HFGNN-Proto: Hesitant Fuzzy Graph Neural Network-Based Prototypical Network for Few-Shot Text Classification

Xinyu Guo 1, Bingjie Tian 2 and Xuedong Tian 1,3,*

1 School of Cyber Security and Computer, Hebei University, Baoding 071002, China; guoxinyu0813@outlook.com
2 International Education College, Hebei Finance University, Baoding 071051, China; tianbingjie@sina.com
3 Hebei Machine Vision Engineering Research Center, Hebei University, Baoding 071002, China
* Correspondence: xuedong_tian@126.com

Abstract: Few-shot text classification aims to recognize new classes with only a few labeled text instances. Previous studies mainly utilized text semantic features to model the instance-level relation among partial samples. However, the single relation information makes it difficult for many models to address complicated natural language tasks. In this paper, we propose a novel hesitant fuzzy graph neural network (HFGNN) model that explores the multi-attribute relations between samples. We combine HFGNN with the Prototypical Network to achieve few-shot text classification. In HFGNN, multiple relations between texts, including instance-level and distribution-level relations, are discovered through dual graph neural networks and fused by hesitant fuzzy set (HFS) theory. In addition, we design a linear function that maps the fused relations to a more reasonable range in HFGNN. The final relations are used to aggregate the information of neighbor instance nodes in the graph to construct more discriminative instance features. Experimental results demonstrate that the classification accuracy of the HFGNN-based Prototypical Network (HFGNN-Proto) on the ARSC, FewRel 5-way 5-shot, and FewRel 10-way 5-shot datasets reaches 88.36%, 94.45%, and 89.40%, respectively, exceeding existing state-of-the-art few-shot learning methods.

Keywords: few-shot text classification; multi-attribute relations; graph neural networks; hesitant fuzzy set; prototypical networks

1. Introduction

In recent years, the great success of deep learning has promoted the development of multitudinous fields such as computer vision and natural language processing [1–3], but the effectiveness of deep learning models relies on a large amount of labeled data. The generalization ability of deep learning models is severely limited when labeled data are scarce. Humans, on the other hand, have the ability to learn quickly and can easily build awareness of new things with just a few examples. This significant gap in learning between machine learning models and humans inspires researchers to explore few-shot learning (FSL) [4].

Inspired by the human-learning process, researchers proposed a meta-learning strategy for FSL, which utilizes the distribution of similar tasks to learn how to identify unseen classes accurately and efficiently with a small amount of training data. A cross-task meta-learner learns from multiple similar tasks and provides better initialization for unseen classes based on the knowledge acquired from prior experience. One of the typical meta-learning methods is the Prototypical Network [5], which computes the class prototypical representation of the support set and classifies the query sample to the nearest prototype.

Prototypical Network [5] and its variants [6–9] have been widely used for few-shot text classification tasks. Different from the Prototypical Network that computes class prototypes and query sample embeddings separately, MLMAN [6] interactively encodes text based
on the matching information between the query and support sets at the local and instance levels. Gao et al. [7] proposed a hybrid attention-based prototypical network that employs instance-level attention and feature-level attention to highlight important instances and features, respectively. Sun et al. [8] improved the Prototypical Network by using feature level, word level, and instance level multi cross attention. Geng et al. [9] proposed the Induction Network that induces a better class-level representation using a dynamic routing algorithm [10]. Although the above methods have considered the intra-class similarity of the support set and the relations between support and query, they ignore inter-class dissimilarity and the relations between query samples. Furthermore, these methods only measure instance-level relations while neglecting other substantive relations. Due to the complexity and diversity of textual forms, simple relations have difficulty describing the true connection between texts and even introduce additional noise to the model.

Instead, we measure the commonalities and differences among all samples in the task from multiple aspects. Inspired by the Distribution Propagation Graph Neural Network [11], we introduce the distribution-level information of one sample to all other support samples into few-shot text classification. To better explore relations information, a dual-graph structure consisting of instance graphs and distribution graphs is adopted in HFGNN. The instance graph models instance-level relations and directional relations based on instance features. The distribution graph aggregates instance-level relations to model distribution-level relations and the distance relations. The relations in our model represents the similarity between samples at multiple levels. However, similarity is a fuzzy concept, which is not clearly defined. To solve the problem of multiple fuzzy relations, we introduce HFS [12] theory, which can handle multi-attribute decision-making problems well for the HFGNN model. We design the membership functions corresponding to the multi-attribute relations for the comprehensive evaluation to avoid the loss of relations information. In addition, we use a linear function that can further enhance stronger relations and weaken weaker relations to help the model generate more reasonable graph structures and provide inductive biases for the enhancement of instance features. Finally, a prototypical network takes the instance features generated by HFGNN as input and quickly classifies query samples. HFGNN-Proto adopts an episodic strategy [13] for meta-training in an end-to-end manner. It has a stronger generalization ability and can adapt to new classes without retraining.

In summary, our main contributions are summarized as follows:

1. We propose an HFGNN-Proto model that comprehensively considers multiple substantive relations between texts for few-shot text classification. Subsequent ablation experiments demonstrate the effectiveness of multi-attribute relations.

2. To ensure the integrity of multi-attribute relations, we develop a new hesitant fuzzy strategy to fuse all relations into a more precise relational representation.

3. Considering the noise impact of fully connected graph on information transfer, we design a linear function to provide inductive biases for the transfer of relations in graph neural networks.

4. To verify the effectiveness of our model, we conduct extensive experiments on the ARSC and FewRel datasets. Experimental results demonstrate that the proposed HFGNN-Proto model achieves a significant improvement over other few-shot methods.

2. Related Work

2.1. Few Shot Learning

Early studies mainly applied fine-tuning [14] and data augmentation [15] to alleviate the overfitting problem caused by insufficient training data but achieved unsatisfactory results. In the meta-learning strategy [16–19], transferable knowledge that can guide the learning of models are extracted from various tasks, so that the model has the ability of learning to learn. Current meta-learning methods mainly include optimization-based methods [18–20] and metric learning [5,13,21]. Some representative few-shot learning methods and their corresponding descriptions are listed in Table 1.
### Table 1. Representative few-shot learning methods and their descriptions.

<table>
<thead>
<tr>
<th>Type</th>
<th>Core Idea</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optimization-based Methods</strong></td>
<td>This type of approach aims to learn to optimize the model parameters given the gradients computed from the few-shot examples.</td>
<td>MAML [18]</td>
<td>MAML trains with a set of initialization parameters, and on the basis of the initial parameters, one or more steps of gradient adjustment can achieve the purpose of quickly adapting the model to new tasks with only a small amount of data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNAIL [20]</td>
<td>A novel combination of temporal convolution and soft attention to learn the optimal optimization strategy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATAML [22]</td>
<td>ATAML facilitates task-agnostic representation learning through task-agnostic parameterization and enhances the adaptability of the model to specific tasks through attention mechanism.</td>
</tr>
<tr>
<td><strong>Metric Learning</strong></td>
<td>In metric-based methods, instances are mapped into the feature space, the distance between the query and support sets are measured, and the classification is completed using the nearest neighbors concept.</td>
<td>Siamese Network [23]</td>
<td>Siamese network contains two parallel neural networks that are trained to extract pair-wise sample features, and the Euclidean distance between features are measured.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relation Network [21]</td>
<td>Different from the Siamese network and Matching network which adopt a single and fixed metric, Relation network compares relations with a nonlinear metric learned by a neural network.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MsKPRN [24]</td>
<td>MsKPRN extends the Relation Network to be position-aware and integrates multi-scale features.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSFN [25]</td>
<td>MSFN learns a multi-scale feature space and similarities between the multi-scale and class representation are computed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Adaptive Metric Learning Model [26]</td>
<td>Yu et al. [26] proposed an adaptive metric learning model that is able to automatically determine the best weighted combination for emerging few-shot tasks from a set of metrics obtained by meta-learning.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Knowledge-Guided Relation Network [27]</td>
<td>Sui et al. [27] proposed a knowledge-guided metric model that uses external knowledge to imitate human knowledge and generate relational networks that can apply different metrics to different tasks.</td>
</tr>
<tr>
<td><strong>Other Methods</strong></td>
<td>There is no unified core idea for these methods. They solve few-shot tasks in different ways, but all achieve competitive results.</td>
<td>BERT-PAIR [28]</td>
<td>BERT-PAIR combines query and each support sample into a sequence and utilizes BERT [29] to predict whether each pair expresses the same class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LeSAML [30]</td>
<td>It utilizes the information implied by class labels to assist pre-trained language models extracting more discriminative features.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SALNet [31]</td>
<td>This method trains a classifier from labeled data through an attention mechanism and collects lexicons containing important words for each category, and then uses new data labeled by the combination of classifiers and lexicons to guide the learning of the classifier.</td>
</tr>
</tbody>
</table>

2.2. **Graph Neural Network**

Graph neural networks (GNNs) were originally designed to process graph-structured data. GNNs can efficiently handle data structures containing complex relations and discover potential connections between data with the ability to transform and aggregate neighbors. Some of the GNN models for few-shot tasks are listed in Table 2.
Table 2. Some of the current GNN models for few-shot tasks and their descriptions.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple GNN [32]</td>
<td>Garcia et al. [32] constructed a graph model in which the query and all support samples are closely connected and used a node-focused GNN to transfer instance-level relations and label information.</td>
</tr>
<tr>
<td>TPN [33]</td>
<td>This method further considers the relations among query samples.</td>
</tr>
<tr>
<td>EGNN [34]</td>
<td>It adopts an edge-labeling framework to explicitly model the intra-class similarity and inter-class dissimilarity of samples, and dynamically update node and edge features to achieve complex information interactions.</td>
</tr>
</tbody>
</table>

However, these models are all designed for image classification tasks, and these methods only transfer instance-level relations in GNNs, which make it difficult to handle elusive NLP tasks. In contrast, the HFGNN model proposed in this study considers relations between samples from multiple perspectives, and the accurate and sufficient relations help the model construct more discriminative features.

2.3. Multi-Criteria Decision-Making

Zadeh [35] proposed fuzzy set theory to address problems related to fuzzy, subjective, and imprecise judgments. However, this theory lacks the ability to solve the problem of multi-criteria decision-making (MCDM). In this regard, Torra [12] proposed HFS, which determines the corresponding evaluation index and membership function according to the different attributes of the elements in the universe. HFS is a powerful tool for solving problems involving many uncertainties.

In recent years, some more efficient MCDM methods have been proposed. Deveci et al. [36] explored a novel approach that integrates Combined Compromise Solution (CoCoSo) with the context of type-2 neutrosophic numbers to overcome the challenging decision process in Urban freight transportation tasks. Pamucar et al. [37] developed a novel integrated decision-making model which is based on Measuring Attractiveness by a Categorical Based Evaluation Technique (MACBETH) for calculating the criteria weights and Weight Aggregated Sum Product Assessment (WASPAS) methods under the fuzzy environment with Dombi norms.

Considering the operating efficiency of the graph neural network model and the simplicity and effectiveness of the HFS theory, we introduce the HFS theory instead of other complex MCDM methods into the dual graph neural networks to fuse the relations between few-shot examples.

3. Problem Definition

The few-shot classification task trains a classifier that can accommodate new classes not seen in training, for which only a few examples are available in each class. Usually, the dataset used for this task is divided into two parts: the large training set $C_{train}$ containing a series of categories and the target test set with a disjoint set of new classes $C_{test}$. $C_{test}$ contains a support set $S$ with a few labeled samples. If the support set contains $N$ categories and each category contains $K$ samples, the target problem is called an $N$-way $K$-shot problem. In principle, the classification model can be directly trained with the support set, but the performance will be poor because $K$ is too small. Therefore, it is necessary to perform meta-learning on the training set. Meta-learning extracts transferable knowledge on the training set to help the model perform better few-shot learning on the support set, thereby improving the classification accuracy of the query samples in the test set.

In meta-learning, the episodic strategy [13] constructs training episodes to simulate FSL test scenario so that the classifier can perform well with a small number of annotations. More specifically, the training episode is formed by first randomly selecting $N$ categories...
Electronics 2022, 11, 2423

from the training set and then choosing \( K \) samples within each selected class to act as the support set \( S = \{(x_i, y_i)\}_{i=1}^{m} (m = N \times K) \), as well as a fraction of the remaining samples to serve as the query set \( Q = \{(x_i, y_i)\}_{i=1}^{n} \), where \( x_i \) represents the sample, and \( y_i = \{1, 2, \ldots, N\} \) represents the label corresponding to the sample. During the meta-learning process, the support set is used to train the model to minimize the prediction loss on the query set. Meta-training performs this procedure iteratively episode by episode until the model converges.

4. Methods

4.1. Overview

As shown in Figure 1, HFGNN-Proto consists of three parts: the text embedding component, the HFGNN component, and the prototypical network component.

Figure 1. The overall framework of HFGNN-Proto. An example of a 2-way 2-shot task is illustrated. Different colored circles represent different classes. Circles with special symbols represent distribution features, and those without symbols represent instance features. Solid circles represent support samples, and dashed circles represent query samples. (a) the process of feature extraction; (b) The HFGNN component discovers, transmits, and integrates multi-attribute relations between samples to enhance embedding features. Note that the HFGNN contains several layers. For simplicity, only one layer is shown in the figure. The detailed process of this framework is described in Section 4.3. (c) The prototypical network component takes the augmented features as input and classifies query samples.

4.2. Text Embedding

To better reflect the semantic information of the text, the pre-trained language model BERT [29] is used to extract text semantic feature representations. BERT utilizes a masked language model for pre-training and employs a deep bidirectional transformer component, which is capable of generating language representations containing contextual information, to build the entire model. Given the input text \( x = [w_1, w_2, \ldots, w_n] \), a special token [CLS] used for classification is inserted at the beginning of the word sequence, and the output of the token in the last transformer is the text embedding. The output of BERT contains information about the context of the text \( x \), and it can be represented as \( h = f_{emb}(x | \theta_{emb}) \), where \( h \in \mathbb{R}^d \), \( d \) represents the output dimension of BERT, and \( \theta_{emb} \) represents the parameters in the BERT encoder, which is fine-tuned during the training process.

4.3. HFGNN

4.3.1. Overview

This section introduces the HFGNN model in detail. As shown in Figure 1b, the HFGNN model contains three modules: a relation generator, a relation fusion module and an instance node updater. In each layer, the multi-attribute relations in task \( T \) are learned by
The instance node is initialized by the output of BERT, and the initial node of text
relations are unknown, the distribution graph node features are initialized according to the
sample and other support samples. These instance-level relations are aggregated to reflect
dimension vector, in which each element represents the instance-level relations between the
instance graph is denoted as
and final relation
between instance features and distance relations between distribution features, respectively;
features and edge features in the instance graph and distribution graph, respectively;
The process of multi-attribute relation transfer in HFGNN.

Before the iteration process starts, we need to initialize the nodes in the dual-graph. The
instance node is initialized by the output of BERT, and the initial node of text \( x_i \) in the
instance graph is denoted as \( v_{0,i} = f_{emb}(x_i) \). The node in distribution graph is a \( N \times K \)
dimension vector, in which each element represents the instance-level relations between the
sample and other support samples. These instance-level relations are aggregated to reflect
the overall distribution of the sample in the support set. Since the current instance-level
relations are unknown, the distribution graph node features are initialized according to the
following rules:

\[
v_{0,i}^{dist} = \begin{cases} \left[ \frac{\delta(y_i, y_j) \mathbf{1}}{\mathbf{1} \cdot \mathbf{1}} \right]_{i=1:|T|} & \text{if } x_i \text{ belongs to } S \\ \frac{1}{T, \ldots, \frac{1}{T}} & \text{if } x_i \text{ belongs to } Q, \end{cases}
\]  

4.3.2. Relation Generator

More specifically, in the relation generator, the initial features extracted by BERT
are used to compute instance-level relations \( E_i^{inst} \) as well as the orientation difference
\( N_i^{inst} \) of instance features in the instance graph. For the distribution graph, the instance-level
relations \( E_i^{dist} \) are aggregated to construct distribution features \( V_i^{dist} \) of samples and
distribution-level relations \( D_i^{dist} \) are computed between distribution features. The above relations are transmitted to the relation fusion module to obtain the hesitant fuzzy relation \( R_i^h \). A linear function \( F(\cdot) \) converts \( R_i^h \) to the final relation
representation \( R_i \). The instance node updater combines \( R_i \) and instance features \( v_{i-1}^{inst} \) to
construct a hesitant fuzzy graph \( G_i^h = (V_i^{inst}, R_i, T) \), and updates the instance features by
aggregating the information of neighbors. This process is continuously repeated in HFGNN
to fully explore the multi-attribute relations between samples in the task. The process of
relation transfer is shown in Figure 2.

Figure 2. The process of multi-attribute relation transfer in HFGNN.

The specific meanings of some symbols are as follows: \( V_i^{inst} = \{v_{i,j}^{inst}\}_{i=1:|T|} \), \( E_i^{inst} = \{e_{i,j}^{inst}\}_{i,j=1:|T|} \) and \( V_i^{dist} = \{v_{i,j}^{dist}\}_{i=1:|T|} \), \( E_i^{dist} = \{e_{i,j}^{dist}\}_{i,j=1:|T|} \) represent the set of node
features and edge features in the instance graph and distribution graph, respectively; \( N_i^{inst} = \{n_{i,j}^{inst}\}_{i,j=1:|T|} \) and \( D_i^{dist} = \{d_{i,j}^{dist}\}_{i,j=1:|T|} \) are the collection of direction relations
between instance features and distance relations between distribution features, respectively;
\( R_i^h = \{r_{i,j}^h\}_{i,j=1:|T|} \) and \( R_i = \{r_{i,j}\}_{i,j=1:|T|} \) represent the set of hesitant fuzzy relations \( r_{i,j}^h \)
and final relation \( r_{i,j} \), respectively; \( T = N \times K + \bar{T} \) denotes the number of samples in task
\( T \); and \( l \) indicates the layer of HFGNN.
where \( T = N \times K \) represents the number of support samples in the task, \( \delta(y_i, y_j) \) is the Kronecker delta function, which outputs one when label \( y_i = y_j \) and zero otherwise. \( S \) and \( Q \) represent the support set and query set, respectively.

**Instance-level Relations** In the first layer, the edge features representing instance-level relations in the instance graph are first computed by the instance-edge-compute function:

\[
e_{\text{inst}}^{l,ij} = f_{\text{inst}}^{e,l}(v_{\text{inst}}^{l-1,i} - v_{\text{inst}}^{l-1,j}),
\]

where \( f_{\text{inst}}^{e,l} : \mathbb{R}^m \rightarrow \mathbb{R} \) is a neural network that maps \( m \)-dimensional edge features to one-dimensional values in a fixed range, consisting of a linear-BN-LeakyReLU block, a single linear layer, and a sigmoid layer with parameters \( \theta_{\text{inst}}^{e,l} \).

**Direction Relations** Then, the cosine similarity between instance features, which can reflect the difference in the direction of the features, are calculated:

\[
n_{\text{inst}}^{l,ij} = \cos(v_{\text{inst}}^{l-1,i}, v_{\text{inst}}^{l-1,j}),
\]

where the value of the directional relations range between \([-1, 1]\] and are inversely proportional to the magnitude of the directional difference.

**Distribution-level Relations** Next, in the distribution graph, instance-level relations are aggregated and distribution-level feature representations are generated through the dist-node-update function:

\[
v_{\text{dist}}^{l,i} = f_{\text{dist}}^{v,l}(\sum_{j=1}^{T} e_{\text{dist}}^{l,ij}),
\]

where \( \| \) represents the concatenation operator and \( f_{\text{dist}}^{v,l} : \mathbb{R}^{2T} \rightarrow \mathbb{R}^T \) is composed of a linear layer and LeakyReLU with parameters \( \theta_{\text{dist}}^{v,l} \).

The dist-edge-compute function takes distribution features as input and computes edge features representing the distribution-level relations of the samples:

\[
e_{\text{dist}}^{l,ij} = f_{\text{dist}}^{e,l}(\|v_{\text{dist}}^{l,i} - v_{\text{dist}}^{l,j}\|),
\]

where \( f_{\text{dist}}^{e,l} : \mathbb{R}^T \rightarrow \mathbb{R} \) transforms the distribution features using the combination of a linear-BN-LeakyReLU block, a single linear layer, and a sigmoid layer. The parameters in \( f_{\text{dist}}^{e,l} \) can be expressed as \( \theta_{\text{dist}}^{e,l} \).

**Distance Relations** In HFGNN, the distance relations between distributed nodes are measured by the following methods:

\[
d_{\text{dist}}^{l,ij} = \text{SUM}(v_{\text{dist}}^{l,i} - v_{\text{dist}}^{l,j}),
\]

where \( \text{SUM}(\cdot) \) represents the sum of the distance values in each dimension.

4.3.3. Relations Fusion Module

First, we determine the membership function corresponding to each relation. Since the instance-level and distribution-level relations have been standardized by the sigmoid layer to a one-dimensional value, \( f_{\text{inst}}^{e,l} \) and \( f_{\text{dist}}^{e,l} \) can be considered as the membership function of the instance-level and distribution-level relations, and \( e_{\text{inst}}^{l,ij} \) and \( e_{\text{dist}}^{l,ij} \) are the corresponding membership values. The membership functions of the remaining two relations are defined as follows:

In the \( l \)-th iteration, the membership function of the directional relations between instance features is:

\[
U_{n_{\text{inst}}}^{e,l}(v_{\text{inst}}^{l,i}, v_{\text{inst}}^{l,j}) = \frac{n_{\text{inst}}^{l,ij} + 1}{2}.
\]
In the $l$-th iteration, the membership function of the distribution feature distance relations is:

$$U_{dist}(v_{ij}^{dist}, v_{lj}^{dist}) = \exp\left(-\frac{d_{dist}^{l_{ij}}}{\sum_{k=1}^{l} d_{dist}^{l_{jk}}}\right).$$  \hspace{1cm} (8)

The membership values obtained from the relation membership functions defined above are all in the range $[0, 1]$ and are proportional to the relations strength. The membership functions convert the relations between the samples into the corresponding hesitant fuzzy set, and all the membership values of the relations between a sample and itself are set to 1 to construct an ideal HFS $h_p$, which can be used as a standard to measure the similarity between different samples. For example, the HFS between $x_i$ and itself can be represented as $\{1, 1, 1, 1\}$, while the HFS between $x_i$ and $x_j$ is $\{e_{l_{ij}}^{\text{inst}}, e_{l_{ij}}^{\text{dist}}, \frac{\sigma_{l_{ij}}^{\text{inst}}}{\sigma_{l_{ij}}^{\text{dist}}}, \exp\left(-\frac{d_{dist}^{l_{ij}}}{\sum_{k=1}^{l} d_{dist}^{l_{jk}}}\right)\}$.

The similarity between the HFS and the ideal set can be measured by a distance metric, and it is inversely proportional to the distance calculation result. Here, the hesitant standard Euclidean distance is used as a measure. Assuming $h_{l_{ij}}$ is the HFS between $x_i$ and $x_j$, the hesitant standard Euclidean distance between $h_{l_{ij}}$ and $h_p$ can be expressed as:

$$d_{ghnh}(h_{l_{ij}}, h_p) = \left[\frac{1}{l} \sum_{\beta=1}^{l} |h_{l_{ij}}^{\beta} - h_p^{\beta}|^2\right]^\frac{1}{2},$$  \hspace{1cm} (9)

where $\frac{1}{l}$ represents the largest number of elements in the HFS $h_{l_{ij}}$ and $h_p$, and $h_{l_{ij}}^{\beta}$ and $h_p^{\beta}$ are the $\beta$-th largest values in $h_{l_{ij}}$ and $h_p$, respectively. The similarity between HFSs, that is, the hesitant fuzzy relation $r_{l_{ij}}^h$, can be expressed as:

$$r_{l_{ij}}^h = 1 - d_{ghnh}(h_{l_{ij}}, h_p).$$  \hspace{1cm} (10)

Hesitant fuzzy relations contain multi-attribute similarity relations information. Considering that the nodes in the graph are easily affected by noise caused by adjacent nodes with low correlation, the hesitant fuzzy relations are further adjusted by a linear function $F(\cdot)$:

$$F(r_{l_{ij}}^h) = \begin{cases} 
1 & \text{if } r_{l_{ij}}^h > \beta, \\
r_{l_{ij}}^h & \text{if } \beta \geq r_{l_{ij}}^h \geq \alpha, \\
0 & \text{if } r_{l_{ij}}^h < \alpha,
\end{cases}$$  \hspace{1cm} (11)

where $F(\cdot)$ further strengthens the relations with high similarity ($r_{l_{ij}}^h > \beta$), while weakening the relations with extremely low similarity ($r_{l_{ij}}^h < \alpha$). The transformed final relation $r_{l_{ij}}^h$ has strong inductive biases.

4.3.4. Instance-Node Updater

The instance node updater combines $r_{l_{ij}}$ and $V_{l_{i}}^{\text{inst}}$ to construct a hesitant fuzzy graph. Note that the hesitant fuzzy graph is no longer a fully connected structure since a part of relations $R_l$ are zero. The inst-node-update function in the hesitant fuzzy graph aggregates neighbors to update instance features:

$$v_{l_{id}}^{\text{inst}} = f_{l_{id}}^h \left(\frac{\sum_{j} r_{l_{ij}}^h v_{l_{jd}}^{\text{inst}}}{\sum_{j} r_{l_{ij}}} \right),$$  \hspace{1cm} (12)

where $\parallel$ represents the connection operator and $f_{l_{id}}^h$ is a linear-BN-LeakyReLU block with parameters $\theta_{l_{id}}^h$. The completion of the update marks the completion of an iteration. In HFGNN, the instance feature $v_{l_{id}}^{\text{inst}}$ output from the current layer is taken as the in-
put of the next layer to start a new round of iteration. Then, the above process is repeated until more discriminative instance features are constructed.

4.4. Prototypical Network

The Prototypical Network embeds the instance features enhanced by HFGNN into the prototypical space, computes the class prototypical representation, and classifies query samples. The class prototype representation is the mean of instance embeddings for each class in the support set:}

\[
c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\phi(v_i),
\]

where \(v_i\) represents the instance feature of the last layer in the HFGNN, \(f_\phi\) is a linear layer with learnable parameters \(\theta_\phi\), and \(S_k\) represents the support samples labeled as category \(k\). The probability that query \(x\) belongs to one category in support set \(S\) can be calculated as follows:

\[
p_\phi(y = k | x) = \frac{\exp(-d(f_\phi(x), c_k))}{\sum_{k'}^{\left|S\right|} \exp(-d(f_\phi(x), c_{k'}))},
\]

where \(d(\cdot, \cdot)\) is the distance calculation function between vectors. The cross-entropy loss function is used to train HFGNN-Proto, and the parameters \(\theta_{emb}, \theta_{inst}^{gl}, \theta_{dist}^{gl}, \theta_{h}^{gl}\) and \(\theta_\phi\) in the model are optimized by minimizing the following loss values:

\[
L = -\log p_\phi(y = k | x).
\]

5. Experiments

5.1. Datasets

We conduct experiments on two widely used few-shot text classification datasets to evaluate HFGNN-Proto.

The Amazon Review Sentiment Classification (ARSC) dataset contains English reviews for 23 domains of products on Amazon. For each product domain, Yu et al. [26] constructed three binary classification tasks with different scoring thresholds. These buckets form \(23 \times 3 = 69\) tasks in total. Following previous works, 12 tasks in 4 domains (Books, DVD, Electronics, and Kitchen) are selected as the target test set, and each category in the test set contains only five labeled support samples. We create a 2-way 5-shot classification task on this dataset.

The Few-Shot Relation Classification (FewRel) dataset [38] contains 100 relations from Wikipedia, and each relation consists of 700 instances. The number of relations in FewRel that are used for the training, validation and test sets is 64, 16 and 20, respectively. Following the settings used by Sun et al. [6], we conduct 5-way 5-shot and 10-way 5-shot experiments on the FewRel dataset. It should be noted that the label of test samples in this dataset has not been released to the public. The classification results are submitted to the FewRel evaluation website [39] provided by Gao et al. [38] for online evaluation of effectiveness.

5.2. Implementation Details

5.2.1. Meta-Training and Meta-Testing

The experiments on ARSC consist of 20,000 iterations with eight episodes randomly selected for meta-training in each iteration, while FewRel consists of 10,000 iterations with four episodes per iteration. Our model is trained by the Adam optimizer with an initial learning rate of \(1 \times 10^{-5}\) and a weight decay rate of \(5 \times 10^{-8}\). The learning rate decays by 0.01 every 100 iterations.

Following previous work, few-shot classification accuracy is used as the evaluation metric for performance. For FewRel, every 1000 iterations, the model is evaluated on 600 randomly drawn validation episodes to find the best parameters. The model with the best parameters is finally applied to the test set, which contains 10,000 test episodes.
The classification results are uploaded and evaluated online. For ARSC, tests are performed every 100 iterations. Note that the support set for testing in ARSC was determined by Yu et al. [26]. Consequently, we just need to sample the query from the target task to form the test episode. The average classification accuracy of the 12 target tasks in ARSC is used as the final accuracy.

5.2.2. Parameters Setting

The text encoder adopts Hugging Face’s implementation of BERT (base version) [40], and the parameters of the encoding layer are initialized by the pre-trained model publicly provided by Google. BERT-base converts text into 768-dimensional feature vectors. The HFGNN-Proto model is conducted with two GNN layers ($l = 2$) to prevent features from being over-smoothed. In order to maintain the balance between the degree of weakening and strengthening of the relations by the linear function $F(\cdot)$, we keep the sum of $\alpha$ and $\beta$ as always 1. We perform small-scale experiments to determine the value of $\alpha$ and $\beta$ on a small subset of ARSC following the experimental setup above. According to the results shown in Figure 3, the thresholds $\alpha$ and $\beta$ are set to 0.3 and 0.7, respectively. The linear layer $f_\phi(\cdot)$ in the prototypical network embeds features into a 128-dimensional prototypical space.

![Figure 3. Accuracy of the model on a subset of ARSC with different values of $\alpha$ and $\beta$.](image)

5.3. Results and Analysis

We compare the effect of HFGNN-Proto on the ARSC and FewRel datasets with many baseline models. The experimental results are shown in Tables 3 and 4, respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching Network [13]</td>
<td>65.73</td>
</tr>
<tr>
<td>Prototypical Network [3]</td>
<td>68.17</td>
</tr>
<tr>
<td>MAML [18]</td>
<td>78.33</td>
</tr>
<tr>
<td>Graph Network [32]</td>
<td>82.61</td>
</tr>
<tr>
<td>Relation Network [21]</td>
<td>83.07</td>
</tr>
<tr>
<td>SNAIL [20]</td>
<td>82.57</td>
</tr>
<tr>
<td>ROBUSTTC-FSL [26]</td>
<td>83.12</td>
</tr>
<tr>
<td>Induction Network [9]</td>
<td>85.63</td>
</tr>
<tr>
<td>HFGNN-Proto (OURS)</td>
<td>88.36</td>
</tr>
</tbody>
</table>
Table 4. Comparison of classification accuracy (%) on FewRel.

<table>
<thead>
<tr>
<th>Model</th>
<th>5-Way 5-Shot</th>
<th>10-Way 5-Shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune [36]</td>
<td>68.66</td>
<td>55.04</td>
</tr>
<tr>
<td>kNN [36]</td>
<td>68.77</td>
<td>55.87</td>
</tr>
<tr>
<td>Meta Network [41]</td>
<td>80.57</td>
<td>69.23</td>
</tr>
<tr>
<td>Graph Network [32]</td>
<td>81.28</td>
<td>64.02</td>
</tr>
<tr>
<td>SNAIL [20]</td>
<td>79.40</td>
<td>68.33</td>
</tr>
<tr>
<td>Prototypical Network [5]</td>
<td>89.05</td>
<td>81.46</td>
</tr>
<tr>
<td>HATT-Proto [7]</td>
<td>90.12</td>
<td>83.05</td>
</tr>
<tr>
<td>HAPN [8]</td>
<td>91.02</td>
<td>84.16</td>
</tr>
<tr>
<td>EGNN-Proto [42]</td>
<td>92.29</td>
<td>86.09</td>
</tr>
<tr>
<td>BERT-PAIR [28]</td>
<td>93.22</td>
<td>87.02</td>
</tr>
<tr>
<td>HFGNN-Proto (OURS)</td>
<td>94.45</td>
<td>89.40</td>
</tr>
</tbody>
</table>

Results on ARSC

The results in Table 3 show that HFGNN-Proto achieves a classification accuracy of 88.36% and outperforms most previous baseline models. Some existing FSL methods, such as Matching Network [13], MAML [18], and Relation Network [21], perform poorly on ARSC despite the fact that they possess outstanding performance in the vision domain. Induction Network [9] uses a routing mechanism to induce better class representations, which improves the classification accuracy to a new level, but the performance of HFGNN-Proto still far exceeds it. Compared with these methods, the reason why HFGNN-Proto improves the classification accuracy is that it fully learns the potential relations of samples in the meta-task at the multi-attribute level.

Results on FewRel

The results in Table 4 show that the 5-way 5-shot and 10-way 5-shot classification accuracy of HFGNN-Proto on the FewRel dataset are 94.45% and 89.40%, respectively, which both exceed the state-of-the-art method BERT-PAIR [28]. The classification performance in the 5-shot scenario is 1.23% higher than that of BERT-PAIR, and the accuracy improvement in the 10-shot scenario setting is even more significant, with an accuracy improvement of 2.28% over that of BERT-PAIR. EGNN-Proto [42] also uses the combination of GNNs and Prototypical Network, but the effect is far less than that of our model. EGNN-Proto uses the fully connected graph structure to transmit instance-level relations and edge label information. This process faces the problem of single relational information and the influence of noise caused by a fully connected structure. The relations in HFGNN-Proto are more sufficient, and the function $F(\cdot \cdot)$ provides a more reasonable graph structure for the update of instance features and effectively avoids the influence of irrelevant noise.

5.4. Comparison with PLMs

We also compare the performance of HFGNN-Proto with the pretrained language model GPT and its variants on the real-world few-shot classification datasets constructed by Alex et al. [43]. The results are shown in Table 5. We can see that the performance of GPT-2 [44] and GPT-Neo [43] is far inferior to our model, while GPT-3 [45] with up to 175 billion parameters achieves the highest performance. It is worth noting that our proposed HFGNN-Proto method achieves performance close to GPT-3 with several orders of magnitude fewer parameters than GPT-3.

Table 5. Comparison of HFGNN-Proto and PLMs. ADE and NIS are datasets constructed by Alex et al. [43].

<table>
<thead>
<tr>
<th>Model</th>
<th>ADE</th>
<th>NIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-2 [44]</td>
<td>60.0</td>
<td>56.1</td>
</tr>
<tr>
<td>GPT-Neo [43]</td>
<td>45.2</td>
<td>40.8</td>
</tr>
<tr>
<td>GPT-3 [45]</td>
<td>68.6</td>
<td>67.9</td>
</tr>
<tr>
<td>HFGNN-Proto</td>
<td>68.4</td>
<td>67.5</td>
</tr>
</tbody>
</table>
5.5. Ablation Experiment

To analyze the influence of different modules in HFGNN-Proto on the performance, we conduct ablation experiments on ARSC.

The abundant information in multi-attribute relations builds more reasonable feature representations, and measuring multiple relations is crucial to the performance of the model. To support this idea, we compare the classification performance of HFGNN-Proto with different relation combinations. The experimental results in Table 6 show that the accuracy is greatly improved after HFGNN-Proto learns distribution-level relations on the basis of the initial instance-level relations, and the best performance is achieved when all relations mentioned in Section 4.3.2 are considered, corresponding to the best result reported in Table 3.

Table 6. Classification accuracy (%) of HFGNN-Proto with different relation combinations on ARSC. IR stands for instance-level relations and DR stands for Distribution-level relations.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR</td>
<td>87.15</td>
</tr>
<tr>
<td>IR + DR</td>
<td>87.99</td>
</tr>
<tr>
<td>All relations</td>
<td>88.36</td>
</tr>
</tbody>
</table>

We further report the effect of HFS strategy, the linear function $F(\cdot)$, and the layers of GNN on model performance as shown in Table 7. We can see that the best performance is achieved when the number of GNN layers is 2, corresponding to the result reported in Table 3. More layers of GNN did not further improve the performance. The table also demonstrates the effectiveness of HFS and the linear function $F(\cdot)$. HFGNN-Proto performs poorly when we replace the HFS with the averaging strategy to process the relations, which proves the contribution of HFS to the performance. On the other hand, the performance of the model without $F(\cdot)$ is significantly lower, indicating that our designed function generates a more reasonable graph structure and provides inductive biases for the transfer of relations.

Table 7. Further ablation study of HFGNN-Proto on ARSC. LF stands for linear function. Average means that the average strategy is used instead of HFS to handle the relations.

<table>
<thead>
<tr>
<th>GNN Layers</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HFS + LF</td>
<td>88.36</td>
</tr>
<tr>
<td>HFS + LF</td>
<td>87.79</td>
</tr>
<tr>
<td>HFS + LF</td>
<td>88.13</td>
</tr>
<tr>
<td>HFS + LF</td>
<td>87.96</td>
</tr>
<tr>
<td>HFS</td>
<td>87.46</td>
</tr>
<tr>
<td>Average</td>
<td>86.22</td>
</tr>
<tr>
<td>Average + LF</td>
<td>87.29</td>
</tr>
</tbody>
</table>

5.6. Visualization

To further analyze the benefit of HFGNN on instance features, we randomly select an episode from the FewRel 10-way 5-shot test set and visualize the support set features before and after HFGNN transformation through t-SNE, as shown in Figure 4. The initial features extracted by BERT diverge in space and are interleaved with each other. After the transformation of the first layer in the HFGNN, the features belonging to the same class are aggregated, and the features of different classes are far away from each other. After the second transformation, the distribution of features in the space is further improved. These results fully demonstrate the effectiveness of the HFGNN in discovering relations between samples and enhancing instance features.
5.7. Limitations

General GNNs can efficiently perform edge classification. The edges between nodes represent the similarity between samples, and the classification results can be directly generated by the edges. However, our proposed HFGNN method does not support edge classification. This is because we change the edges in hesitant fuzzy graph according to Equation (11). These edges with weak relations are directly cut off, which results in the inability of HFGNN to perform efficient edge classification.

Another limitation of HFGNN-Proto is that the model can only handle English tasks currently, and it is still a challenge for our model to handle few-shot tasks in other languages.

6. Conclusions

In this paper, we propose a HFGNN-Proto model that can fully explore the multi-attribute relations between samples for few-shot text classification. Abundant relation information helps the model to better handle complex NLP tasks. Relations are transmitted in a dual-graph and integrated by HFS theory. The use of HFS effectively avoids the loss of information and improves the accuracy of the overall relation representation. Moreover, the linear function further improves the rationality and accuracy of the relations, which helps the model construct a more accurate hesitant fuzzy graph for message transmission and provides strong inductive biases for feature enhancement. Finally, a prototypical network can perform more quick and efficient classification based on the enhanced features. HFGNN-Proto achieves better generalization on unseen tasks given its ability to discover precise potential connections between texts. Experimental results demonstrate that HFGNN-Proto outperforms existing state-of-the-art few-shot models. In the future, we will focus on exploring more substantial relations among few-shot samples, trying to use other MCDM methods to handle relations, and generalizing HFGNN-Proto to few-shot tasks in other domains.

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

Funding: This work is supported by the Natural Science Foundation of Hebei Province, China (No. F2019201329) and the Key Project of the Science and Technology Research Program in the University of Hebei Province, China (No. ZD2019131).

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


