future internet

MDPI

Article

Transfer Learning for Multi-Premise Entailment with Relationship Processing Module

Pin Wu, Rukang Zhu * and Zhidan Lei

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; wupin@shu.edu.cn (P.W.); leizd@outlook.com (Z.L.)
* Correspondence: rukang.zhu@outlook.com

Abstract: Using the single premise entailment (SPE) model to accomplish the multi-premise entailment (MPE) task can alleviate the problem that the neural network cannot be effectively trained due to the lack of labeled multi-premise training data. Moreover, the abundant judgment methods for the relationship between sentence pairs can also be applied in this task. However, the single-premise pre-trained model does not have a structure for processing multi-premise relationships, and this structure is a crucial technique for solving MPE problems. This paper proposes adding a multi-premise relationship processing module based on not changing the structure of the pre-trained model to compensate for this deficiency. Moreover, we proposed a three-step training method combining this module, which ensures that the module focuses on dealing with the multi-premise relationship during matching, thus applying the single-premise model to multi-premise tasks. Besides, this paper also proposes a specific structure of the relationship processing module, i.e., we call it the attention-backtracking mechanism. Experiments show that this structure can fully consider the context of multi-premise, and the structure combined with the three-step training can achieve better accuracy on the MPE test set than other transfer methods.

Keywords: transfer learning; multi-premise entailment; natural language inference; attention mechanism

Citation: Wu, P.; Zhu, R.; Lei, Z. Transfer Learning for Multi-Premise Entailment with Relationship Processing Module. *Future Internet* **2021**, *13*, 71. https://doi.org/10.3390/fi13030071

Academic Editor: Rafael Valencia-Garcia

Received: 3 January 2021
Accepted: 1 March 2021
Published: 13 March 2021

1. Introduction

The multi-premise entailment task or multi-premise natural language inference task [1] is an extension of the standard natural language inference task [2]. In this paper, we call the standard natural language inference task the single-premise entailment task. The definition of a single-premise entailment task is as follows [3]: typically, a human reading P would be justified in inferring the proposition expressed by H from the proposition expressed by P. The P is a premise sentence. The H is a hypothesis sentence. Similarly, the multi-premise entailment task should be defined as follows: typically, a human reading multiple P would be justified in inferring the proposition expressed by H from the proposition expressed by multiple P. The single-premise entailment task and the multi-premise entailment task are shown in Figures 1 and 2, respectively.

The multi-premise entailment task describes multiple descriptions of the same scene or event by multiple people. For example, multiple witnesses' memory or description of a crime scene may have different language forms and may have a different emphasis on observation due to different perspectives. The MPE task aims to determine whether another description (the suspect's statement, in this task, we generally call it the hypothesis) is true, false, or unknowable in the context of the semantic scenario formed by these premises together.

Compared with the SPE data, we conclude that the MPE data has the following characteristics:

1. Each premise is a complete sentence and has clear semantics. This means that one premise of the SPE data cannot be cut up into multiple parts to construct multiple premises because, after segmentation, neither the syntax nor the semantics is complete;

2. There is no sequential relationship among multiple premises;
3. The information contained in each premise is unbalanced. In other words, for the semantic scenario constituted by all premises, each premise does not necessarily provide unique information for this scenario, and the description of one premise may also be a subset of another premise. This means that each premise may have a different level of contribution to the final classification.
4. Compared with the SPE data, the MPE data is more scarce, so that the corresponding network model cannot be effectively trained.

Premises:
A soccer game with multiple males playing.
Hypothesis:
Some men are playing a sport. ⇒ ***ENTAILMENT***

Premises:
An older and younger man smiling.
Hypothesis:
Two men are smiling and laughing at the cats playing on the floor.
⇒ ***NEUTRAL***

Premises:
A man inspects the uniform of a figure in some East Asian country.
Hypothesis:
The man is sleeping ⇒ ***CONTRADICTION***

Figure 1. Single-premise entailment task.

Premises:
1. *Two girls sitting down and looking at a book.*
2. *A couple laughs together as they read a book on a train.*
3. *Two travelers on a train or bus reading a book together.*
4. *A woman wearing glasses and a brown beanie next to a girl with long brown hair holding a book.*

Hypothesis:
Women smiling. ⇒ ***ENTAILMENT***

Premises:
1. *Three men are working construction on top of a building.*
2. *Three male construction workers on a roof working in the sun.*
3. *One man is shirtless while the other two men work on construction.*
4. *Two construction workers working on infrastructure, while one worker takes a break.*

Hypothesis:
A man smoking a cigarette. ⇒ ***NEUTRAL***

Premises:
1. *A group of individuals performed in front of a seated crowd.*
2. *Woman standing in front of group with black folders in hand.*
3. *A group of women with black binders stand in front of a group of people.*
4. *A group of people are standing at the front of the room, preparing to sing.*

Hypothesis:
A group having a meeting. ⇒ ***CONTRADICTION***

Figure 2. Multi-premise entailment task.

This paper proposes a method suitable for transferring from SPE pre-trained model to the MPE model. This method has the following characteristics:

- Considering the existing SPE and MPE corpus, we think the biggest difference between the latter and the former is that the latter needs to deal with the relationship between multiple premises. We propose using SNLI, a large-scale SPE data set, to learn the weight parameters of the pre-trained model, which can realize the feature extraction and classification between single premise and hypothesis. In addition, we then use a small-scale MPE corpus to learn a module that can infer the relationship

from multiple premises. We call this module the relationship processing module (R module), which combines with the pre-trained model to construct a complete MPE classification model;

- To make the R module focus on the purpose of capturing the relationship from multiple premises, a three-step training method is used in our model. First, the model parameters are learned based on the source corpus. Second, the pre-trained parameters are frozen, the R module parameters are learned based on the target corpus. Finally, the pre-trained parameters are unfrozen to fine-tune the whole model based on the target corpus. We believe that this approach limits the R module to deal with multi-premise relationships, thus keeping the pre-trained model intact.
- We propose a specific R module structure, the Attention-backtracking mechanism, which integrates all premises' sentence representation to obtain the preliminary contextual representation. The preliminary contextual representation and the context-related word vector of each premise sentence are calculated to obtain the final premise representation of multiple premises. Experiments show that this mechanism can fully consider the relationship of multiple premises.

Through the experiments of different transferring methods, it is proved that the proposed method can inherit the prior knowledge of the source corpus most effectively and improve the accuracy of MPE problem judgment the most.

This article proceeds as follows: in the Introduction section, we introduce the definitions of SPE and MPE tasks and their respective problems. This section also describes the contributions made by this article. In the related work section, some technical frameworks in related fields are introduced. In the methods section, a three-step training algorithm and attention-backtracking mechanism are proposed. In the experimental part, we compare the results of each model. The last part is the summary, which summarizes the advantages and disadvantages of the model.

2. Related Works

For natural language inference alone, the deep learning method using neural networks to construct classification models has become the most mainstream and accurate method. Other methods exist, such as the Siamese-based framework [4,5] focusing on the formation of accurate sentence vectors and the "matching-aggregation" framework [6–8] focusing on the comparison of word-level alignment [9]. These models are no exception using the large-scale labeled corpus the Stanford Natural Language Inference (SNLI) [10] as their training data set. Compared with traditional small data sets, SNLI possessing huge amount of labeled data that can make the weight parameters of neural network are fully training. Therefore, it is not so much the achievements of these different frameworks and models as the achievements of the large-scale annotated corpus.

The SPE task exploration is still going on, and a more complex semantic multi NLI dataset [11] has been recently launched. However, these corpora are all composed of a single premise sentence. The annotation data of MPE tasks is relatively scarce, which fundamentally limits the exploration of MPE problems in deep learning. Therefore, we believe that using a large-scale SPE corpus as the source knowledge domain to train an SPE model and then moving it to the MPE target knowledge domain for fine-tuning is a method worth studying.

Transferring deep, large convolutional neural networks to specific image recognition tasks is very common in the field of machine vision [12]. The low layer convolutional kernels in the network are considered to be able to learn graphic features [13], while high layer convolutions can learn more abstract image features [14]. In natural language processing, the most widely used pre-trained model comes from the language model [15,16], and its parameters are called word embedding vectors, which has now become the standard configuration of various natural language processing applications on the input side. In addition, in recent years the trend of ultra-large-scale pre-trained models initiated by

GPT [17], Bert [18] and so on [19,20] demonstrated the strong potential of transfer learning in natural language processing field.

More generally, the task we want to accomplish is called target task/domain T, its data set is called data T, and the task could be solved by the pre-trained model is called source task/domain S, its data set is called data S. In practice, a neural network is constructed and trained for task S with data S, we call this process pre-training, and then a model for task T is initialized with part or all of the obtained parameters of the pre-trained model. Finally, the target model is then iteratively trained based on the target data T. Intuitively, this approach is considered to enable the target model to inherit some prior knowledge of the source domain [21]. When dealing with the loss function, this is equivalent to changing the starting point for finding the minimum value. If the loss function is convex, no matter where the starting point is, there is only one minimum point. However, the loss function of a neural network is hugely complicated, so the starting point limited to some range may be easier to find excellent minimum points.

There may be many differences between the source task and the target task in the specific transferring process, such as different task targets and different data forms. Therefore, during the transferring process, it is often necessary to modify the pre-trained model. For example, the source task IMDB with two-classification vs. target task QC with six-classification, the sigmoid function of the pre-trained model should be replaced by the softmax function. At this time, a new random initialization is required for the full connection layer connected to softmax. This phenomenon of only transferring some parameters is called partial transferring. In the research content of this paper, the target task MPE with multiple premises is different from the source task SPE with a single premise in the form of data.

We use the simplest Siamese framework [22] as an example. We need an encoding parameter sharing layer (such as LSTM [23] or CNN [24]) to encode the premise and hypothesis separately, and then use an inference layer (such as a full connection and softmax function) for three classifications. In the Siamese framework, it is generally believed that the encoding layer can extract syntactic and semantic information [25] and convert them into feature vectors, while the inference layer can project feature vectors nonlinearly into distinguishable spaces. Because this space may be unique in each data set, the classification layer is often considered only suitable for a specific corpus task, which makes it difficult for the classification layer to use after transferring directly [21]. In this case, it is an effective method to use target data to fine-tune the pre-trained model. In transferring from SPE to MPE, the focus is on how to deal with multiple premises in the fine-tuning process.

Given the existence of an SPE model based on the Siamese framework as Figure 3, we summarize the following transferring scenarios:

- Data form transformation: Concatenating multiple premises together to form a single premise [1,26] ;
- Statistics of results after matching: Each premise and hypothesis is matched separately, and statistics of results after matching are made to obtain the final result [1] ;

Neither of these two schemes made any changes to the structure of the pre-trained model, we only need to fine-tune the pre-trained model with the target training data, but both of them have their disadvantages: First of all, the data form transformation is an attempt to convert the MPE task directly from the data level to the SPE task, which is the simplest, requiring simple preprocessing of multi-premise data. However, with the number of premises, the concatenated single premise will form a long text. In the face of the tens of thousands of premises that may exist, the encoding layer may have difficulty understanding such complex semantics. Encoders such as LSTM or Transformer [27] will consume much running time and memory when processing such a long text, which is why we do not recommend adopting large-scale pre-trained models such as Bert to do this task. The statistics of results after matching, similar to the premise-wise sum of experts (SE) model [1], take advantage of the MPE data characteristics, and treat the MPE

model as an ensemble of the SPE model. However, due to the third characteristic of MPE data(in Section 1), we do not consider this a suitable method. Because each premise in the multi-premise only has incomplete information, each match is based on the judgment of the incomplete information, which will cause all matches to be misled, and the conclusions drawn from multiple misleading combinations will deviate from the correct answer.

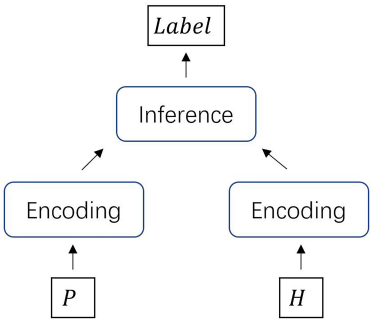

Figure 3. The Siamese framework.

3. Methods

Since the method proposed in this article is theoretically applicable to transfering all SPE models based on the Siamese framework or the "matching-aggregation" framework to the MPE task, we will illustrate our ideas in this section based on a simple conceptual model of the Siamese framework (Figure 3) for convenience. In the following subsections, we will respectively explain the existence significance of the R module, the purpose of three-step training, and the composition of a specific R module we proposed: Attention-backtracking mechanism.

3.1. Relationship Processing Module

The R module position in the transferring process is shown in Figure 4, which is between the encoding layer and the inference layer of the premise processing part. In the fine-tuning phase, the parameters of the encoder are shared by multiple premises and computed in parallel when matching. The input of the encoding layer is the sentence representation composed of the word embedding vector matrix, and the output is the sentence representation of a single vector; that is, the encoding layer will output the sentence vector representation of each premise. The classification layer trained from the SPE corpus is only suitable for inputting sentence representation of one premise, so the so-called R module is a module that can compress multiple premise sentence representations into one premise sentence representation.

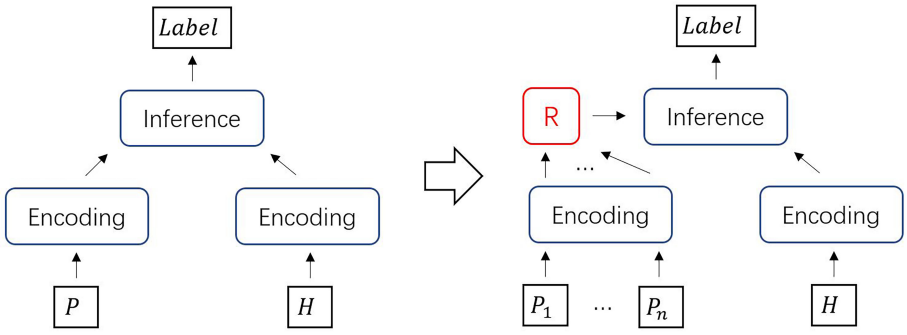

Figure 4. From SPE model to MPE model.

From the point of view of compression as a single vector representation, the most commonly used choices are summation, average pooling and maximum pooling [28]. Summing can get the whole semantics, average pooling can get the average features among multiple premises, and maximum pooling can get the most significant features. These

operations do not require parameters, and there is no nonlinear calculation. The single premise sentence vector formed is kept in the same feature space as the hypothesis sentence vector, which avoids the need for the classification layer to learn how to distinguish vectors between different feature spaces.

However, we must realize that the premise representation of the encoder for each premise will lack some semantics. The other premise information is not known when encoding one premise, which results in the overall premise vector lacking some information. Therefore, our ideal R module should be able to make up for this shortcoming. We will discuss this problem in Section 3.3. We first assume that there is such an ideal R module and it has some randomly initialized parameters, so we propose a three-step training method to learn and fine-tune the R module parameters.

3.2. Three-Step Training

In theory, we hope that the R module is relatively independent. The other parts of the model, namely the encoding layer and inference layer, does the same work in the MPE task as in the SPE task. This means that we can easily transfer the model used for SPE tasks to MPE tasks, and its frame and parameters can be relatively complete and collaborative to transfer.

Therefore, we have to ensure that the R module has the independence of completing only specific tasks in the matching process. We try to achieve this goal in training and propose the three-step training in Algorithm 1:

Algorithm 1: Three-step training.

Input: Single-premise entailment data set S, multi-premise entailment data set T
Output: MPE task model M_2

1 **SPE model parameter pre-training:** use the Siamese framework and SPE data S to train the SPE model to obtain the model M_0, and its corresponding encoding layer parameters θ_E and inference layer parameters θ_I;
2 **Training of multi-premises fusion model R:** (a) As shown in Figure 4, a suitable fusion model R is selected to fuse multiple premise vectors into one premise vector to obtain the MPE task model M_1. (b) Initialize the model R, and set its parameter to θ_R. (c) Freeze the parameters θ_E and θ_I, use MPE data T to train the model M_1, update the parameters θ_R. Update the preliminary MPE model M_1.
3 **Fine-tune the final model:** (a) Unfreeze the parameters θ_E and θ_I of the model M_1. Then, set the joint parameters $\theta = (\theta_E, \theta_I, \theta_R)$, named M_2 model. (b) Use the MPE data T again to train the M_2 model, update the parameters θ, and get the final MPE task model M_2.

In the three-step training, the first step is to use SPE data for pure SPE model training. In the Siamese framework, the encoding layer will learn how to encode context-free word vectors for sentence representations, and the inference layer will learn how to process both premise representation and hypothesis representation for classification results. In the second step, the weight parameters of the SPE model are frozen, the R module Learning the fusion method of multiple premise vectors, i.e., the relationship from multiple premises and their good fusion, which ensures the relative independence of the R module in the matching process. In the third step, all weight parameters are defrosted. Pre-trained parameters and R module parameters are trained together at a small learning rate to avoid the previously learned parameters being disrupted by large steps. Besides, the purpose of this step is also to learn a task-specific classification layer.

The R module design should fully consider the following ideas: (a) It should be independent of the encoding layer and the inference layer. (b) It should have the ability to fully integrate information between multiple premises. (c) The fused vector should be able to be accepted by the inference layer. If average pooling or maximum pooling is used with no parameters, they are treated as a particular case without using the second step of

the three-step training. We propose using the attention-backtracking mechanism to build R modules.

3.3. Attention-Backtracking Mechanism

The overall mechanism is shown in Figure 5. First of all, the Siamese-based framework is committed to building a sentence embedding vector after obtaining each sentence representation of premise sentence with an encoder. We hope to use a sentence embedding vector to build a preliminary contextual representation, which is a general description of the semantics of multiple premises:

$$c = Add(P) \tag{1}$$

$P \in \mathbb{R}^{d \times n}$, its column vectors is the sentence representation of n d-dimensional premises, $Add(.)$ is the sum of all column vectors element-by-element, $c \in$ means obtained the preliminary contextual representation that can represent all premise sentences.

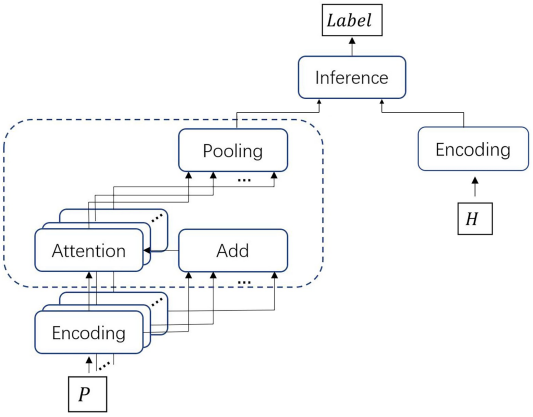

Figure 5. Attention-backtracking mechanism.

Next, vector c and all context-related word vectors in each premise are calculated for attention. Let $Y \in \mathbb{R}^{d \times l}$ be the context-related word vector matrix of a premise sentence after the encoding layer, and l is the number of words:

$$M = tanh(W^y Y + W^c c \otimes e_l) \tag{2}$$

$$\alpha = softmax(w^T M) \tag{3}$$

$$r = Y\alpha^T \tag{4}$$

$W^y, W^c \in \mathbb{R}^{d \times d}, w \in \mathbb{R}^d$ are weight parameters to be learned, $e_l \in \mathbb{R}^l$ is a vector and its elements are all 1, note that the outer product $W^c c \otimes e_l$ is repeating the linearly transformed c as many times as there are words in this premise (i.e., l times). The column vector in $M \in \mathbb{R}^{d \times l}$ is the attention representation of each word in the premise sentence, which is essentially a nonlinear combination of each word with the preliminary contextual representation. $\alpha \in \mathbb{R}^l$ is an attention weight vector, which means the importance of each word in one premise. Therefore, the $r \in \mathbb{R}^d$ obtained represents a new sentence representation combining attention information. The above operation is that when a single premise sentence is first encoded into a sentence vector, the resulting sentence vector contains biased information because there is no context information of other premise sentences. The idea of the attention-backtracking mechanism is that, after one premise is informed of other premises, the model hopes to confirm whether there is missing information or whether there is more critical information in the original sentence. Therefore, all the details will be encoded again in this section for more exciting information.

Therefore, we will get a matrix $R \in \mathbb{R}^{d \times n}$ by doing the above operations in each premise sentence, in which the column vector is composed of r calculated by each premise sentence.

Finally, we need to compress all the obtained premise vectors into a single premise vector equivalent to SPE data to input the inference layer, and we can still use pooling operation to achieve this:

$$\overline{p} = AveragePooling(R) \tag{5}$$

$\overline{p} \in \mathbb{R}^d$ is the representation of multiple premises input to inference layer corresponding to SPE model.

Since there is no sentence representation in the general sense in the "matching-aggregation" framework, as shown in Figure 6, we call the vector after the comparison module as aligned evaluation representation and the vector after the aggregate module as comprehensive evaluation representation. In our method, multiple premise sentences will have multiple comprehensive evaluation representations. The comprehensive evaluation representation can be equivalent to the sentence representation above. The aligned evaluation representation can be equivalent to the context-related word embedding representation in the previous text. In other words, after obtaining the information of other premises, the comprehensive evaluation representation of one premise sentence will go back to its own aligned evaluation representation, and the operation process is basically consistent with that described above. In the Attend process in this framework, the single hypothesis sentence will match each premise sentence, thus generating the same number of hypothesis alignment evaluation representations as to the premise sentence, i.e., after the Attend module, the hypothesis sentence should also be processed using the R module.

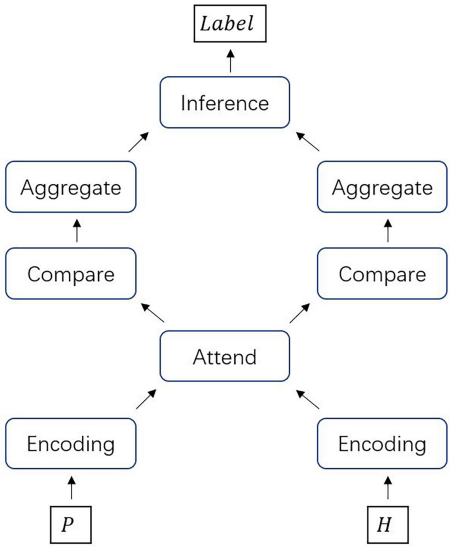

Figure 6. The "matching-aggregation" framework.

4. Experiments

4.1. Implementation Details

We use a pre-training data set from the Stanford Natural Language Inference (SNLI) [29]. The premise comes from the image annotation of Flickr30r. The corresponding hypothesis is artificially generated. There are 549,367 pairs of sentences in the training set, 10,000 pairs in the validation set and the test set, and each pair of sentences is marked as one of Entailment, Neutral, Contradiction and -. "-" means that the reviewers cannot make a consistent judgment on the relationship between sentences. We filter such sentences in data preprocessing. The Multiple Premise Entailment NLI Corpus (MPE) [1] has premises and hypotheses both from Flickr30k image labeling. Its training set has 8000 sets of sentences, the validation set and the test set have 1000 sets of sentences, respectively, each set of sentences is labeled as one of Entailment, Neutral, or Contradiction. The unique characteristic of this corpus is that each set includes four premise sentences and one hypothesis sentence.

To verify the effectiveness of the three-step training method and Attention-backtracking mechanism transferring on different strategies or different models, we will perform ex-

periments on two different SPE models, respectively. We take Inner-attention [4] as a typical Siamese framework, which averages the context-related word vectors encoded by Bi-LSTM and pools them, uses the pooling results and context-related word vectors to perform nonlinear calculations. that is, weighted summation to obtain a sentence vector representation, concatenate the premise and hypothetical sentence representations, and then send them to the fully connected layer for classification. Soft-alignment [8] is typical of the "matching-aggregation" framework. firstly, it performs word-to-word alignment calculations on premise and hypothesis to obtain the alignment feature vector (alignment evaluation representation) of each word. Secondly, we sum up the alignment evaluation representation of the premise (hypothesis) to get the comprehensive evaluation representation of the premise (hypothesis). Finally, we concatenate the comprehensive evaluation representation of premise and hypothesis, and send it to the fully connected layer.

The pre-training and fine-tuning processes of both frameworks use the same set of hyperparameters. All words are vectorized using GloVe's 300-dimensional pre-trained word vector [30], and they do not participate in training during all training processes. In the pre-training process, the dimension of the hidden layer neuron is 100 (that is, the 300-dimensional pre-trained word vector will be mapped to the 100-dimensional vector before inputting the encoding layer, and then the default dimension of the single vector in the network is 100), L2 regularization with a coefficient of 0.001 is added to all weights in the optimization process, and a dropout ratio of 0.2 is applied to all full connection layers [31]. The optimizer is Adam [32], and the learning rate is set to 0.001. Each model is only trained for five epochs on SNLI, and we believe that this can prevent the model from being overfitted to the source corpus to some extent. In the process of fine-tuning (or in the second and third steps of three-stage training), the learning rate is only changed to 0.0003. We think that a smaller learning rate can avoid the target corpus quickly disturbing the pre-trained parameters [33].

We experimented with different transferring methods for comparison, including:

- Concat: Concatenating multiple premises together to form a single premise in data form;
- Vote: Each premise and hypothesis is matched separately, and statistics of results after matching are made to obtain the final result;
- AveR: Average pooling R module, which means only average pooling for all premise representations;
- ABR: Attention backtracking R module, as described in Section 3;
- MLPR: A fully connected R module with the same parameters as ABR means that only two nonlinear full connections are made to the premise representation.

Each method will show the results of random parameter initialization and pre-trained parameter initialization (ft) for comparison. ABR and MLPR will also show the results of skipping the second or third steps of training, respectively, which are marked as (1, 3) and (1, 2), and the complete three-step training method is marked as (1, 2, 3).

4.2. Results and Discussion

It can be seen from Table 1 that in the case of completely random initialization of parameters, ABR reaches the highest 59.2%, and AveR follows, but MLPR with R module mechanism is only 56.9%, which is the lowest under the same conditions. This shows that adding a reasonable R module to the Siamese framework can be a good solution to the MPE problem even no pre-trained. We guess that the reason for the unsatisfactory results of MLPR is that after the nonlinear transformation of the premise representation, the compressed premise representation and the hypothesis representation are no longer in the same feature space. This makes the inference layer initialized by pre-trained parameters have to re-learn new parameter weights to accommodate these changes, and this results in a waste of pre-trained parameters.

Table 1. Train/test accuracies on the MPE data set and number of parameters for each approach based on Inner-attention.

Method	Parameters	Train	Test
Concat	272K	0.6310	0.5840
Concat (ft)	272K	0.7350	0.6290
Vote	272K	0.6411	0.5810
Vote (ft)	272K	0.7057	0.6370
AveR	272K	0.6258	0.5900
AveR (ft)	272K	0.7205	0.6340
ABR	292K	0.6251	0.5920
ABR (1,2)	20K	0.5140	0.5480
ABR (1,3)	292K	0.7166	0.6390
ABR (1,2,3)	292K	0.7058	**0.6440**
MLPR	292K	0.6089	0.5690
MLPR (1,2)	20K	0.5470	0.5560
MLPR (1,3)	292K	0.7079	0.6170
MLPR (1,2,3)	292K	0.7052	0.6230

Initialization based on pre-trained parameters has obviously improved the performance of all models running on the target data set, and the accuracy has increased by an average of 9% on the training set and an average of 5% on the test set. In the comparison of different transferring strategies, it can be seen that the R module mechanism still performs differently, with ABR (1,2,3) reaching the highest 64.4% and MLPR (1,2,3) being the lowest. Among other transferring methods, Vote (ft) performs best, and we will do some analysis in the following (visual display).

It can be seen from the comparison of the same model and different training methods that ABR (1,2) with frozen pre-trained parameters and only learning R module parameters performs worse than ABR, which reflects the sensitivity of the neural network to different data sets, i.e., fine-tuning is crucial. Comparing ABR (1,3) and ABR (1,2,3), as well as MLPR (1,3) and MLPR (1,2,3), we can see that our three-step training method can improve accuracy; it is meaningful to study the parameters of the R module separately.

In Figure 7, we respectively show the attention weight assigned to each word in the four premise sentences of the example sentences. It must be emphasized that the pre-trained model we used has an attention operation in itself (Inner-attention), plus the attention in the attention-backtracking mechanism, so we can make a visual comparison diagram. In each sentence, the upper is the weight assigned to the word by the pre-trained model Inner-attention, and the lower is the weight assigned to the word by the Attention-backtracking mechanism. In general, the vocabulary of attention in backtracking is basically the same as that of Inner-attention, and the less important words are more ignored.

We further observe that the model after the backtrack pays more attention to the 'dog' in the premise sentence, and we speculate that this is due to the preliminary contextual representation formed by summing the multiple premise representations formed by Inner-attention, the words that each premise cares about will be superimposed, i.e., the semantics extracted by all premises will be repeated. The most repeated words will become the protagonists in the multi-premise context. The model confirms the subject in the multi-premise in this way. In the subsequent attention backtracking process, each premise will reconfirm the predicate and object related to the subject 'dog'. In this example, the keyword 'leaps to' corresponding to 'in the air' gets greater weight after backtracking because it is a predicate related to the subject.

Besides, we also notice that 'white' in the third premise sentence reduces its weight after backtracking, even though it is a crucial descriptive word for the subject in the hypothesis sentence. However, it must be noted that in the Inner-attention model, the premise and hypothesis are not aware of each other when encoding, so the encoder can

only "intuitively" remember the most important parts of the sentence. Namely, the subject-verb-object, which is similar to the human reading mechanism.

However, because 'white' is not remembered, we think there is another reason, i.e., the problem of target corpus MPE we used. The MPE data set hypothesis is subject-verb-object structures, which results in that in most cases, the model only needs to judge whether the subject-verb-object extracted from multiple premises is consistent with the hypothesis. We believe that this is also why the Vote (including Vote (ft)) model can achieve high accuracy. Despite the unbalance of multiple premises, the subjects of multiple premises of any set of sentences in the MPE data set are the same. The example in Figure 8 is a set of sentences different from the MPE data set, which has an obvious imbalance:

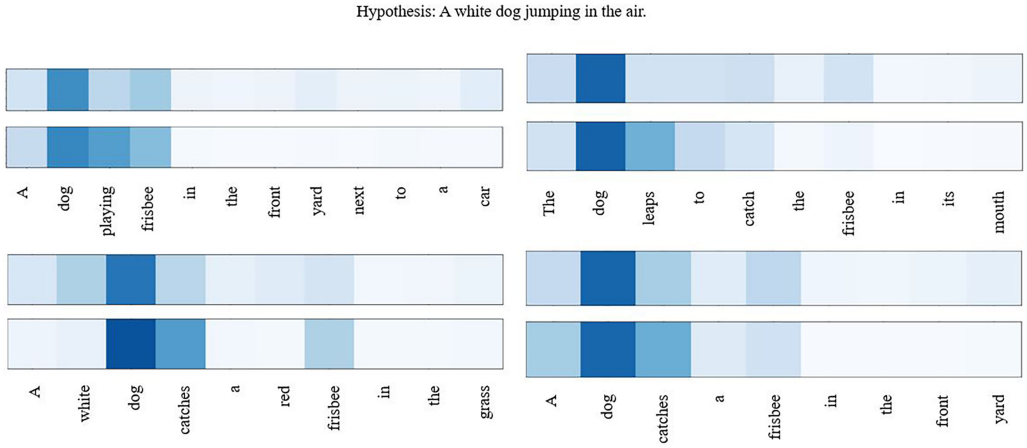

Figure 7. Attention visualizations.

It can be seen that the subject 'dog' in the hypothesis only appears in the second premise in Figure 8, and it is ignored by both Inner-attention and Attention-backtracking mechanism, which leads to the result that the model classifies this set of sentences as "contradiction", because the classification layer found that the semantics of the sentence pairs to be judged are "cat is sleeping"and "dog is sleeping", the subject is contrary, so gave a wrong classification.

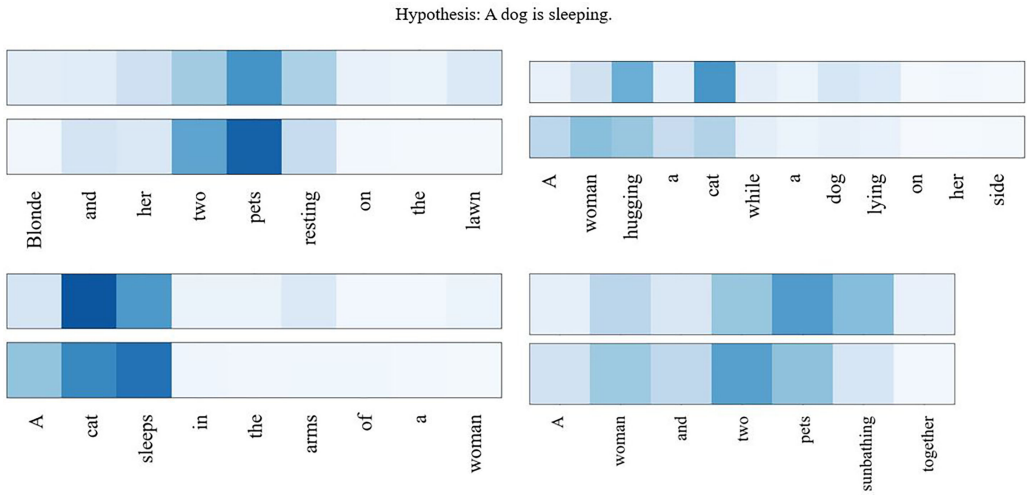

Figure 8. Attention visualizations for another example.

Table 2 shows the results based on the Soft-alignment pre-trained model, which is consistent with the conclusions above, and ABR (1,2,3) also achieved the highest accuracy.

In summary, we have made a lot of comparative experiments. When the three-step training method is used to train the r module of the fusion Attention-backtracking mechanism, both Inner-attention and Soft-alignment pre-trained model have reached the

highest accuracy. The accuracy rates are 64.4% and 68.2%, respectively. The effect of the voting method is second only to our method because the data set used is too balanced.

Table 2. Train/test accuracies on the MPE data set and number of parameters for each approach based on Soft-alignment.

Method	Parameters	Train	Test
Concat	51K	0.7914	0.6270
Concat (ft)	51K	0.7311	0.6430
Vote	51K	0.6940	0.6290
Vote (ft)	51K	0.7076	0.6710
AveR	51K	0.7625	0.6150
AveR (ft)	51K	0.7637	0.6670
ABR	71K	0.7846	0.6450
ABR (1,2)	20K	0.5431	0.5840
ABR (1,3)	71K	0.7546	0.6680
ABR (1,2,3)	71K	0.7058	**0.6820**
MLPR	71K	0.7331	0.6270
MLPR (1,2)	20K	0.6161	0.6460
MLPR (1,3)	71K	0.7946	0.6610
MLPR (1,2,3)	71K	0.6811	0.6640

5. Conclusions

We proposed a transferring method from the SPE model to the MPE model. The best migration effect can be achieved by adding a simple relation processing module and a training method adapted to the pre-training model.

We also explored what characteristics a good R module should have: (a) It should be independent of the encoding layer and the inference layer. (b) It should have the ability to fully integrate information between multiple premises. (c) The fused vector can be accepted by the inference layer.

Our R model has many advantages, such as: (a) It transfers the prior knowledge of single-premise tasks to multi-premise tasks through transfer learning. (b) It is valid for different basic models, including but not limited to Siamese framework and "matching-aggregation" framework. (c) It is more beneficial to solve the problem of data imbalance. (d) It effectively improves the accuracy of the results. However, there are also disadvantages: (a) The model requires much pre-training and fine-tuning, which increases the difficulty of model training. (b) The research on the R module is at a beginning stage.

In summary, in the future, we hope to study further the attention backtracking mechanism or a more effective R module that meets the above characteristics to achieve higher accuracy and reduce the complexity of model training.

Author Contributions: Conceptualization, P.W. and Z.L.; methodology, R.Z. and Z.L.; software, R.Z. and Z.L.; validation, Z.L.; formal analysis, R.Z.; investigation, Z.L.; resources, P.W.; data curation, R.Z.; writing—original draft preparation, R.Z.; writing—review and editing, R.Z.; visualization, Z.L.; supervision, P.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not Applicable, the study does not report any data.

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

References

1. Lai, A.; Bisk, Y.; Hockenmaier, J. Natural language inference from multiple premises. *arXiv* **2017**, arXiv:1710.02925.
2. Dagan, I.; Glickman, O. Probabilistic textual entailment: Generic applied modeling of language variability. *Learn. Methods Text Underst. Min.* **2004**, *2004*, 26–29.
3. Korman, D.Z.; Mack, E.; Jett, J.; Renear, A.H. Defining textual entailment. *J. Assoc. Inf. Sci. Technol.* **2018**, *69*, 763–772. [CrossRef]

4. Liu, Y.; Sun, C.; Lin, L.; Wang, X. Learning natural language inference using bidirectional LSTM model and inner-attention. *arXiv* **2016**, arXiv:1605.09090.
5. Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; Zhang, C. Disan: Directional self-attention network for rnn/cnn-free language understanding. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
6. Ghaeini, R.; Hasan, S.A.; Datla, V.; Liu, J.; Lee, K.; Qadir, A.; Ling, Y.; Prakash, A.; Fern, X.Z.; Farri, O. Dr-bilstm: Dependent reading bidirectional lstm for natural language inference. *arXiv* **2018**, arXiv:1802.05577.
7. Gong, Y.; Luo, H.; Zhang, J. Natural language inference over interaction space. *arXiv* **2017**, arXiv:1709.04348.
8. Parikh, A.P.; Täckström, O.; Das, D.; Uszkoreit, J. A decomposable attention model for natural language inference. *arXiv* **2016**, arXiv:1606.01933.
9. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
10. Rocktäschel, T.; Grefenstette, E.; Hermann, K.M.; Kočiskỳ, T.; Blunsom, P. Reasoning about entailment with neural attention. *arXiv* **2015**, arXiv:1509.06664.
11. Williams, A.; Nangia, N.; Bowman, S.R. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv* **2017**, arXiv:1704.05426.
12. Huh, M.; Agrawal, P.; Efros, A.A. What makes ImageNet good for transfer learning? *arXiv* **2016**, arXiv:1608.08614.
13. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 818–833.
14. Bengio, Y.; Laufer, E.; Alain, G.; Yosinski, J. Deep generative stochastic networks trainable by backprop. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 226–234.
15. Bengio, Y.; Ducharme, R.; Vincent, P.; Jauvin, C. A neural probabilistic language model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
16. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
17. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf (accessed on 3 March 2021).
18. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
19. Liu, X.; He, P.; Chen, W.; Gao, J. Multi-task deep neural networks for natural language understanding. *arXiv* **2019**, arXiv:1901.11504.
20. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *8*, 9.
21. Mou, L.; Meng, Z.; Yan, R.; Li, G.; Xu, Y.; Zhang, L.; Jin, Z. How transferable are neural networks in nlp applications? *arXiv* **2016**, arXiv:1603.06111.
22. Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; Shah, R. Signature verification using a "siamese" time delay neural network. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 28 November–1 December 1994; pp. 737–744.
23. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
24. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef] [PubMed]
25. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, arXiv:1802.05365.
26. Liu, C.; Jiang, S.; Yu, H.; Yu, D. Multi-turn inference matching network for natural language inference. In *CCF International Conference on Natural Language Processing and Chinese Computing*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 131–143.
27. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
28. Boureau, Y.L.; Bach, F.; LeCun, Y.; Ponce, J. Learning mid-level features for recognition. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 2559–2566.
29. Bowman, S.R.; Angeli, G.; Potts, C.; Manning, C.D. A large annotated corpus for learning natural language inference. *arXiv* **2015**, arXiv:1508.05326.
30. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
31. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
32. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
33. Howard, J.; Ruder, S. Universal language model fine-tuning for text classification. *arXiv* **2018**, arXiv:1801.06146.