few-Shot Text Generation with Pattern-Exploiting Training. *arXiv* **2020**, arXiv:2012.11926.
37. Gao, T.; Fisch, A.; Chen, D. Making Pre-trained Language Models Better Few-shot Learners. *arXiv* **2020**, arXiv:2012.15723.
38. Cui, Y.; Liu, T.; Che, W.; Chen, Z.; Wang, S. Teaching machines to read, answer and explain. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2022**, *30*, 1483–1492. [CrossRef]
39. Qin, L.; Xu, X.; Wang, L.; Zhang, Y.; Che, W. Modularized Pre-training for End-to-end Task-oriented Dialogue. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2023**, *31*, 1601–1610. [CrossRef]
40. Peters, M.E.; Neumann, M.; Logan, R.L.; Schwartz, R.; Joshi, V.; Singh, S.; Smith, N.A. Knowledge Enhanced Contextual Word Representations. In Proceedings of the EMNLP, Hong Kong, China, 3–7 November 2019.
41. Ke, P.; Ji, H.; Liu, S.; Zhu, X.; Huang, M. SentiLR: Linguistic Knowledge Enhanced Language Representation for Sentiment Analysis. *arXiv* **2019**, arXiv:1911.02493.

42. Wang, X.; Gao, T.; Zhu, Z.; Liu, Z.; Li, J.; Tang, J. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *arXiv* **2019**, arXiv:1911.06136.
43. Sun, T.; Shao, Y.; Qiu, X.; Guo, Q.; Hu, Y.; Huang, X.; Zhang, Z. CoLAKE: Contextualized Language and Knowledge Embedding. *arXiv* **2020**, arXiv:2010.00309.
44. Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Ju, Q.; Deng, H.; Wang, P. K-BERT: Enabling Language Representation with Knowledge Graph. *arXiv* **2019**, arXiv1909.07606.
45. Ma, H.; Ng, B.K.; Lam, C.T. PK-BERT: Knowledge Enhanced Pre-trained Models with Prompt for Few-Shot Learning. In *Computer and Information Science*; Lee, R., Ed.; Springer International Publishing: Cham, Switzerland, 2023; pp. 31–44. [CrossRef]
46. Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; Liu, Q. ERNIE: Enhanced language representation with informative entities. *arXiv* **2019**, arXiv:1905.07129.
47. Bai, J.; Yang, Z.; Yang, J.; Guo, H.; Li, Z. KIN ET: Incorporating Relevant Facts into Knowledge-Grounded Dialog Generation. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2023**, *31*, 1213–1222. [CrossRef]
48. Liu, J.; Qin, X.; Ma, X.; Ran, W. FREDA: Few-Shot Relation Extraction Based on Data Augmentation. *Appl. Sci.* **2023**, *13*, 8312. [CrossRef]
49. Yang, L.; Huang, B.; Guo, S.; Lin, Y.; Zhao, T. A Small-Sample Text Classification Model Based on Pseudo-Label Fusion Clustering Algorithm. *Appl. Sci.* **2023**, *13*, 4716. [CrossRef]
50. Ma, J.; Cheng, J.; Chen, Y.; Li, K.; Zhang, F.; Shang, Z. Multi-Head Self-Attention-Enhanced Prototype Network with Contrastive–Center Loss for Few-Shot Relation Extraction. *Appl. Sci.* **2024**, *14*, 103. [CrossRef]
51. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning transferable visual models from natural language supervision. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 8748–8763.
52. Robinson, J.; Chuang, C.Y.; Sra, S.; Jegelka, S. Contrastive Learning with Hard Negative Samples. In Proceedings of the International Conference on Learning Representations, Virtual, 3–7 May 2021.
53. Xu, L.; Lu, X.; Yuan, C.; Zhang, X.; Xu, H.; Yuan, H.; Wei, G.; Pan, X.; Tian, X.; Qin, L.; et al. Fewclue: A chinese few-shot learning evaluation benchmark. *arXiv* **2021**, arXiv:2107.07498.
54. Xu, B.; Xu, Y.; Liang, J.; Xie, C.; Liang, B.; Cui, W.; Xiao, Y. CN-DBpedia: A Never-Ending Chinese Knowledge Extraction System. In Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Arras, France, 27–30 June 2017.
55. Schick, T.; Schütze, H. Exploiting Cloze Questions for Few-Shot Text Classification and Natural Language Inference. *arXiv* **2020**, arXiv:2001.07676.
56. Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; Tang, J. GPT Understands, Too. *arXiv* **2021**, arXiv:2103.10385.
57. Wang, S.; Fang, H.; Khabsa, M.; Mao, H.; Ma, H. Entailment as few-shot learner. *arXiv* **2021**, arXiv:2104.14690.