Natural Language Engineering (2023), **29**, pp. 1066–1096
doi:10.1017/S135132492200050X

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

Parameter-efficient feature-based transfer for paraphrase identification

Xiaodong Liu[1,*], Rafal Rzepka[2] and Kenji Araki[2]

[1]Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Hokkaido, Japan and [2]Faculty of Information Science and Technology, Hokkaido University, Sapporo, Hokkaido, Japan
*Corresponding author. E-mail: xiaodongliu@ist.hokudai.ac.jp

(Received 23 August 2021; revised 19 November 2022; accepted 21 November 2022;
first published online 19 December 2022)

Abstract

There are many types of approaches for Paraphrase Identification (PI), an NLP task of determining whether a sentence pair has equivalent semantics. Traditional approaches mainly consist of unsupervised learning and feature engineering, which are computationally inexpensive. However, their task performance is moderate nowadays. To seek a method that can preserve the low computational costs of traditional approaches but yield better task performance, we take an investigation into neural network-based transfer learning approaches. We discover that by improving the usage of parameters efficiently for feature-based transfer, our research goal can be accomplished. Regarding the improvement, we propose a pre-trained task-specific architecture. The fixed parameters of the pre-trained architecture can be shared by multiple classifiers with small additional parameters. As a result, the computational cost left involving parameter update is only generated from classifier-tuning: the features output from the architecture combined with lexical overlap features are fed into a single classifier for tuning. Furthermore, the pre-trained task-specific architecture can be applied to natural language inference and semantic textual similarity tasks as well. Such technical novelty leads to slight consumption of computational and memory resources for each task and is also conducive to power-efficient continual learning. The experimental results show that our proposed method is competitive with adapter-BERT (a parameter-efficient fine-tuning approach) over some tasks while consuming only 16% trainable parameters and saving 69-96% time for parameter update.

Keywords: Parameter-efficient feature-based transfer; Paraphrase identification; Natural language inference; Semantic textual similarity; Continual learning

1. Introduction

Measuring semantic relatedness of two pieces of text entails a wide range of tasks in Natural Language Processing (NLP). When the measurement comes to sentence level, it involves three common NLP tasks: Paraphrase Identification (PI) for determining whether a sentence pair has equivalent semantics; Natural Language Inference (NLI) for inferring relation between a sentence pair; and Semantic Textual Similarity (STS) for scoring semantic similarity of a sentence pair. Many types of approaches have been published in the previous works (Chandrasekaran and Mago 2020) for PI task and two of its variations (NLI and STS). In this article, we propose a new method to improve the usage of parameters efficiently for feature-based transfer, a transfer learning approach involving customizing a task-specific architecture for PI. In this section, we first introduce the research goal that motivates our proposed method based on the relevant technical background. Then in the second subsection, we describe the benefit for a real-world scenario based on our technical novelty.

1.1 Technical background and research goal

When it comes to PI, there exist a wide range of traditional approaches that are relatively effective for the task: (1) lexical overlap features such as n-gram overlap (Wan *et al.* 2006) and machine translation evaluation metrics (Madnani, Tetreault, and Chodorow 2012); (2) using external lexical knowledge like WordNet (Fellbaum, 1998; Fernando and Stevenson 2008); (3) modeling divergence of dependency syntax between two sentences (Das and Smith 2009); (4) distributional models with matrix factorization (Guo and Diab 2012; Ji and Eisenstein 2013). The traditional approaches mainly consist of unsupervised methods and feature engineering. Their demand for computational resources is low while task performance is moderate nowadays. For example, the computational cost involving parameter update for processing the lexical overlap features is only consumed by classifier-tuning, but the task performance is comparatively less effective as reflected by MRPC task (Dolan, Quirk, and Brockett 2004) in the ACL link.[a] In light of that fact, we come up with our research question—*how can we preserve the low computational costs of traditional approaches but yield better task performance?* To seek an approach to it as our research goal, we take a further investigation on current neural network-based transfer learning approaches.

With the advent of various deep neural network models (Dong *et al.* 2015; Vaswani *et al.* 2017; Howard and Ruder 2018; Yang *et al.* 2019), transfer learning approaches have achieved state-of-the-art performance on many NLP downstream tasks including PI. Basically, there are two most common transfer learning techniques in NLP: fine-tuning and feature-based transfer. As for fine-tuning, the parameters of pre-trained language models like BERT (Devlin *et al.* 2019) need to be fine-tuned. On the other hand, feature-based transfer does not require parameter update of pre-trained embedding models like ELMo (Peters *et al.* 2018); however, the parameters of any customized task-specific architectures need to be updated. For both transfer learning techniques, they follow the same convention: entire architectural parameters need to be initialized and then updated for each individual task dataset (see 1 & 2 in Figure 1).

Accompanied by the successful performance, there comes one challenge to transfer learning approaches: expensive computational resources (Strubell, Ganesh, and McCallum 2019). As fine-tuning tends to achieve better performance than feature-based transfer shown in recent works (Howard and Ruder 2018; Devlin *et al.* 2019), recently proposed resource-lean approaches are mainly based on BERT (Devlin *et al.* 2019) architecture, focusing on model compression such as network layers repeating (Lan *et al.* 2019) and knowledge distillation (Sanh *et al.* 2019). The effectiveness of these compressed models can reduce the size of entire architectural parameters, but cannot go beyond the aforementioned convention, and thus, whether the usage of the parameters is efficient remains uninvestigated.

Still, the adapter module proposed by Houlsby *et al.* (2019a) is based on the research direction regarding parameter efficiency—only a comparatively small number of task-specific parameters are initialized and then updated for every single task (see 3 in Figure 1). Houlsby *et al.* (2019a) applied the adapter module to BERT (Devlin *et al.* 2019) for testing effectiveness, and the adapter-BERT attained within 0.4% of the performance of full fine-tuning, adding only 3.6% parameters per task. Such performance level reflects the fact that there is no need to initialize and then to update entire architectural parameters for each task. Instead, focus should be shed on the efficient usage of parameters. Most recently, the research direction has a tendency to extract an optimal subset of architectural parameters (de Wynter and Perry 2020) and also works for autoregressive models like P-tuning (Liu *et al.* 2021) for GPT (Radford *et al.* 2019).

The efficient usage of architectural parameters also resonates with a current trend[b] in NLP community, which encourages researchers to empirically justify the model complexity beyond benchmark performance. However, to the best of our knowledge as of writing, direct research attempts regarding parameter efficiency are rarely made for feature-based transfer. In the last

[a] https://aclweb.org/aclwiki/Paraphrase_Identification_(State_of_the_art).
[b] https://sites.google.com/view/sustainlp2020/home?authuser = 0.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

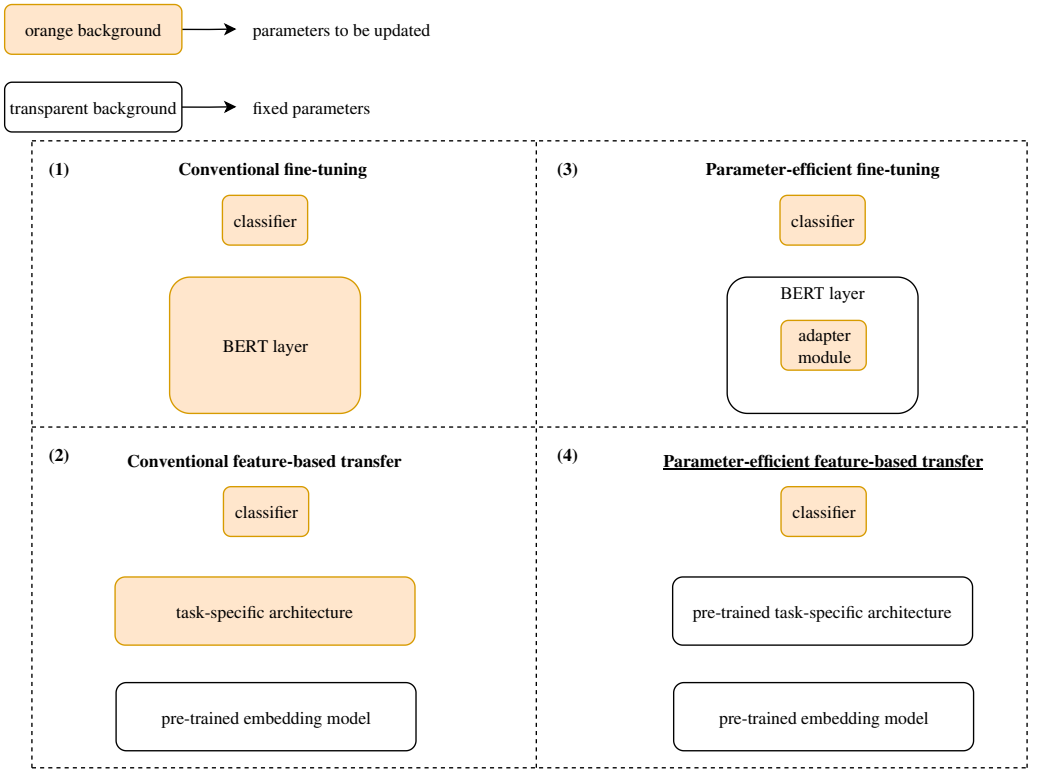

Figure 1. Parameters that need to be updated for each PI, NLI, or STS task. The underlined denotes our work.

two years, only one indirect attempt was conducted, the PAR (Paraphrase-Aware Retrofitting) method proposed by Shi *et al.* (2019), aiming to address unstable semantics of contextualized word embeddings of shared words when context is paraphrased. Besides, for any customized task-specific architecture, although it is an option to train task datasets through continual learning (Thrun 1998), the parameters of the re-trained network are inclined to forget how to perform previous tasks—catastrophic forgetting (McCloskey and Cohen 1989; French 1999).

While related works are centered around language model-based fine-tuning, we discover that one advantage of feature-based transfer tends to be neglected and consequently left unexplored. As various task-specific architectures are customized for a particular task like PI, we discover that it is viable to fix the architectural parameters trained on a single task dataset and then transfer the fixed parameters to other task datasets. With this discovery, while yielding better task performance, feature-based transfer can enjoy the low computational costs as traditional approaches do: the initialization of pre-trained architectural parameters is required only once, and there is no need of further parameter update for different tasks (see 4 in Figure 1), the only computational costs are consumed by classifier-tuning. The technical scenario cannot be realized by language model-based fine-tuning. For example, task-specific parameters of adapter module (Houlsby *et al.* 2019b) tuned on task A cannot be directly used for task B without any modification. However, in this article, we show that it is feasible to feature-based transfer.

For readability in the rest of this article, we use the acronym PEFBAT (Parameter-Efficient Feature-BAsed Transfer) to denote our proposed method. The technical novelty of PEFBAT is that the fixed parameters of the pre-trained task-specific architecture can be shared by multiple classifiers with small additional parameters. This mechanism can address our research goal. For each PI, NLI, or STS task, the computational cost left involving parameter update is only generated

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

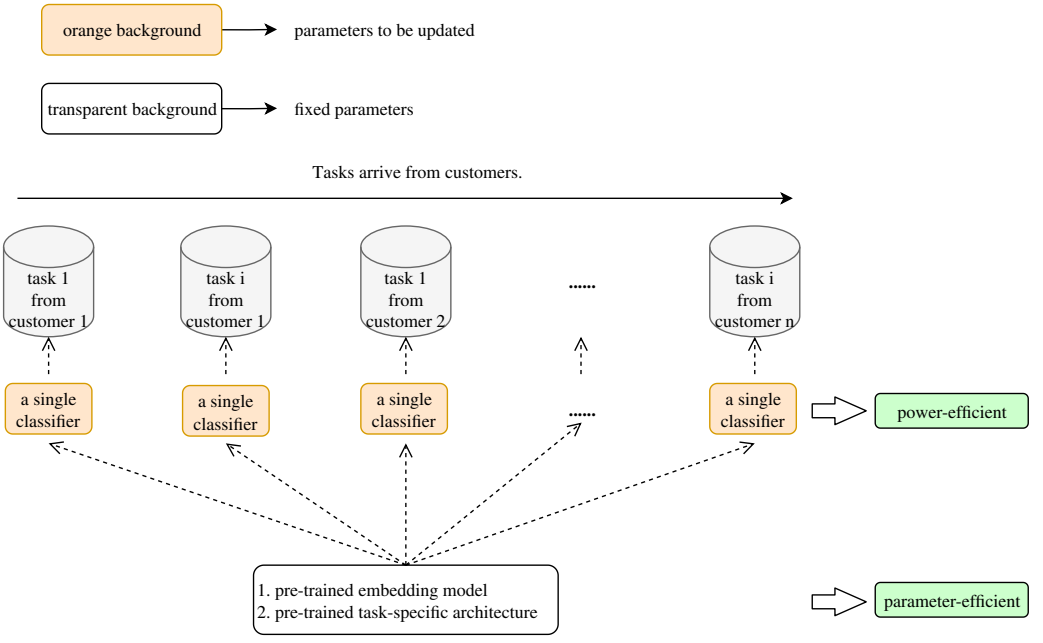

Figure 2. Parameter-efficient feature-based transfer for power-efficient continual learning.

from classifier-tuning: the features output from the architecture combined with lexical overlap features are fed into a single classifier for tuning. Such technical novelty is also conducive to a real-world scenario, which is another contribution of PEFBAT described in the next subsection.

1.2 Practical use in real-world scenario

Similar to adapter module (Houlsby *et al.* 2019b), PEFBAT can be applied to continual learning, but the differences exist from two aspects. Since the former is language model-based (e.g., adapter-BERT when it is applied to BERT model), it can handle more than sentence-pair tasks. On the other hand, although PEFBAT can address only three types of sentence-pair tasks, it exhibits a more power-efficient manner: for each task arriving from customers, only a single classifier needs to be instantiated (see Figure 2). The power cost is measured by the time required for parameter update of each task, which directly reflects the demand for computational resources. For example, in the case of Multi-Layer Perceptron (MLP)[c] with batch size 32 and epochs 100, for small datasets like MRPC (Dolan *et al.* 2004) (4k training data), training a MLP takes up approximately 1 minute. For large datasets like QQP (Iyer, Dandekar, and Csernai 2017) (363k training data), training a MLP can be finished in 37 minutes. More importantly, each task performance is not compromised by catastrophic forgetting as each classifier is tailored individually for each task.

Furthermore, additional parameters per task contained in each classifier are small (0.479M trainable parameters explained in Section 4), which can be considered as parameter-efficient model expansion, as the size of 230 classifiers for 230 tasks from customers amounts to a single BERT-base model. Although using only engineered features also consumes less computational and memory resources, PEFBAT can yield better task performance, and thus, it is comparatively a good fit in terms of power-efficient continual learning.

[c]The GPU device that we use for our experiments is NVIDIA RTX 2080 TI.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

2. Related works

There have been many approaches proposed in the previous works for PI (Chandrasekaran and Mago 2020). In this section, we discuss the traditional and transfer learning approaches that are related to our work. The strengths of the traditional approaches are integrated into PEFBAT. The transfer learning approaches, although adopt different strategies, are related to parameter efficiency. Besides, we extend our discussion on two topics: (1) task-specific DNNs, which are the conventional strategy adopted for feature-based transfer, and (2) the solutions to catastrophic forgetting in continual learning, because PEFBAT can be considered as a variant of parameter-efficient model expansion.

2.1 Traditional approaches

In PEFBAT, the lexical overlap features (Wan *et al.* 2006) are combined with the transferred features (the features output from our pre-trained task-specific architecture) as input to each individual classifier. This technique is not unusual for feature-based transfer; for example, Yin and Schütze (2015a) combined the machine translation metrics (Madnani *et al.* 2012) with the flattened features output from their Bi-CNN architecture. In our case, the combination is based on a particular consideration—mutual complementation. As the Jaccard distance[d] illustrated in Figures 3 and 4 for two paraphrase corpora,[e] lexical overlap features are noticeably cost-effective for task datasets like PAN (Madnani *et al.* 2012) (Figure 3); however, they become unviable when lexical overlaps are indistinguishable between paraphrase and non-paraphrase sentence pairs in task datasets like PAWS-wiki (Zhang, Baldridge, and He 2019) (Figure 4). The combination can have the transferred features enjoy their merits while their demerits can be improved by the transferred features. We delve deeper into the mutual complementation in Section 4 based on our experimental results. From another perspective that is not directly related to our work, lexical overlap features are also beneficial to paraphrase generation task. While the quality of generated paraphrases can be decided by state-of-the-art models like Sentence-BERT (Reimers and Gurevych 2019) shown in a recent work (Corbeil and Abdi Ghavidel 2021) for data augmentation, some works still consider lexical overlap features as criteria: Nighojkar and Licato (2021) use BLEURT (Sellam, Das, and Parikh 2020) metric to calculate reward for sentence pairs that are mutually implicative but lexically and syntactically disparate; Kadotani *et al.* (2021) use edit distance to decide whether source and target sentences require drastic transformation, so that the training order of curriculum learning (Bengio *et al.* 2009) can be determined for better performance of paraphrase generation; Jaccard distance is used in Meng *et al.* (2021)'s work as one metric for filtering generated paraphrase candidates.

Ji and Eisenstein (2013) utilize TF-KLD weighting scheme to assign weights to each feature (single word) in distributional models. The weighted distributional models then can generate discriminative sentence latent representations after matrix factorization, which are conducive to PI. This approach has achieved competitive performance on MRPC (Dolan *et al.* 2004) task with this special feature engineering process. Inspired by Ji and Eisenstein (2013), the pre-trained task-specific architecture of PEFBAT is also designed to generate the features containing discriminative semantics (details are presented in Section 3). In the meanwhile, two of their limitations are not reflected in PEFBAT: (1) the TF-KLD weighting scheme relies on transductive learning (Gammerman, Vovk, and Vapnik 1998) to weight unseen words for optimal performance; (2) the scheme is strictly MRPC-dependent, and therefore it is not applicable to other PI task datasets or real-world scenarios. TF-KLD-KNN proposed by (Yin and Schütze, 2015b) can address the first limitation, but is not viable for the second limitation.

[d] All the tokens are converted into small case without stemming or removing stopwords and punctuation.
[e] Jaccard distance calculated for two PI corpora take into account training, development (if it exists), and test sets.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

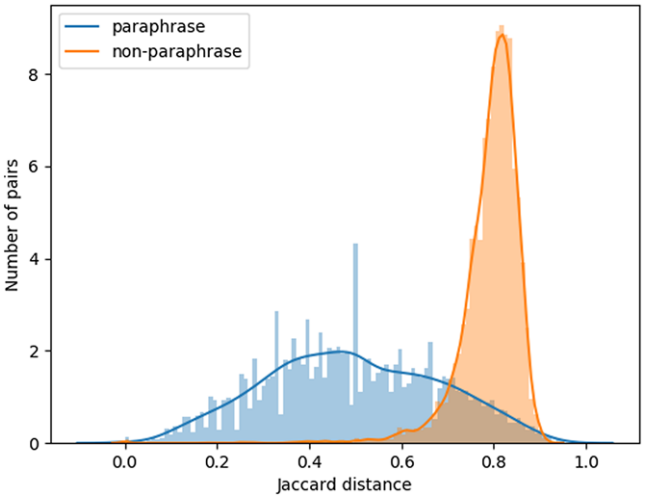

Figure 3. PAN.

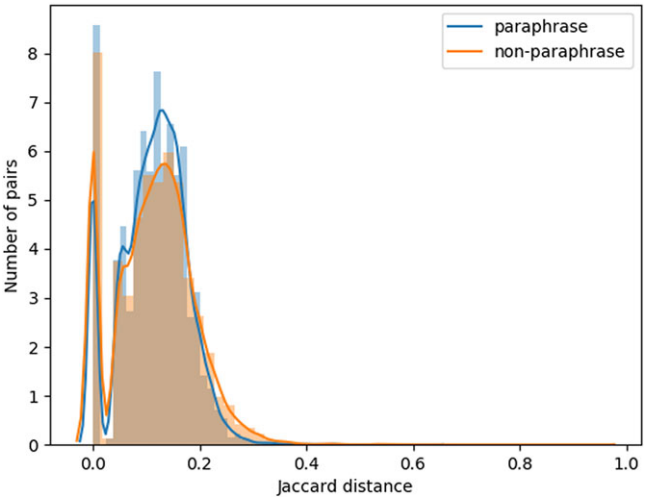

Figure 4. PAWS-wiki.

2.2 Transfer learning approaches

Shi *et al.* (2019) have discovered that in many cases, contextualized embeddings of shared words in paraphrased contexts change drastically. To minimize the difference of the shared words, they propose a PAR (Paraphrase-Aware Retrofitting) method: to reshape input representations of contextualized models with an orthogonal transformation matrix. They apply the PAR method to ELMo (Peters *et al.* 2018) to test its effectiveness. The task-specific architecture (orthogonal transformation matrix) placed prior to embedding models is referred to as retrofitting methods (Faruqui *et al.* 2015; Yu *et al.* 2016; Glavaš and Vulić, 2018): to incorporate semantic knowledge from external resources into word embeddings. The external resources that Shi *et al.* (2019) used to train their task-specific architecture are the paraphrase sentence pairs from three corpora: PAN (Madnani *et al.* 2012), Sampled Quora (Iyer *et al.* 2017), and MRPC (Dolan *et al.* 2004).

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

The parameters of the trained architecture then can be used for PI, NLI, and STS tasks, which is parameter-efficient.

The adapter module (Houlsby *et al.* 2019a) comprises the network layers with comparatively small size of parameters stitched into BERT (Devlin *et al.* 2019) layers, aiming to address parameter efficiency for language models. For every task, the parameters of BERT layers remain fixed while the ones of adapter module are updated. Such design results in a model that is compact and extensible: a small number of additional parameters per task without forgetting how to perform previous tasks. The adapter module can be considered as a variation of layer transfer. The technique is commonly adopted in the field of computer vision. For example, in the task of image classification (Deng *et al.* 2009), Yosinski *et al.* (2014) discovered that by transferring only the bottom layer of the network trained on source data, decent performance can be obtained for target data by only re-training the top layers, because the image data tend to share similar patterns at the lower layers of the network. The adapter module exhibits similar property as it automatically prioritizes higher layers (Houlsby *et al.* 2019a), which matches the popular strategy in fine-tuning (Howard and Ruder 2018). PEFBAT also benefits from the strategy: pre-trained embeddings plus task-specific architecture (fixed parameters at lower layers), and classifier (parameters at higher layers updated for different tasks). In Section 4, adapter-BERT is considered as our upper bound.

2.3 Task-specific DNNs

When it comes to feature-based transfer, it is a vital step to customize a task-specific architecture. The input of the architecture is pre-trained word embeddings (Mikolov *et al.* 2013; Pennington, Socher, and Manning 2014; Mikolov *et al.* 2018; Peters *et al.* 2018), and output is typically flattened features connected to a classifier. The conventional strategy adopted for the task-specific architecture is task-specific DNNs. One successful embodiment is Siamese neural network (Bromley *et al.* 1993), which shows the key insight of extracting interaction features from input word embeddings of two sentences at multiple levels of granularity (unigram, short n-gram, long n-gram, and sentence levels). A comparatively effective implementation of Siamese architecture is Bi-CNN (Yin and Schütze 2015a; He, Gimpel, and Lin 2015; Yin *et al.* 2016): using two sub-networks (double convolutional layers) to process word embeddings of two sentences. In the Bi-CNN architecture, each sentence of a sentence pair at the beginning is represented as a matrix, in which every column vector corresponds to a word embedding. Then the matrix representations of two sentences are processed by convolution filters with n-gram width and multiple types of pooling at different network layers, during which, interaction features representing multi-granular semantics are extracted from the gradually processed matrix representations. The final step is to connect the flattened multi-granular interaction features to a classifier for supervised learning.

Siamese architecture implemented by Bi-CNN is not the only approach to extracting multi-granular interaction features. For example, the RAE model proposed by Socher *et al.* (2011) first uses recursive neural network (also known as TreeRNN) to pre-train embeddings at word, phrase, and sentence levels. Those pre-trained embedding representations are denoted as multi-granular nodes. Then for the nodes of a sentence pair, a $n_1 \times n_2$ similarity matrix is computed as interaction features, where n_1 and n_2 are the number of nodes of two sentences, respectively, and each similarity is the Euclidean distance between two nodes. Finally, the similarity matrix is fed into a dynamic pooling layer to fix its size for supervised learning, as each sentence pair has different size of nodes. The RAE model is comparatively less effective than Bi-CNN in terms of the performance on MRPC task (Dolan *et al.* 2004). As explained in Yin and Schütze (2015a)'s work, this is due to unavailability of highly accurate parsers for tree structure.

PEFBAT can yield competitive performance level compared to Bi-CNN (demonstrated in Section 4), although its mechanism for PI (described in Section 3) is different. From technical

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

perspective, unlike Bi-CNN, the task-specific architecture of PEFBAT is pre-trained, and therefore there is no need of further parameter update for each task, which is parameter-efficient and cost-friendly.

2.4 Continual learning

Continual learning (Thrun 1998), also known as lifelong learning, never-ending learning, or incremental learning, is a machine learning technique of training tasks sequentially using a single instance of a model. The task range of continual learning nowadays is typically the same task but in different domains; for example 20 QA tasks in bAbi corpus (Weston *et al.* 2015), permuted handwritten digits recognition (van de Ven and Tolias 2019), text classification tasks with different class (de Masson d'Autume *et al.* 2019), and so on. A recent research attempt (Sun, Ho, and Lee 2020) has managed to handle 5 disparate NLP tasks by following decaNLP (McCann *et al.* 2018) to treat all tasks as QA format. The biggest problem for continual learning is catastrophic forgetting (McCloskey and Cohen 1989; French 1999)—the network trained on a new task is inclined to forget how to perform previous tasks. The common solutions to catastrophic forgetting can be concluded into three categories listed below.

(1) regularization-based methods: the paradigm of this method is EWC (Elastic Weight Consolidation) (Kirkpatrick *et al.* 2017), the key mechanism of which is to add constraints to the parameters that are sensitive to previous tasks. As a result, those sensitive parameters are not modified to a large extent when a new task is being trained.

(2) parameter-efficient model expansion: from earlier works like Net2Net (Chen, Goodfellow, and Shlens 2016) and Progressive Neural Networks (Rusu *et al.* 2016) to adapter module (Houlsby *et al.* 2019b) nowadays, the major concept of this solution is to expand a model by adding additional parameters per task. It can also work out in a reverse way—initiate a large network at the beginning and then distribute a portion of parameters to each task (Hung *et al.* 2019).

(3) memory replay: the main idea of this solution is to keep a small number of old data of previous tasks, and the old data can be either real data (de Masson d'Autume *et al.* 2019) or generated pseudo-data (Sun *et al.* 2020). With this benefit, a new task can be trained together with the old data in a multi-task learning manner (Caruana 1998), which is considered as the upper bound of continual learning. It is also shown in GEM (Gradient Episodic Memory) (Lopez-Paz and Ranzato 2017) that the old real data can be used to prevent gradient update from being biased towards a new task during optimization.

Curriculum learning (Bengio *et al.* 2009), although not as common as the categories above, is also a solution to catastrophic forgetting. It seeks to find out a proper learning order of tasks. A well-known study of curriculum learning is taskonomy (task + taxonomy) proposed by Zamir *et al.* (2018), which explores the learning order of various image-related tasks; for example detecting 3D edges and normal vectors of images first can help learn point matching and reshading effectively.

PEFBAT can be considered as a variant of the second solution. The limitation of the second solution is that when task number grows larger and larger, it will eventually use up memory resources. However, as we discussed in Section 1.2, PEFBAT can alleviate this problem as parameters contained in each classifier is small—the size of 230 MLP classifiers for 230 tasks is equal to a single BERT-base model. If we use SVM instead of MLP as the classifier for small tasks like MRPC (Dolan *et al.* 2004) without sacrificing task performance, the total size becomes further smaller. Hence, PEFBAT is not only a solution to catastrophic forgetting but also competent in power-efficient continual learning.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

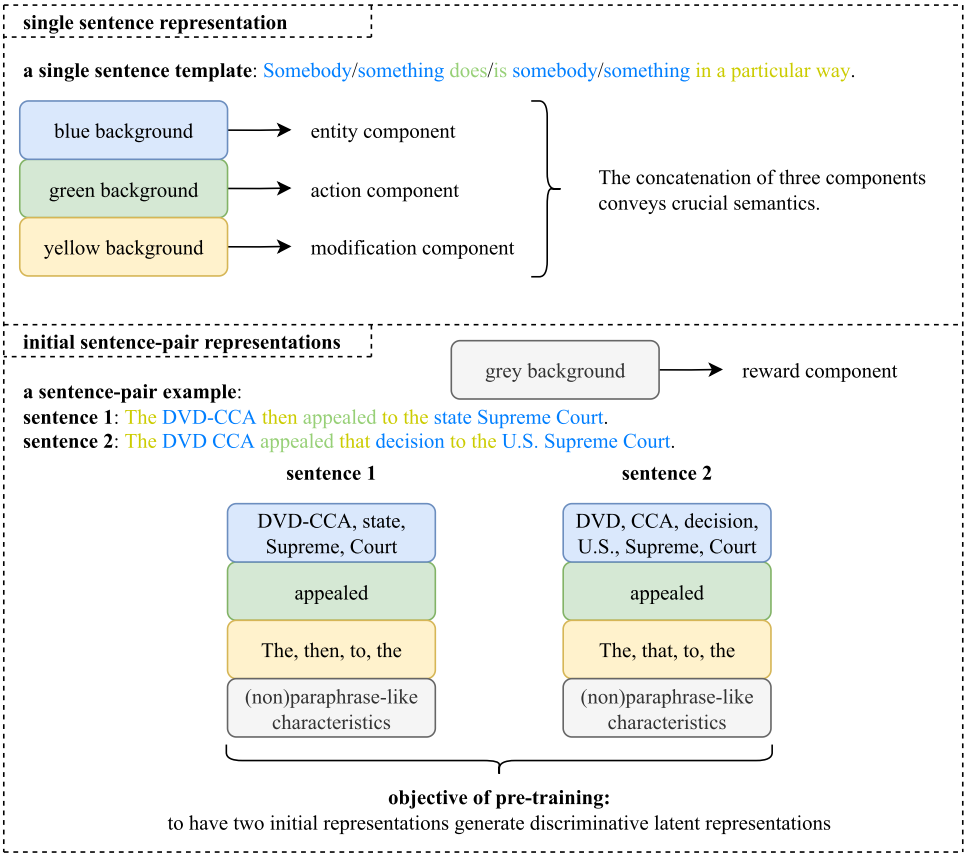

Figure 5. Modeling of initial sentence-pair representations.

3. Proposed method

Our technical motivation towards PEFBAT is introduced in Section 1.1. In this section, we move into the implementation details and mechanism of PEFBAT. We first describe the modeling of initial sentence-pair representations and the objective of our pre-training. Then how to pre-train the task-specific architecture of PEFBAT using the representations is described in the second subsection. The third subsection explains the discriminative features that our pre-trained task-specific architecture can output, which are presented together with other features in the last subsection.

3.1 Initial sentence-pair representations and objective of pre-training

In this work, the crucial semantics conveyed in a single sentence are represented by the three components (see the upper part of Figure 5). The entity and action components basically describe "what is the state or activity that somebody or something is on." The modification component conveys the semantics that modify the entity and action components. For example, "in a particular way" can be interpreted as either adverbs that modify verbs or determiners and adjectives that modify nouns.

The concatenation of three components plays the important role in measuring semantic similarity of a sentence pair (see the lower part of Figure 5). In addition, to facilitate PI, the reward component (details explained in Sections 3.2.1 and 3.2.2) containing (non)paraphrase-like

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

characteristics is combined with the other three components to form the initial sentence representations of a sentence pair. The objective of our pre-training, therefore, is to have the initial representations generate discriminative latent representations. The latter contains discriminative features, which is conducive to PI.

3.2 Latent space pre-training

To generate discriminative sentence latent representations, we refer to Ji and Eisenstein (2013)'s work by taking advantage of labeled data, and our concrete strategy in this article is the dual weighting scheme shown in Figure 6. The paraphrase weighting scheme is used to weight five components of a sentence belonging to paraphrase sentence pairs. Then the concatenation of five weighted components is mapped to its latent representation for pre-training the paraphrase latent space. The same step is applied to pre-training the non-paraphrase latent space, where the components of non-paraphrase sentences are weighted by the non-paraphrase weighting scheme. Besides, two reward components are used to represent (non)paraphrase-like characteristics, which is the best setting for pre-training. Later in Section 4, an ablation study is performed for validation.

The two pre-trained latent spaces are namely our task-specific architecture, the fixed parameters of which can be used for multiple PI, NLI, and STS tasks. During inference shown in Figure 7, for each sentence of any sentence pairs, two types of weighted components based on the dual weighting scheme are first concatenated, respectively, and then two latent representations can be obtained after two concatenations are projected from two pre-trained latent spaces, respectively—paraphrase and non-paraphrase latent representations. They contain discriminative features, which is described at length in Section 3.3.

Before stepping into further implementation details, we introduce the embeddings, toolkits, and corpus resource that are employed for pre-training. As we consider the application to low-resource natural languages with limited labeled and unlabeled data (Hedderich *et al.* 2021) as our future work (discussed in the last section), the choice of the resources is based on the consideration of "lightness": requirement of small training data size and easy availability.

Corpus Resource: The corpus that we use is only the training dataset of Microsoft Research Paraphrase Corpus (MRPC) (Dolan *et al.* 2004) consisting of 4076 sentence pairs, in which 2753 pairs are labeled as paraphrase, which is so far the smallest English paraphrase corpus.
Pre-trained Embeddings: We choose fastText (Mikolov *et al.* 2018) trained with subword information on CommonCrawl as our pre-trained embeddings because it is not deep learning architecture-based, which is applicable to low-resource natural languages (Mohiuddin and Joty 2020). Moreover, it achieves better performance than GloVe (Pennington *et al.* 2014) trained on CommonCrawl in SentEval framework (Conneau and Kiela 2018).

Toolkits: Considering easy availability, we utilize part-of-speech tags (POS-tags) provided by NLTK[f] for categorizing entity, action, and modification components. The scikit-learn tool[g] is used for distributional models and matrix factorization.

3.2.1 Five components

Categorizing words as the elements of entity, action, and modification components is based on three specific sets of POS-tags shown below.

- **Entity set**: singular noun, plural noun, singular proper noun, plural proper noun, personal noun

[f] https://www.nltk.org/.
[g] https://scikit-learn.org/stable/.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

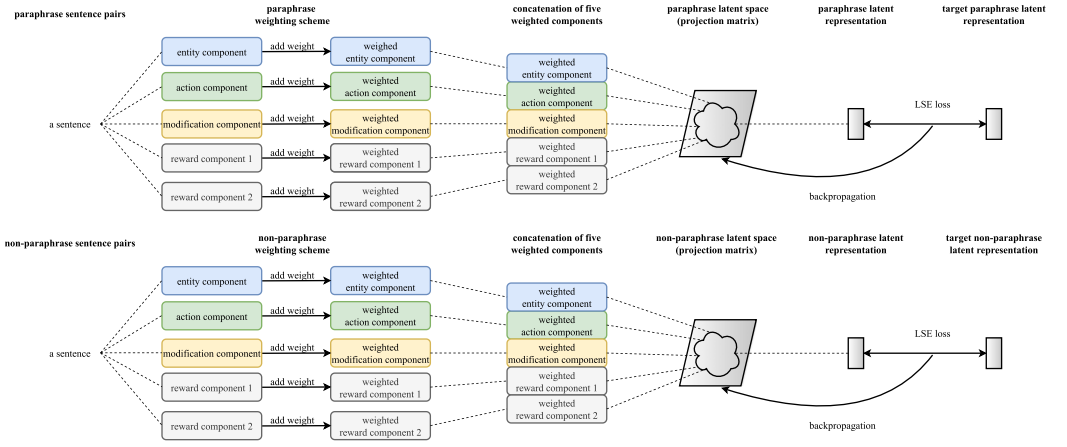

Figure 6. Mapping mechanism of pre-training.

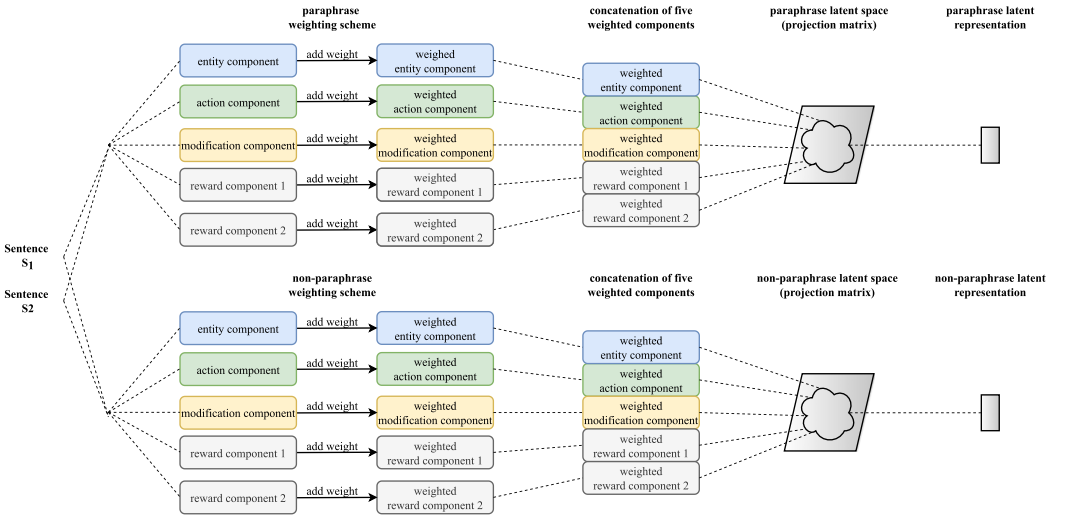

Figure 7. During inference, each sentence of any sentence pairs can obtain two types of latent representations.

- **Action set**: base form verb, past tense verb, present particle verb, past participle verb, present verb, third person present verb
- **Modification set**: determiner, predeterminer, adjective, comparative adjective, superlative adjective, possessive pronoun, possessive ending, existential there, modal, adverb, comparative adverb, superlative adverb, particle, to

Suppose the sentence "He likes apples." has three word embeddings (0.1, 0.1, 0.1) for 'he', (0.2, 0.2, 0.2) for 'likes', and (0.3, 0.3, 0.3) for 'apples', then the entity component is [(0.1, 0.1, 0.1) + (0.3, 0.3, 0.3)]/ 2 = (0.2, 0.2, 0.2); the action component is (0.2, 0.2, 0.2)/ 1 = (0.2, 0.2, 0.2); the modification component is (0, 0, 0). If the word embedding of a word does not exist in the embedding space, our strategy is to skip it, because the sentences in MRPC corpus are derived from old news resources, rarely containing newly coined words like "infodemic."

Each reward component is initiated as a vector with the same length as fastText word embeddings (300), and the values of all dimensions are initially set to 1.0. Two weighted reward

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Table 1. Dual weighting scheme.

Components	Paraphrase weighting scheme	Non-paraphrase weighting scheme
Entity	if digit-count ≤ 4:	if digit-count > 4:
	Entity $= 1.5 *$ Entity Component	*Entity* $= 1.5 *$ Entity Component
	else:	else:
	Entity $= 0.5 *$ Entity Component	*Entity* $= 0.5 *$ Entity Component
Action	*Action* $= 1.5 *$ Action Component	*Action* $= 0.5 *$ Action Component
Modi.	*Modi.* $= 0.5 *$ Modification Component	*Modi.* $= 1.5 *$ Modification Component
RC_1	if *sent-len* ≥ 23 or *sent-len-diff* ≤ 5:	if *sent-len* < 23 or *sent-len-diff* > 5:
	$RC_1 = 1 *$ Reward Component 1	$RC_1 = 1 *$ Reward Component 1
	else:	else:
	$RC_1 = 0.2 *$ Reward Component 1	$RC_1 = 0.2 *$ Reward Component 1
RC_2	if Jaccard distance ≤ 0.6:	if Jaccard distance > 0.6:
	$RC_2 = 1 *$ Reward Component 2	$RC_2 = 1 *$ Reward Component 2
	else:	else:
	$RC_2 = 0.2 *$ Reward Component 2	$RC_2 = 0.2 *$ Reward Component 2

components are meant to provide strong or weak rewards for the concatenation of components. With their functionality, sentence pairs that have (non)paraphrase-like characteristics tend to obtain similar latent representations in (non)paraphrase latent spaces.

3.2.2 Dual weighting scheme

Our dual weighting scheme is summarized in Table 1, which is utilized to determine whether a component should be strongly or weakly weighted. In the scheme, our weighting criteria are based on two elements: "label" (universal to all task datasets) and "occurrence difference" (manually adjustable for different task datasets).

For both reward components, our weighting criterion is "occurrence difference." We utilize the characteristics-based thresholds (see Figures 8, 9, and 10) to determine strong or weak rewards. The main threshold is Jaccard distance (abbreviated as *j-dist* shown in Figure 10) (Jaccard 1912). We perform PI on MRPC (Dolan *et al*. 2004) using Jaccard distance and found out at 0.6, accuracy and F1 score increase significantly. Thus, the threshold of 0.6 is utilized to determine one reward component. However, during inference, when the threshold of *j-dist* is adjusted for other task datasets, using the weighting criterion of "occurrence difference" is empirically proved enough in Section 4, which is time-saving. Only for pre-training our latent spaces, we need to ensure every default setting is well set. The sentence length (abbreviated as *sent-len* shown in Figure 8) and absolute difference of sentence length (abbreviated as *sent-len-diff* shown in Figure 9) are the main factors that can affect Jaccard distance, so we combine their thresholds together to determine another reward component so as to assist the one determined by *j-dist*. To briefly sum up, the three thresholds are considered as a whole package to determine whether two reward components should be weighted strongly or weakly, which is the way to represent whether a sentence pair exhibits (non)paraphrase-like characteristics.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

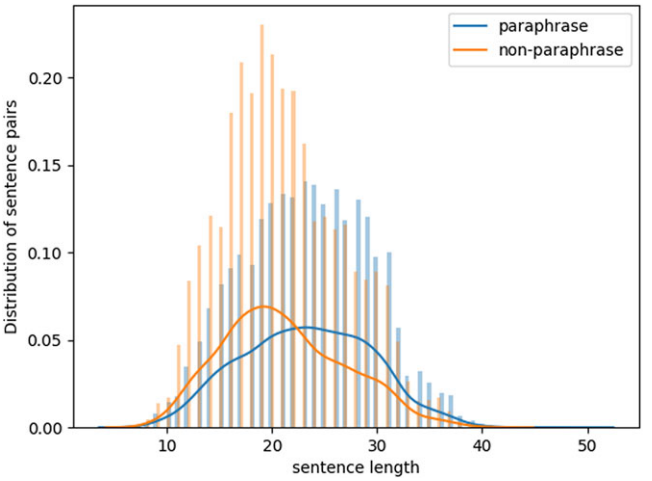

Figure 8. Sentence length.

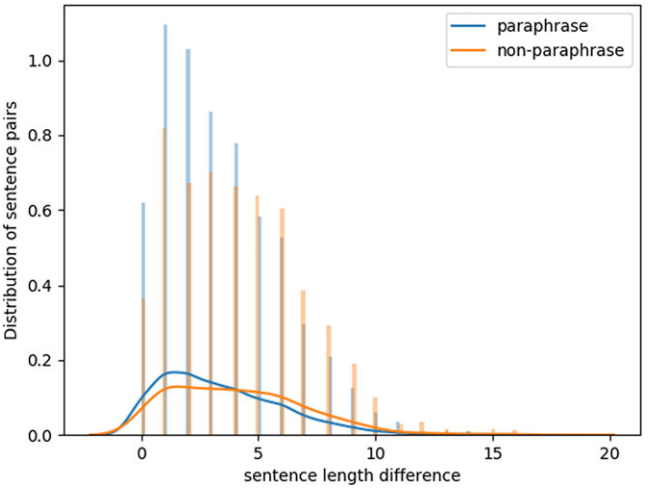

Figure 9. Absolute difference of sentence length.

For the three word embedding components (entity, action, and modification), our weighting criterion is "label." As adverbs like negation (not) convey discriminative semantics, we assign more weight to the modification component if a sentence pair is labeled as non-paraphrase. On the other hand, the entity and action components basically describe "what is the state or activity that somebody or something is on" (mentioned earlier in this subsection), which is the principal requirement for semantic similarity of sentences, so we assign more weight to the entity and action components if a sentence pair is labeled as paraphrase. The weighting criterion of "label" is universal to all task datasets. The *digit-count* threshold (see Figure 11) is the assumption we make that when there are too many digital numbers occurring in a sentence pair, the influence of semantics represented in the entity component should be lessened. To verify this assumption, we also pre-train another two latent spaces without the threshold. In Section 4, we conduct the experiments using the latent spaces pre-trained "*with digit-count*" or "*without digit-count.*" Below we provide an example from MRPC training dataset, where the sentence pair is labeled as non-paraphrase and *digit-count* is 6 (6 digital numbers in total are included in both sentences).

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

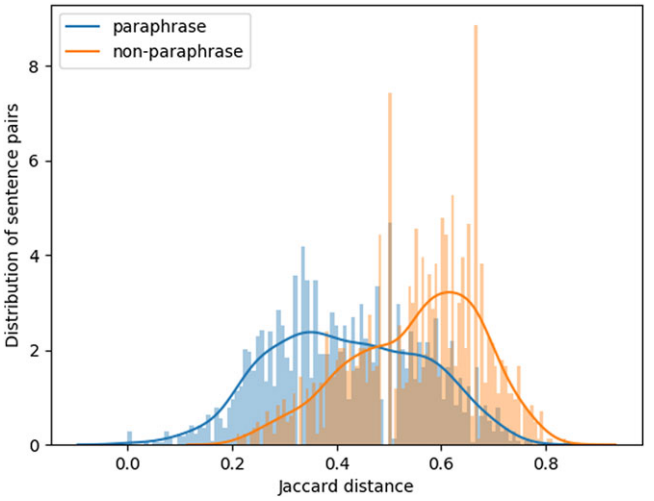

Figure 10. Jaccard distance.

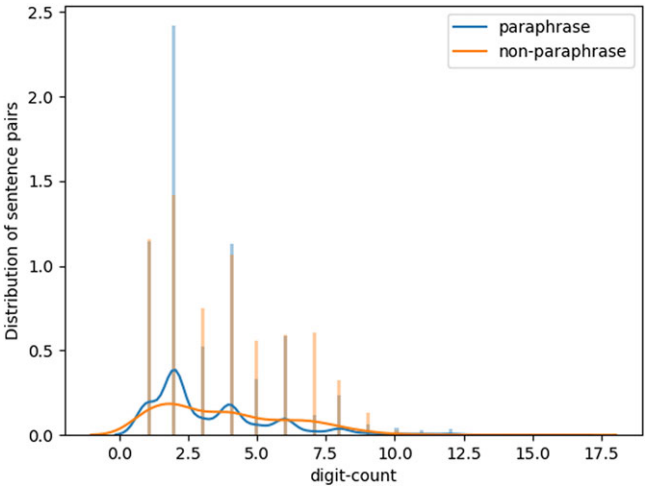

Figure 11. Digit-count.

(1) Yucaipa owned Dominick's before selling the chain to Safeway in 1998 for $2.5 billion.

(2) Yucaipa bought Dominick's in 1995 for $693 million and sold it to Safeway for $1.8 billion in 1998.

3.2.3 Target latent representations

We use distributional models with matrix factorization to generate latent representations for all sentences in the training dataset of MRPC (Dolan *et al*. 2004). For factorization, we follow Guo and Diab (2012)'s work by choosing Singular Value Decomposition (Deerwester *et al.* 1990) and the length of 100 for latent dimensionality. Moreover, we normalize the factorized matrix, as the

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

latent representations are the target layer of our architecture, and the activation function is tanh. As a result, all values in the factorized matrix are included within an open interval (−1, 1).

Furthermore, we refine the latent representations for paraphrase pairs. Suppose the latent representations for a sentence S_1 and a sentence S_2 of a sentence pair are (0.9, 0.8, 0) and (0.7, 0.8, 0.6), respectively, the pair shares a latent representation by averaging the elementwise addition of two vectors; in this case, the shared latent representation is (0.8, 0.8, 0.3). The rationale of sharing is that paraphrase pairs should have similar or even identical dimension values. By doing so, two initial sentence representations of any paraphrase sentence pair can be mapped to the same latent representation during pre-training, and therefore during inference, paraphrase pairs tend to obtain similar latent representations after mapping from paraphrase latent space.

3.2.4 Hyperparameters of pre-training and method of determining weights

For each sentence in the training dataset of MRPC (Dolan *et al*. 2004), the concatenation of its five weighted components is mapped to its latent representation. As mentioned earlier in this subsection shown in Figure 6, all (non)paraphrase sentences are used to pre-train the (non)paraphrase latent space based on (non)paraphrase weighting scheme.

Since the size of training dataset is small, back-propagation is performed for each mapping, following vanilla Stochastic Gradient Descent (vanilla SGD). The loss is accumulated from all samples of each training epoch, and the accumulation is averaged at the end of each epoch. The hyperparameters of our pre-training are input vector length (1500), target vector length (100), activation function (tanh), loss function (LSE), epochs (500), learning rate (5e-4), and optimization (vanilla SGD).

Before formal pre-training, we first tune the weights with small training epochs ($\leq$ 50). The determined weights are illustrated in Table 1. For both reward components, the strong and weak weights are 1 and 0.2, respectively; 1.5 and 0.5 (we call it "weight pair") are set for the three word embedding components. How they are determined is described below.

(1) First, we hypothesize that 1 is the optimal strong weight for both reward components, because if too big it will make the dimensions of other input components less significant; when too small it cannot be considered as a strong weight.

(2) Then we keep the "weight pair" (1.5 & 0.5) as it is, and begin to tune the weak weight from 0.1 to 0.2 for the best optimization of pre-training. After only 3 epochs of pre-training, we observe that the error loss decreases significantly in case of 0.2, and hardly drops when 0.1 is used. Therefore, we decide 0.2 as the weak weight for both reward components.

(3) After the decision of the weak weight, we start to tune the "weight pair." We use only 50 epochs to pre-train our latent spaces and experiment with different pairs from (1.1, 0.9) to (1.9, 0.1) at the pivot of 1. We found out that with the pair of (1.5, 0.5), better discriminative similarity (explained in the next subsection) can be achieved.

After the weights are determined, we set out to formally pre-train the two latent spaces with the full training epochs of 500 (4 GPU^3 hours). We save the latent spaces at every 50 epochs during pre-training and choose the optimally pre-trained ones—that can be used to achieve the optimal PI performance on the test dataset of MRPC—for different PI, NLI, and STS tasks. The experimental results in Section 4 show that our pre-trained latent spaces are not overfitting and limited to MRPC. As for reproducibility, we conduct pre-training several times without setting a particular seed, and the experimental results do not fluctuate extremely (details are provided in the next subsection).

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Table 2. Dsicriminative similarities are shown for the spaces pre-trained *"with digit-count"*.

	Lp(S1) & Lp(S2)	Lp(S1) & Lp(S2)	Lnp(S1) & Lnp(S2)	Lnp(S1) & Lnp(S2)
Sentence pairs	Mean	Std.	Mean	Std.
Paraphrase	0.808	0.150	0.810 (+0.002)	0.143
Non-paraphrase	0.681	0.192	0.728 (+0.047)	0.158

Table 3. Dsicriminative similarities are shown for the spaces pre-trained "*without digit-count*".

	Lp(S1) & Lp(S2)	Lp(S1) & Lp(S2)	Lnp(S1) & Lnp(S2)	Lnp(S1) & Lnp(S2)
Sentence pairs	Mean	Std.	Mean	Std.
Paraphrase	0.817	0.139	0.808 (−0.009)	0.142
Non-paraphrase	0.711	0.167	0.730 (+0.019)	0.164

3.3 Discriminative similarity

As mentioned earlier in Section 3.1, the objective of our pre-training is to have two initial sentence-pair representations to generate the latent representations containing discriminative features. This effect is reflected by discriminative similarity, which is explained in this subsection.

As shown earlier in Figure 7, each sentence of any sentence pairs can obtain two types of latent representations based on two pre-trained latent spaces. For conciseness, paraphrase latent representation obtained for a sentence is abbreviated as Lp(S), and Lnp(S) is an abbreviation for non-paraphrase latent representation. Based on the optimally pre-trained latent spaces, we calculate the cosine similarities for Lp(S_1) & Lp(S_2) and Lnp(S_1) & Lnp(S_2), using the sentence pairs in the training dataset of MRPC. The results are shown in Tables 2 and 3.

Compared to the mean cosine similarity in the paraphrase latent space, non-paraphrase sentence pairs tend to have higher cosine similarity (≈ 0.73) in the non-paraphrase latent space, while paraphrase sentence pairs tend to remain the same or decrease marginally (≈ 0.81). We call this phenomenon "discriminative similarity." The *"with digit-count"* assumption makes the discriminative similarity comparatively more noticeable, as the mean cosine similarity of non-paraphrase sentence pairs is 0.681 in the paraphrase latent space, which is lower than 0.711 when "*without digit-count*" is the case.

To exploit the advantage of discriminative similarity for PI, the cosine similarities for Lp(S_1) & Lp(S_2) and Lnp(S_1) & Lnp(S_2) are incorporated into our feature set as the primary features. All the features are collectively presented in the next subsection.

Furthermore, we use the discriminative similarities (the examples with concrete figures are shown in the parentheses of Tables 2 and 3) to measure the reproducibility of our latent space pre-training. We additionally pre-train both types of latent spaces for 5 times, respectively, using the same hyperparameter setting mentioned in Section 3.2.4 with different random seeds. The results are shown in Figures 12 and 13. In both figures, it is consistent that non-paraphrase pairs tend to have high similarities in non-paraphrase latent space while paraphrase pairs tend to maintain the same or decrease slightly. Out of five runs of pre-training, compared to the previously pre-trained spaces whose discriminative similarities are shown in Tables 2 and 3, the results of 5th run of both types can yield nearly identical PI, NLI, or STS task performance while the others decrease marginally. The results of reproducibility test confirm that our pre-training is robust for random seeds.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

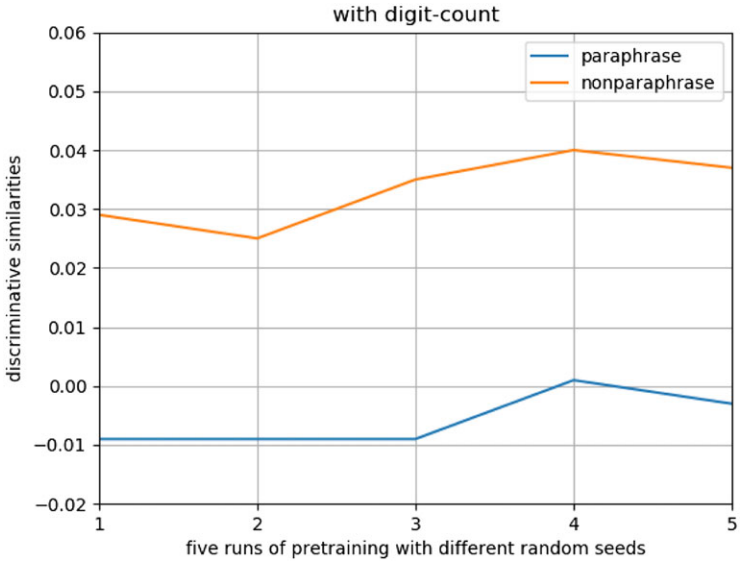

Figure 12. Reproducibility test for "*with digit-count*".

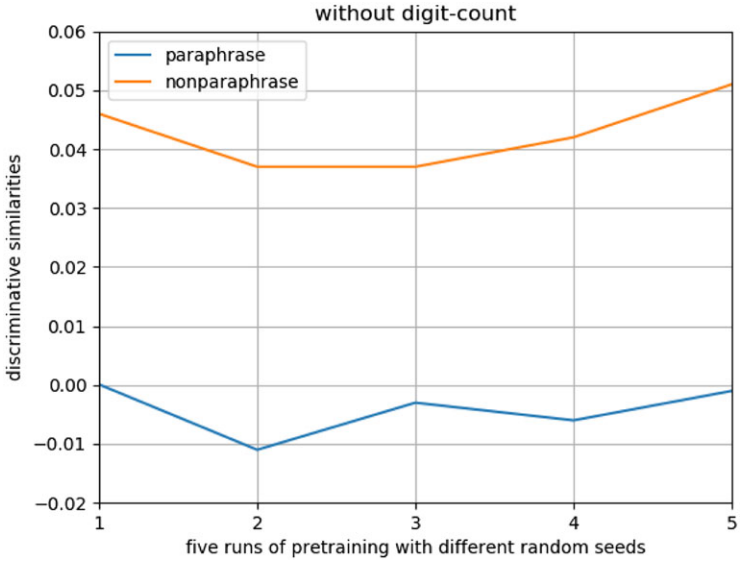

Figure 13. Reproducibility test for "*without digit-count*".

3.4 Features

Our latent space-related features (1-12) and lexical overlap features (13-19) are summarized in Table 4. The features serve as input to multiple independent classifiers for different task datasets. Inspired by Ji and Eisenstein (2013), 1-4 are concatenated as basic features. Although 9-12 are not directly related to our latent spaces, they are created along with 7-8 to augment the discriminative similarity (5-6) for every sentence pair. Besides the fine-grained n-gram overlap features (14-18) (Wan *et al.* 2006), we also enrich the granularity level by 13 (sentence level) and 19 (character level) (Popović 2015). As mentioned first in Section 2.1, both types of features are mutually

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Table 4. Latent space-related features and lexical overlap features.

Features
1 $Lp(S_1) + Lp(S_2)$
2 $\|Lp(S_1 - Lp(S_2)\|$
3 $Lnp(S_1) + Lnp(S_2)$
4 $\|Lnp(S_1) - Lnp(S_2)\|$
5 cosine similarity between $Lp(S_1)$&$Lp(S_2)$
6 cosine similarity between $Lnp(S_1)$&$Lnp(S_2)$
7 euclidean distance from $Lp(S_1)$ to $Lnp(S_1)$
8 euclidean distance from $Lp(S_2)$ to $Lnp(S_2)$
9 $\|entity\_count(S_1) - -entity\_count(S_2)\|$
10 $\|action\_count(S_1) - -action\_count(S_2)\|$
11 $\|digit\_count(S_1) - -digit\_count(S_2)\|$
12 word_mover_distance(entity_action_tokens_S_1, entity_action_tokens_S_2)
13 Levenshtein_edit_distance(S_1, S_2)
14 unigram recall/precision
15 bigram recall/precision
16 trigram recall/precision
17 BLEU recall/precision
18 absolute difference of sentence length
19 chrf recall/precision (1-6)

complementary to each other, and we delve deeper into this by performing a concrete analysis in Section 4.

4. Experimental results

The experimental results presented in this section are to verify what we claimed in the previous sections. First, to test the effectiveness of two weighted reward components, the ablation study is performed. Then we apply the pre-trained latent spaces to multiple PI, NLI, and STS benchmarks to confirm the pre-trained task-specific architecture of PEFBAT is useful for different tasks. The experimental results presented in the last subsection explain the mutual complementation mentioned in Sections 2.1 and 3.4.

4.1 Ablation study for reward components

As the implementation details introduced in Section 3.2.2, three thresholds (*sent-len* with *sent-len-diff* to assist *j-dist*) are considered as a whole package to determine whether two reward components should be weighted strongly or weakly, which is the way to represent whether a sentence pair exhibits (non)paraphrase-like characteristics. With their functionality, sentence pairs

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

that have (non)paraphrase-like characteristics tend to obtain similar latent representations in (non)paraphrase latent spaces, which is conducive to the generation of discriminative features.

To verify our claim, the results of ablation study are presented in this subsection. Note that two things listed below are invariable in the study.

(1) No matter for a single sentence or a sentence pair, three word embedding components (entity, action, and modification) convey the principal semantics in the initial sentence representation, so they are deemed as a must in our work and not changed in the study. This is validated by one good experimental result produced by the latent spaces pre-trained without the participation of two weighted reward components.

(2) The maximum number of the reward component is two, namely total dimensionality is 600 (300 each), because when the number exceeds two, the latent spaces are not trainable—the error loss hardly drops. Even though the spaces were trainable, we would not increase the number as too many reward components would depreciate the effectiveness of word-embedding components.

The models that participate in the ablation study are concluded below.

- **baseline**: our baseline is pure feature engineering: the lexical overlap features (13-19) in Table 4.
- **origin**: the two latent spaces are pre-trained based on the implementation details described in Section 3.2.
- **no j-dist**: the two latent spaces are pre-trained based on the implementation details described in Section 3.2 except that the reward component weighted by *j-dist* is removed during pre-training.
- **no sent-factor**: the two latent spaces are pre-trained based on the implementation details described in Section 3.2 except that the reward component weighted by the combination of *sent-len* and *sent-len-diff* is removed during pre-training.
- **no reward**: the two latent spaces are pre-trained based on the implementation details described in Section 3.2 except that the two weighted reward components are removed during pre-training.
- **extra factor**: the total dimensionality of two reward components is 600. So besides *j-dist*, *sent-len* and *sent-len-diff*, we include the thresholds of the lexical overlap features (13-19) in Table 4 to weight two reward components together. The 600 dimensionality is evenly distributed by the thresholds. Accordingly, this model is the two latent spaces pre-trained based on the implementation details described in Section 3.2 except that the two reward components are weighted by multiple thresholds during pre-training.

As for evaluation, we perform PI on MRPC (Dolan *et al.* 2004) using the SVM[7] with linear kernel as the classifier, and metric is accuracy and F1 score.[h] Except the baseline, the other models produce the latent space-related features (1-12), which are combined with the lexical overlap features (13-19) in Table 4 as input to the classifier. The experimental results are shown in Table 5, from which we can conclude the following three main points.

Firstly, the performance of "extra factor" model, regardless of pre-trained type, is not satisfactory. The results are within our expectation, because using lots of characteristics-based thresholds

[h]Unlike other tasks like VUA-18 (Leong, Beigman Klebanov, and Shutova., 2018) and VUA-20 (Leong *et al.* 2020) for metaphor detection, target labels are extremely unevenly distributed with 90 and 10%, where recall should also be considered as a critical metric for evaluation given similar F1 score. However, since there is no such task in our experiments, we use the two metrics only.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Table 5. The experimental results of the ablation study. The **bold** values indicate the best performance for both pre-trained types.

Models	Pre-trained type	Accuracy (%)	F1 score (%)
baseline	-	75.2	82.6
origin	*"with digit-count"*	**77.6**	**84.4**
no j-dist	*"with digit-count"*	76.8	83.9
no sent-factor	*"with digit-count"*	76.9	84.1
no reward	*"with digit-count"*	77.2	84.1
extra factor	*"with digit-count"*	76.5	83.4
origin	*"without digit-count"*	**78.2**	**84.8**
no j-dist	*"without digit-count"*	76.4	83.5
no sent-factor	*"without digit-count"*	77.3	84.0
no reward	*"without digit-count"*	77.1	84.0
extra factor	*"without digit-count"*	76.6	83.5

to weight two reward components is tantamount to pure feature engineering, as the effectiveness of three word embedding components is lessened.

Secondly, the performance of "no reward" model, although is lower than "origin" in both pre-trained types, reflects the fact that the three weighted word-embedding components are fairly robust to convey important semantics in the initial sentence-pair representations. As shown in the table, it is competitive with the other two models ("no j-dist" & "no sent-factor") containing a single weighted reward component in the initial sentence-pair representations.

Thirdly, the performance of "origin" is competitive with various task-specific DNNs.[1] While performance level is similar, our pre-trained latent spaces have an additional advantage: the fixed parameters can be used for other task datasets as well, which is parameter-efficient and demonstrated in the next subsection. In addition, two sub-points listed below need to be further explained.

(a) The pre-trained type *"with digit-count"* is the assumption we make in Section 3.2.2 that when there are too many digital numbers occurring in a sentence pair, the influence of semantics represented in the entity component should be lessened. Therefore, the threshold of *digit-count* is used to determine whether the entity component should be weighted strongly or weakly. Although the assumption is less effective for MRPC, it can indeed take effect in some other benchmarks, which is shown in the experimental results in Section 4.2.

(b) The functionality of two weighted reward components is to have sentence pairs with (non)paraphrase-like characteristics tend to obtain similar latent representations in (non)paraphrase latent spaces. As a result shown in Tables 2 and 3, non-paraphrase sentence pairs tend to have higher cosine similarity (≈ 0.73) in the non-paraphrase latent space, while paraphrase sentence pairs have the tendency to remain the same or decrease marginally (≈ 0.81). The discriminative similarity is further justified in this ablation study. As shown in Tables 6 and 7, the phenomenon is not reflected by "no reward" model: for both types of sentence pairs, they all tend to have higher cosine similarities in paraphrase latent spaces. This is the main reason why "origin" outperforms "no reward." Later in Section 4.3, we delve deeper into the utility of discriminative similarity.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Table 6. Discriminative similarity is not reflected by "no reward" model *"with digit-count"*.

	Lp(S1) & Lp(S2)	Lp(S1) & Lp(S2)	Lnp(S1) & Lnp(S2)	Lnp(S1) & Lnp(S2)
Sentence pairs	Mean	Std.	Mean	Std.
Paraphrase	0.778	0.164	0.763 (−0.015)	0.174
Non-paraphrase	0.682	0.182	0.654 (−0.028)	0.207

Table 7. Dsicriminative similarity is not reflected by "no reward" model *"without digit-count"*.

	Lp(S1) & Lp(S2)	Lp(S1) & Lp(S2)	Lnp(S1) & Lnp(S2)	Lnp(S1) & Lnp(S2)
Sentence pairs	Mean	Std.	Mean	Std.
Paraphrase	0.755	0.170	0.746 (−0.009)	0.184
Non-paraphrase	0.644	0.196	0.644 (0.000)	0.201

In addition, the latent spaces with randomly initialized parameters without pre-training achieve on-par performance with the baseline, which justifies the usefulness of pre-training. We also take a probe into the replacement; for example, we replace *j-dist* with the Levenshtein distance (feature 13 in Table 4) to weight one reward component, but the results do not outperform "origin."

4.2 Pre-trained latent spaces with manually adjustable thresholds

The experimental results shown in this subsection is to verify three of our claims: (1) the pre-trained latent spaces are not limited to MRPC (Dolan *et al.*); (2) PEFBAT can also handle NLI and STS tasks other than PI tasks; (3) PEFBAT is capable of power-efficient continual learning. We apply the pre-trained latent spaces to multiple benchmarks including six PI tasks, two NLI tasks, and two STS tasks. Besides, the task performance achieved by adapter-BERT (Houlsby *et al.* 2019b) is considered as our upper bound, because without the consideration of power efficiency, it is so far a relatively high-end implementation of parameter-efficient method. Before stepping into the details, we underline that all the tasks are tested with the fixed parameters of pre-trained latent spaces without task-specific re-training. Our open-source code is available under this link.[i]

Metrics & adjusted thresholds: We experiment with six PI tasks, two NLI and STS tasks, respectively, and the results are summarized in Table 8. For the PI tasks including MRPC (Dolan *et al.* 2004), PAN (Madnani *et al.* 2012), QQP[j] (Iyer *et al.* 2017), Twitter-URL (Lan *et al.* 2017), and PARADE (He *et al.* 2020), accuracy/F1 scores are reported; except for PAWS-wiki (Zhang *et al.* 2019), we report accuracy/AUC scores complying with the metric presented in the original paper. Accuracy scores are reported for two NLI tasks: SICK-E (Marelli *et al.* 2014) and SciTail (Khot, Sabharwal, and Clark 2018). Pearson/Spearman correlations are reported for two STS tasks: SICK-R (Marelli *et al.* 2014) and STS-B (Cer *et al.* 2017).

The "default thresholds" are namely the thresholds shown in Table 1, which are determined by the characteristics of the training dataset of MRPC (Dolan *et al.* 2004). The "adjusted thresholds" indicate that we manually adjust the thresholds to weight corresponding components according to the characteristics of different training datasets. For example, the thresholds of *sent-len-diff* and

[i] https://github.com/ryuliuxiaodong/latentspace.

[j] Due to the unavailable labels of test dataset constrained by the policy of GLUE benchmark (Wang *et al.* 2018), the task performance of QQP is reported on the development dataset.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Table 8. The experimental results scored on ten tasks. For clear illustration, the omitted unit of all accuracy metrics is %. The number below each task name is the size of training dataset. The underlined values indicate better performance than the upper bound.

	MRPC 4k	PAN 10k	QQP 363k	Twitter-URL 42.2k	PAWS-wiki 49.4k	PARADE 7.5k	SICK-R 4.4k	STS-B 5.7k	SICK-E 4.4k	SciTail 23k
Default thresholds *"with digit-count"*	77.6/84.4	93.3/93.3	81.2/74.5	88.3/69.3	74.1/78.6	74.5/71.4	82.1/76.7	72.3/72.0	85.6	79.3
Adjusted thresholds *"with digit-count"*	-	93.5/93.5	81.4/74.6	88.4/70.1	74.7/79.6	75.0/71.1	-	-	-	79.4
Default thresholds *"with digit-count"*	78.2/84.8	93.5/93.4	81.5/74.7	88.3/68.6	74.5/79.9	73.6/68.4	82.3/76.9	72.3/71.8	85.2	79.4
Adjusted thresholds *"with digit-count"*	-	93.6/93.5	81.3/75.1	88.3/69.1	74.8/80.4	73.7/68.6	-	-	-	79.8
Trainable parameters	$<$ 0.001M	$\approx$ 0.48M	$\approx$ 0.48M	$\approx$ 0.48M	$\approx$ 0.48M	$\approx$ 0.48M	$\approx$ 0.48M	$\approx$ 0.48M	$\approx$ 0.48M	$\approx$ 0.48M
Parameter update (minutes)	10	3	37	5	6	2	3	3	3	4
Upper bound (adapter-BERT)	82.5/86.7	95.4/95.4	84.7/79.6	88.2/73.9	86.7/93.6	72.7/70.9	86.2/80.4	83.2/81.4	84.8	88.6
Trainable parameters	$\approx$ 3 M	$\approx$ 3 M	$\approx$ 3 M	$\approx$ 3 M	$\approx$ 3 M	$\approx$ 3 M	$\approx$ 3 M	$\approx$ 3 M	$\approx$ 3 M	$\approx$ 3 M
Training epochs	20	10	3	10	10	20	50	50	50	3
Parameter update (minutes)	17	21	183	70	86	30	50	60	50	13

j-dist are adjusted to 8 and 0.8, respectively for PARADE (He *et al.* 2020) task. In addition, the "adjusted thresholds" only works for the task datasets labeled with binary class, including five PI tasks and one NLI task SciTail (Khot *et al.* 2018) whose sentence pairs are labeled with entails or neutral.

Classifier settings and adapter-BERT implementation: For small task datasets like MRPC containing less than 10k sentence pairs, we make attempts on both the SVM[7] with linear kernel and MLP, and report the one with better task performance. The rest of tasks with relatively big data size are tested with only MLP. Accuracy is our measurement to tune classifiers. When tuning the SVM, we use development dataset to determine hyperparameter c by means of grid search and then combine it with training dataset to train the classifier. As for MLP, we use the widely adopted hyperparameters for fine-tuning BERT (Devlin *et al.* 2019); the small difference is that our training epochs are 100 for all tasks and batch size is in {8, 16, 32}. Thus, the development dataset is directly combined with the training dataset to train the classifier. Our MLP structure for PI tasks is "432 - 900 (ReLU) - 100 (ReLU) - 2 (softmax + cross entropy)" (0.479M parameters in total).

For NLI and STS tasks, the hidden layers of MLP are the same, but output length and loss function might be different (see classifier-tuning examples in our open-source code[9]). As reflected in Yin *et al.* (2016)'s work, seven linguistic features like the number of hypernyms in a sentence pair are particularly useful for NLI tasks. We append them to our PI features and thus the input length of MLP for NLI tasks is 439. However, the seven additional linguistic features improve STS tasks marginally and can bring side effect to PI tasks, and therefore are not utilized.

Our MLP structure is uniform and heuristics-based. Although it is possible to achieve better task performance by tailoring task-specific settings—such as different layer length, dropout rate, layer normalization, etc.—for different tasks, we decide to use the simplest classifier setting to test the effectiveness of PEFBAT for a pure academic demonstration.

We use bert-for-tf2[k] implementation for adapter-BERT. In light of adapter's performance on GLUE benchmark (Wang *et al.* 2018), we choose the adapter size 64 based on BERT-base ($\approx$ 3 M trainable parameters) and follow the recommended hyperparameters presented in the original paper. For each task, we perform fine-tuning with four different training epochs {3, 10, 20, 50}, and report the one that achieves optimal performance. An exception is QQP (Iyer *et al.* 2017), because 3 epochs are sufficient to process its large size of data.

Discussion on performance: We discuss our task performance from three facets: the effect of "*with digit-count*" assumption; default and adjusted thresholds; the consumption of computational and memory resources.

The *with digit-count* model works for the task datasets that are sensitive to digits. In particular, PARADE (He *et al.* 2020) comprises computer science-related literature, and thus digits are common in its sentence pairs. As a result, the latent spaces pre-trained with the digit-count threshold contained in the dual weighting scheme are comparatively effective for this task, which validates the assumption that when there are too many digits occurring in a sentence pair, the influence of semantics represented in the entity component should be lessened. On the other hand, the assumption is not always effective. For example, PAWS (Zhang *et al.* 2019) is created to measure models' sensitivity to word order and syntactic structure. As expected, the *without digit-count* model is comparatively effective for this task. To summarize, there is no absolutely right or wrong about this assumption, because from the perspective of continual learning, tasks from customers definitely vary to some extent—some resemble PARADE while some do not.

The adjusted thresholds can improve corresponding tasks to some extent but not significantly, which reflects the fact that our latent spaces pre-trained with default thresholds are already useful for various tasks, and not only limited to MRPC (Dolan *et al.* 2004). This is further validated

[k] https://github.com/kpe/bert-for-tf2.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Table 9. The experimental results of ten tasks tuned with only 3 training epochs on adapter-BERT. The underlined values indicate decreased performance. The **bold underlined** values indicate significantly decreased performance.

	MRPC	PAN	QQP	Twitter-URL	PAWS-wiki	PARADE	SICK-R	STS-B	SICK-E	SciTail
	4k	10k	363k	42.2k	49.4k	7.5k	4.4k	5.7k	4.4k	23k
Adapter-BERT	**74.3/83.0**	94.2/94.1	84.7/79.6	87.3/72.2	**57.7/58.4**	71.0/70.4	**58.9/53.5**	**69.1/65.2**	**56.9**	88.6
Trainable parameters	≈ 3 M	≈ 3 M	≈ 3 M	≈ 3 M	≈ 3 M	≈ 3 M	≈ 3 M	≈ 3 M	≈ 3 M	≈ 3 M
Training epochs	3	3	3	3	3	3	3	3	3	3
Parameter update (minutes)	3	7	183	22	26	5	3	4	4	13

by the task performance of PARADE and SICK-E: the default thresholds based on the *with digit-count* model outperform the upper bound. As mentioned earlier in this subsection, the adjustment only works for the task datasets labeled with binary class, and we recommend it for practical use. To make the reuse of PEFBAT convenient, we have coded programming interfaces including the function of adjusting thresholds, which can be found in our open-source code.[9]

Our task performance can be categorized into three groups: the ones that outperform the upper bound (Twitter-URL and SICK-E in terms of accuracy, PARADE in terms of accuracy and F1 score); moderate difference to the upper bound (MRPC, PAN, QQP, and SICK-R); noticeable difference to the upper bound (PAWS-wiki, STS-B, and SciTail). The performance level is fairly competitive as the consumption of computational and memory resources is substantially light. As shown in Table 8, the trainable parameters are only 16% of the upper bound in the case of MLP, and the time saved for parameter update ranges from 69% (1 - 4/13) to 96% (1 - 2/50). An exception is the classifier tuned for MRPC, but this case is based on the SVM classifier, which consumes CPU resource not as expensively as GPU computation (Strubell *et al.* 2019). It is obvious that tuning adapter-BERT (Houlsby *et al.* 2019b) on the small tasks is more expensive than fine-tuning original BERT (Devlin *et al.* 2019). This is mainly because of the requirement of large training epochs: 20 epochs for MRPC and PARADE; 50 epochs for SICK-E, SICK-R, and STS-B. It can be argued that tuning adapter-BERT with only 3 training epochs for each task can consume less computing time and outperforms PEFBAT. To confirm if this is the case, we perform additional experiments, and the results are presented in Table 9. Except for QQP and SciTail whose previous epochs are 3, only the performance of three tasks (PAN, Twitter-URL, and PARADE) decreases marginally, but the rest drops significantly. Besides, although corresponding computing time indeed decreases a lot, it is still above ours. With the advantage of slight consumption of computational and memory resources for each task, PEFBAT is capable of power-efficient continual learning, especially when tasks arriving from customers increase exponentially.

4.3 Analysis of discriminative similarity

Two experimental results are quite surprising to us. Firstly, PARADE (He *et al.* 2020) is so far the most difficult PI task for the BERT-family, according to the evidence provided in the original paper. However, PEFBAT (75.0/71.1) outperforms not only adapter-BERT (72.7/70.9) (Houlsby *et al.* 2019a) but also BERT-large (73.6/70.9) (Devlin *et al.* 2019) and SciBERT (74.1/72.3) (Beltagy, Lo, and Cohan 2019) in terms of accuracy. Secondly, PAWS (Zhang *et al.* 2019) is created to measure models' sensitivity to word order and syntactic structure. Supposedly, our performance level should not be satisfactory, but when compared to the experimental results reported in the

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

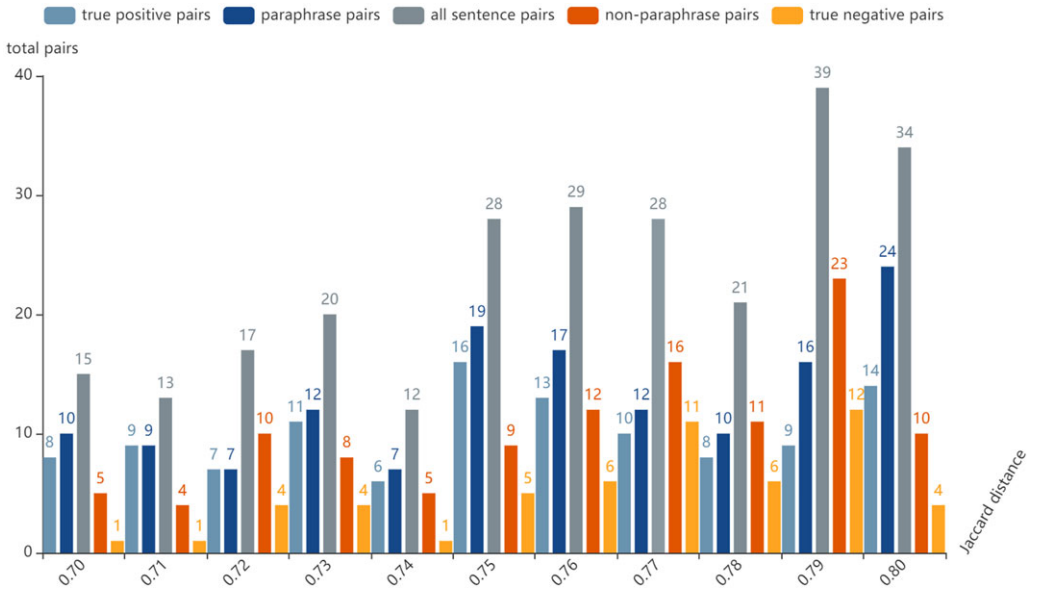

Figure 14. Sentence-pair distributions (predicted test dataset of PARADE) at each level of Jaccard distance from 0.70 to 0.80.

original paper, PEFBAT outperforms the baseline BOW (55.8/41.1) significantly and is better than most task-specific DNNs without pre-training using PAWS unlabeled corpus.

To explore the rationale behind the performance, we perform statistical analysis to examine the effectiveness of our discriminative similarity, namely the features 5-6 in Table 4. At the same level of Jaccard distance between sentence pairs, lexical overlap features are supposedly hard to differentiate between paraphrase pairs and non-paraphrase pairs, then to what degree does our discriminative similarity enhance the ability of differentiation? The visualization to illustrate the analysis is presented in Figures 14, 15, and 16.

The analysis is performed during the process of evaluating the test dataset of PARADE (based on *with digit-count* adjusted thresholds). At 0.78 of Jaccard distance shown in Figure 14, we discover that paraphrase and non-paraphrase have nearly identical number of sentence pairs (10 and 11, respectively), and the accuracy of PI is 67% (see Figure 15) surpassing 52% if twenty-one sentence pairs are all guessed as true negative. Then the cosine similarities of those sentence pairs in paraphrase and non-paraphrase latent spaces are presented in Figure 16 corresponding to the features 5-6 in Table 4. In Figure 16, a hypothetical boundary line is drawn, which can have the two features achieve accuracy of 67% already with 9 true positive pairs and 5 true negative pairs, although the real numbers of true positive and negative pairs are 8 and 6, respectively, as shown in Figure 14. The combination of 8 and 6 is better than 9 and 5 as the numbers of paraphrase and non-paraphrase sentence pairs are 10 and 11, respectively, at 0.78 of Jaccard distance. We believe that it is our design of the features 7-12 (explained in Section 3.4) that augments the features 5-6.

To respond to what we mentioned in Sections 2.1 and 3.4, our latent space-related features are crucial when lexical overlaps are indistinguishable between paraphrase and non-paraphrase sentence pairs, while lexical overlap features are the strong basis when dealing with task datasets like PAN (Madnani *et al.* 2012) (see Figure 3). Both types of features are mutually complementary to each other.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

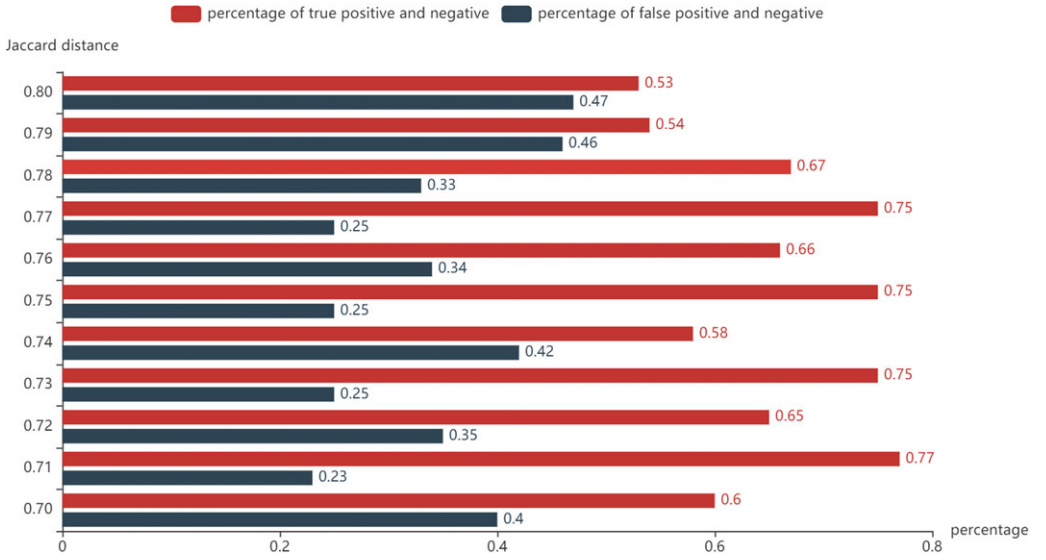

Figure 15. Accuracies of paraphrase identification (predicted test dataset of PARADE) at each level of Jaccard distance from 0.70 to 0.80.

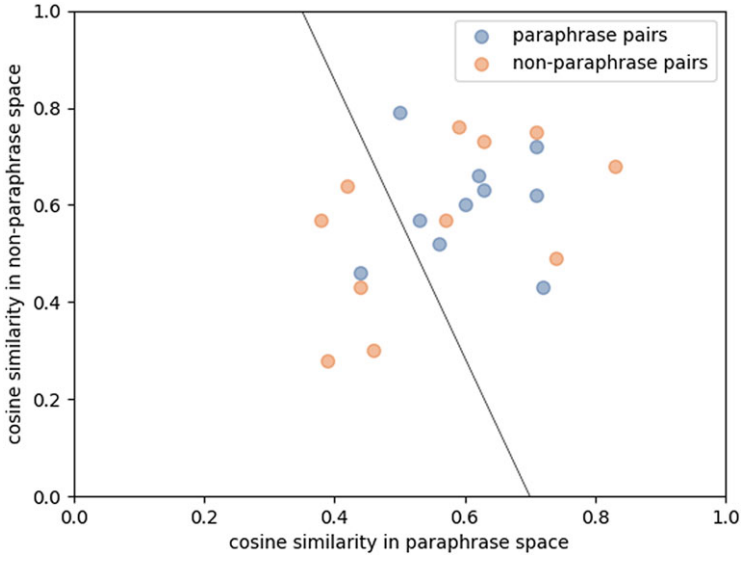

Figure 16. Cosine similarities in (non)paraphrase space for sentence pairs (PARADE test dataset) at Jaccard distance 0.78.

5. Conclusion and future work

In this article, for PI task, we proposed a new method (PEFBAT) to improve the usage of parameters efficiently for feature-based transfer. Our motivation, research goal, and implementation details were described in Sections 1 and 3, respectively. PEFBAT can also handle NLI and STS tasks, and its essence is a pre-trained task-specific architecture, the fixed parameters of which can be shared by multiple classifiers with small additional parameters. As a result, for each task, the computational cost left involving parameter update is only generated from classifier-tuning: the

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

features output from the architecture combined with lexical overlap features are fed into a single classifier for tuning. Such technical novelty can lead to slight consumption of computational and memory resources for each task and is also capable of power-efficient continual learning. In Section 4, we experimented with multiple benchmarks, and the results showed that PEFBAT is competitive with adapter-BERT over some tasks while consuming only 16% trainable parameters and saving 69-96% time for parameter update. We also performed the ablation study and technical analysis to help further understand the mechanism of PEFBAT.

One of our future work is to apply PEFBAT to low-resource natural languages with limited labeled and unlabeled data (Hedderich *et al.* 2021). Given any research project, we will put this idea into practice, as there are three main reasons listed below that PEFBAT is implementable to low-resource settings.

(1) Nowadays, what we can benefit from transfer learning mechanism becomes a common sense: transferring the parameters of a network trained on large corpora to the related problems with little data. However, we have managed to leverage the mechanism in a reverse fashion. Our task-specific architecture (two latent spaces) is pre-trained with the training dataset of MRPC (Dolan *et al.* 2004), which is the smallest English paraphrase corpus. Nevertheless, the fixed parameters of pre-trained latent spaces can be used by other task datasets as well, which is beneficial to the scenario of limited labeled data.

(2) It is shown in Mohiuddin and Joty (2020)'s work that fastText (Mikolov *et al.* 2018) is implementable to low-resource natural languages. The rationale is obvious as fastText is not deep learning architecture-based, which is beneficial to the scenario of limited unlabeled data.

(3) POS-tags provided by the NLTK[6] can be implemented using HMMs (Hidden Markov Models),[1] a probabilistic approach to assigning tags. Since the approach is also not deep learning architecture-based, there is no strict requirement of large labeled data for training parameters.

We also consider another future work by applying PEFBAT to cloud environment (Houlsby *et al.* 2019b). As PEFBAT is capable of power-efficient continual learning, and thus given enough research funds, whether it can also provide the same power-efficient manner for cloud environment is an interesting topic for us. We plan to launch an AWS EC2 instance to deploy PEFBAT, and then provide classifier interfaces for customers. Suppose the MLP classifiers that customers use have the same size as we use in this article (0.479M parameters per classifier), then each classifier accounts only for approximately 3.7 MB memory given that one parameter needs 8-byte memory. We hold a positive attitude towards this future work, as when we perform simulation by running 20 classifiers in parallel on our local device,[3] parameter update is as fast as we run them individually. Our goal is to guarantee at least 1000 classifiers can run in parallel given a powerful EC2 instance, as tasks arriving from customers can be simultaneous.

References

Beltagy I., **Lo K.** and **Cohan A.** (2019). SciBERT: a pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics, pp. 3615–3620.

Bengio Y., **Louradour J.**, **Collobert R.** and **Weston J.** (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning. ICML'09.* New York, NY, USA: Association for Computing Machinery, pp. 41–48.

[1]http://www.nltk.org/api/nltk.tag.html?highlight=hmm.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Bromley J., **Bentz J.W.**, **Bottou L.**, **Guyon I.**, **LeCun Y.**, **Moore C.**, **Säckinger E.** and **Shah R.** (1993). Signature verification using a "siamese"' time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence* **7**(04), 669–688.

Caruana R. (1998). *Multitask Learning*. Boston, MA: Springer US, pp. 95–133.

Cer D., **Diab M.**, **Agirre E.**, **Lopez-Gazpio I.** and **Specia L.** (2017). SemEval-2017 task 1: semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada. Association for Computational Linguistics, pp. 1–14.

Chandrasekaran D. and **Mago V.** (2020). Evolution of semantic similarity - A survey. CoRR, abs/2004.13820.

Chen T., **Goodfellow I.** and **Shlens J.** (2016). Net2net: accelerating learning via knowledge transfer.

Conneau A. and **Kiela D.** (2018). SentEval: an evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Corbeil J.-P. and **Abdi Ghavidel H.** (2021). Assessing the eligibility of backtranslated samples based on semantic similarity for the paraphrase identification task. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pp. 301–308, Held Online. INCOMA Ltd.

Das D. and **Smith N.A.** (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore. Association for Computational Linguistics, pp. 468–476.

de Masson d'Autume C., **Ruder S.**, **Kong L.** and **Yogatama D.** (2019). Episodic memory in lifelong language learning. In **Wallach H.**, **Larochelle H.**, **Beygelzimer A.**, **d'Alché-Buc F.**, **Fox E.** and **Garnett R.** (eds), *Advances in Neural Information Processing Systems*, vol. **32**, Curran Associates, Inc.

de Wynter A. and **Perry D.J.** (2020). Optimal subarchitecture extraction for BERT. *CoRR*. https://arxiv.org/abs/2010.10499

Deerwester S., **Dumais S.T.**, **Furnas G.W.**, **Landauer T.K.** and **Harshman R.** (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science* **41**(6), 391–407.

Deng J., **Dong W.**, **Socher R.**, **Li L.-J.**, **Li K.** and **Fei-Fei L.** (2009). Imagenet: a large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 248–255.

Devlin J., **Chang M.-W.**, **Lee K.** and **Toutanova K.** (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 4171–4186.

Dolan B., **Quirk C.** and **Brockett C.** (2004). Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland. COLING, pp. 350–356.

Dong D., **Wu H.**, **He W.**, **Yu D.** and **Wang H.** (2015). Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China. Association for Computational Linguistics, pp. 1723–1732.

Faruqui M., **Dodge J.**, **Jauhar S.K.**, **Dyer C.**, **Hovy E.** and **Smith N.A.** (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado. Association for Computational Linguistics, pp. 1606–1615.

Fellbaum C. (ed.) (1998). *WordNet: An Electronic Lexical Database*, Language, Speech, and Communication. Cambridge, MA: MIT Press.

Fernando S. and **Stevenson M.** (2008). A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pp. 45–52.

French R.M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* **3**(4), 128–135.

Gammerman A., **Vovk V.** and **Vapnik V.** (1998). Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc, pp. 148–155.

Glavaš G. and **Vulić I.** (2018). Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics, pp. 34–45.

Guo W. and **Diab M.** (2012). Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-volume 1*. Association for Computational Linguistics, pp. 864–872.

He H., **Gimpel K.** and **Lin J.** (2015). Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal. Association for Computational Linguistics, pp. 1576–1586.

He Y., **Wang Z.**, **Zhang Y.**, **Huang R.** and **Caverlee J.** (2020). PARADE: a new dataset for paraphrase identification requiring computer science domain knowledge. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online. Association for Computational Linguistics, pp. 7572–7582.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Hedderich M.A., **Lange L.**, **Adel H.**, **Strötgen J.** and **Klakow D.** (2021). A survey on recent approaches for natural language processing in low-resource scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online. Association for Computational Linguistics, pp. 2545–2568.

Houlsby N., **Giurgiu A.**, **Jastrzebski S.**, **Morrone B.**, **De Laroussilhe Q.**, **Gesmundo A.**, **Attariyan M.** and **Gelly S.** (2019a). Parameter-efficient transfer learning for NLP. In **Chaudhuri K.** and **Salakhutdinov R.** (eds), *Proceedings of the 36th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, vol. **97.** PMLR, pp. 2790–2799.

Houlsby N., **Giurgiu A.**, **Jastrzebski S.**, **Morrone B.**, **de Laroussilhe Q.**, **Gesmundo A.**, **Attariyan M.** and **Gelly S.** (2019b). Parameter-efficient transfer learning for nlp.

Howard J. and **Ruder S.** (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics, pp. 328–339.

Hung C.-Y., **Tu C.-H.**, **Wu C.-E.**, **Chen C.-H.**, **Chan Y.-M.** and **Chen C.-S.** (2019). Compacting, picking and growing for unforgetting continual learning. In **Wallach H.**, **Larochelle H.**, **Beygelzimer A.**, **d'Alché-Buc F.**, **Fox E.** and **Garnett R.** (eds), *Advances in Neural Information Processing Systems*, vol. **32.** Curran Associates, Inc.

Iyer S., **Dandekar N.** and **Csernai K.** (2017). First Quora Dataset Release: Question Pairs. *Available at:* https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs.

Jaccard P. (1912). The distribution of the flora in the alpine zone. 1. *New Phytologist* **11**(2), 37–50.

Ji Y. and **Eisenstein J.** (2013). Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 891–896.

Kadotani S., **Kajiwara T.**, **Arase Y.** and **Onizuka M.** (2021). Edit distance based curriculum learning for paraphrase generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*. Online. Association for Computational Linguistics, pp. 229–234.

Khot T., **Sabharwal A.** and **Clark P.** (2018). SciTail: a textual entailment dataset from science question answering. In *AAAI*.

Kirkpatrick J., **Pascanu R.**, **Rabinowitz N.**, **Veness J.**, **Desjardins G.**, **Rusu A.A.**, **Milan K.**, **Quan J.**, **Ramalho T.**, **Grabska-Barwinska A.**, **Hassabis D.**, **Clopath C.**, **Kumaran D.** and **Hadsell R.** (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of The National Academy of Sciences of The United States of America* **114**(13), 3521–3526.

Lan W., **Qiu S.**, **He H.** and **Xu W.** (2017). A continuously growing dataset of sentential paraphrases. In *Proceedings of The 2017 Conference on Empirical Methods on Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 1235–1245.

Lan Z., **Chen M.**, **Goodman S.**, **Gimpel K.**, **Sharma P.** and **Soricut R.** (2019). ALBERT: a lite BERT for self-supervised learning of language representations. CoRR, abs/1909.11942.

Leong C.W.B., **Beigman Klebanov B.**, **Hamill C.**, **Stemle E.**, **Ubale R.** and **Chen X.** (2020). A report on the 2020 VUA and TOEFL metaphor detection shared task. In *Proceedings of the Second Workshop on Figurative Language Processing*. Online. Association for Computational Linguistics, pp. 18–29.

Leong C.W.B., **Beigman Klebanov B.** and **Shutova E.** (2018). A report on the 2018 VUA metaphor detection shared task. In *Proceedings of the Workshop on Figurative Language Processing*, New Orleans, Louisiana. Association for Computational Linguistics, pp. 56–66.

Liu X., **Zheng Y.**, **Du Z.**, **Ding M.**, **Qian Y.**, **Yang Z.** and **Tang J.** (2021). Gpt understands, too.

Lopez-Paz D. and **Ranzato M.A.** (2017). Gradient episodic memory for continual learning. In **Guyon I.**, **Luxburg U.V.**, **Bengio S.**, **Wallach H.**, **Fergus R.**, **Vishwanathan S.** and **Garnett R.** (eds), *Advances in Neural Information Processing Systems*, vol. **30.** Curran Associates, Inc.

Madnani N., **Tetreault J.** and **Chodorow M.** (2012). Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Montréal, Canada. Association for Computational Linguistics, pp. 182–190.

Marelli M., **Bentivogli L.**, **Baroni M.**, **Bernardi R.**, **Menini S.** and **Zamparelli R.** (2014). SemEval-2014 task 1: evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland. Association for Computational Linguistics, pp. 1–8.

McCann B., **Keskar N.S.**, **Xiong C.** and **Socher R.** (2018). The natural language decathlon: multitask learning as question answering. CoRR, abs/1806.08730.

McCloskey M. and **Cohen N. J.** (1989). Catastrophic interference in connectionist networks: the sequential learning problem. *Psychology of Learning and Motivation* **24**, 109–165, Academic Press.

Meng Y., **Ao X.**, **He Q.**, **Sun X.**, **Han Q.**, **Wu F.**, **Fan C.** and **Li J.** (2021). ConRPG: paraphrase generation using contexts as regularizer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics, pp. 2551–2562.

Mikolov T., **Grave E.**, **Bojanowski P.**, **Puhrsch C.** and **Joulin A.** (2018). Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Mikolov T., **Sutskever I.**, **Chen K.**, **Corrado G. S.** and **Dean J.** (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Mohiuddin T. and **Joty S.** (2020). Unsupervised word translation with adversarial autoencoder. *Computational Linguistics* **46**(2), 257–288.

Nighojkar A. and **Licato J.** (2021). Improving paraphrase detection with the adversarial paraphrasing task. CoRR, abs/2106.07691.

Pennington J., **Socher R.** and **Manning C.** (2014). GloVe: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar. Association for Computational Linguistics, pp. 1532–1543.

Peters M., **Neumann M.**, **Iyyer M.**, **Gardner M.**, **Clark C.**, **Lee K.** and **Zettlemoyer L.** (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana. Association for Computational Linguistics, pp. 2227–2237.

Popović M. (2015). chrF: character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal. Association for Computational Linguistics, pp. 392–395.

Radford A., **Wu J.**, **Child R.**, **Luan D.**, **Amodei D.** and **Sutskever I.** (2019). Language models are unsupervised multitask learners.

Reimers N. and **Gurevych I.** (2019). Sentence-BERT: sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics, pp. 3982–3992.

Rusu A.A., **Rabinowitz N.C.**, **Desjardins G.**, **Soyer H.**, **Kirkpatrick J.**, **Kavukcuoglu K.**, **Pascanu R.** and **Hadsell R.** (2016). Progressive neural networks.

Sanh V., **Debut L.**, **Chaumond J.** and **Wolf T.** (2019). Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR, abs/1910.01108.

Sellam T., **Das D.** and **Parikh A.P.** (2020, BLEURT: learning robust metrics for text generation, CoRR, abs/2004.04696.

Shi W., **Chen M.**, **Zhou P.** and **Chang K.-W.** (2019). Retrofitting contextualized word embeddings with paraphrases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics, pp. 1198–1203.

Socher R., **Huang E.H.**, **Pennin J.**, **Manning C.D.** and **Ng A.Y.** (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pp. 801–809.

Strubell E., **Ganesh A.** and **McCallum A.** (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics, pp. 3645–3650.

Sun F.-K., **Ho C.-H.** and **Lee H.-Y.** (2020). LAMOL: LAnguage MOdeling for lifelong language learning. In *International Conference on Learning Representations*.

Thrun S. (1998). *Lifelong Learning Algorithms*. Boston, MA: Springer US, pp. 181–209.

van de Ven G. M. and **Tolias A. S.** (2019). Three scenarios for continual learning.

Vaswani A., **Shazeer N.**, **Parmar N.**, **Uszkoreit J.**, **Jones L.**, **Gomez A.N.**, **Kaiser Ł.** and **Polosukhin I.** (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Wan S., **Dras M.**, **Dale R.** and **Paris C.** (2006). Using dependency-based features to take the'para-farce'out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop 2006*, pp. 131–138.

Wang A., **Singh A.**, **Michael J.**, **Hill F.**, **Levy O.** and **Bowman S.** (2018). GLUE: a multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium. Association for Computational Linguistics, pp. 353–355.

Weston J., **Bordes A.**, **Chopra S.**, **Rush A. M.**, **van Merriënboer B.**, **Joulin A.** and **Mikolov T.** (2015). Towards ai-complete question answering: a set of prerequisite toy tasks.

Yang Z., **Dai Z.**, **Yang Y.**, **Carbonell J.**, **Salakhutdinov R.R.** and **Le Q.V.** (2019). Xlnet: generalized autoregressive pretraining for language understanding. In **Wallach H.**, **Larochelle H.**, **Beygelzimer A.**, **d'Alché-Buc F.**, **Fox E.** and **Garnett R.** (eds), *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc, pp. 5753–5763.

Yin W. and **Schütze H.** (2015a). Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado. Association for Computational Linguistics, pp. 901–911.

Yin W. and **Schütze H.** (2015b). Discriminative phrase embedding for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado. Association for Computational Linguistics, pp. 1368–1373.

Yin W., **Schütze H.**, **Xiang B.** and **Zhou B.** (2016). ABCNN: attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* **4**, 259–272.

https://doi.org/10.1017/S135132492200050X Published online by Cambridge University Press

Yosinski J., **Clune J.**, **Bengio Y.** and **Lipson H.** (2014). How transferable are features in deep neural networks?. In **Ghahramani Z.**, **Welling M.**, **Cortes C.**, **Lawrence N.** and **Weinberger K. Q.** (eds), *Advances in Neural Information Processing Systems*, vol. **27**, Curran Associates, Inc.

Yu Z., **Cohen T.**, **Wallace B.**, **Bernstam E.** and **Johnson T.** (2016). Retrofitting word vectors of MeSH terms to improve semantic similarity measures. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, Auxtin, TX. Association for Computational Linguistics, pp. 43–51.

Zamir A.R., **Sax A.**, **Shen W.**, **Guibas L.J.**, **Malik J.** and **Savarese S.** (2018). Taskonomy: disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang Y., **Baldridge J.** and **He L.** (2019). PAWS: paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 1298–1308.

Cite this article: Liu X, Rzepka R and Araki K (2023). Parameter-efficient feature-based transfer for paraphrase identification. *Natural Language Engineering* **29**, 1066–1096. https://doi.org/10.1017/S135132492200050X