



HOKKAIDO UNIVERSITY

Title	Study on Fusion of Distributional Representations and Distributed Representations
Author(s)	劉, 曉冬
Degree Grantor	北海道大学
Degree Name	博士(情報科学)
Dissertation Number	甲第16176号
Issue Date	2024-09-25
DOI	https://doi.org/10.14943/doctoral.k16176
Doc URL	https://hdl.handle.net/2115/93520
Type	doctoral thesis
File Information	Liu_Xiaodong.pdf



Study on Fusion of Distributional Representations and Distributed Representations



Xiaodong Liu

Language Media Laboratory
Media Network Division
Graduate School of Information Science and Technology
Hokkaido University

This dissertation is submitted for the doctoral degree of
Information Science

Graduate School of Information
Science and Technology

June 2024

Abstract

Nowadays, connectionism is the mainstream theory that prevails in Artificial Intelligence (AI) domain. The theory is implemented by deep learning of neural networks and big data. Accompanied by the enhancement of computational resources, with any well-designed training objective, a neural network with billions of parameters is trainable for addressing tasks in the sub-domains of Computer Vision (CV), Audio Signal Processing (ASP), and Natural Language Processing (NLP). In the NLP domain that I am dedicated to, the transformer-based architecture wields a significant impact on the existing powerful language models such as auto-encoding models like BERT and autoregressive models like GPT. These neural models are well known as Large Language Models (LLMs), influencing not only academic research directions, but also our daily life.

The semantic representations output from LLMs are distributed representations, which are the same definition for non-transformer-based neural embedding models like Word2Vec, ELMo, and so on. Distributed representations are dense vector representations (commonly referred to as embeddings), where each dimension of all vectors is the semantic axis distributed with numerical values for different words or tokens. These pieces of semantic information are distributed across the entire vector for each single word or token.

On the other hand, distributional representations are sparse count vectors, where the frequency of occurrence of each word in any given document or sentence is used as a feature. The implementation of distributional semantics is known as Vector Space Models (VSMs). The classical one is term-document matrix, where term indicates all individual words comprising different documents. VSMs are commonly used for information retrieval and document classification via the measurement of semantic similarity using frequency distribution in the term-document matrix.

There are two common grounds between distributed representations and distributional representations. The first is that distributional representations are somewhat equivalent to distributed representations once the term-document matrix of the former is factorized into dense vector representations. The second is the distributional hypothesis — you shall know a word by the company it keeps. This hypothesis has contributed to some effective training objectives for contemporary neural models in NLP. For example, the hypothesis is adopted by

the Masked Language Model (MLM) pre-training objective of BERT and the Continuous Bag of Words (CBOW) pre-training objective of word embeddings like Word2Vec. Meanwhile, distributional representations are also based on the hypothesis. The similarity of semantics can be measured by cosine similarity of distributional representations in their vector space; e.g., words having similar document distributions tend to achieve high cosine similarity.

During the period of my PhD study, my research focus is to investigate whether the fusion of distributional semantics and distributed semantics can contribute to NLP development in terms of real-word applications and semantic representations. I come up with two fusion approaches: mapping and structural connection.

Regarding the mapping approach, I propose a novel method by mapping weighed distributed representations to distributional representations via two trainable projection matrices. The trained matrices are frozen during inference. As a result, any sentence pair from different task datasets can obtain their distributional semantic representations via the projection from distributed representations, and the obtained representations are used as input to different classifiers for paraphrase identification or other NLP sentence-pair tasks. In this case, only the parameters of classifiers need to be optimized for different tasks. This approach contributes to power-efficient continual learning of sentence-pair tasks. I write a journal article for this approach titled “Parameter-efficient feature-based transfer for paraphrase identification”, which is accepted by Natural Language Engineering journal.

Regarding the approach of structural connection, I propose DG embeddings also derived from distributional representations, which are created based on the inspiration of a real-world scenario called headword search. DG embeddings are used to gloss bidirectional context words for target-word predictions in the MLM pre-training objective of BERT. As a result, better acquisition of vocabulary is attainable for BERT after pre-training. Also, this work leads to a computational inexpensive future expansion owing to the above-mentioned property of distributional semantics. I write a journal article for this approach titled “DG Embeddings: The unsupervised definition embeddings learned from dictionary and glossary to gloss context words of Cloze task”, which is accepted by Knowledge-Based Systems journal.

Table of contents

List of figures	vii
List of tables	xi
1 Introduction	1
1.1 Distributed Representations	1
1.2 Distributional Representations	4
1.3 Common Grounds in Both Representations	5
1.4 Motivation & Research Goal	7
1.5 Dissertation Structure	7
2 Related Works	9
2.1 Textual Similarities for Paraphrase Identification	9
2.2 Task-specific DNNs for Paraphrase Identification	10
2.3 Transfer Learning Approaches for Paraphrase Identification	11
2.4 Continual Learning	12
2.5 Target Words-glossed Pre-training for Language Models	13
2.6 Knowledge Infusing Before the Era of Language Models	14
3 Representation Fusion Through Mapping	17
3.1 Introduction	17
3.1.1 Technical Background & Research Goal	18
3.1.2 Practical Use in Real-World Scenario	21
3.2 Proposed Method	21
3.2.1 Initial Sentence-Pair Representations & Objective of Pre-training	21
3.2.2 Latent Space Pre-training	22
3.2.3 Discriminative Similarity	29
3.2.4 Features	30
3.3 Experimental Results	31

3.3.1	Ablation Study for Reward Components	31
3.3.2	Pre-trained Latent Spaces with Manually Adjustable Thresholds . .	34
3.3.3	Analysis of Discriminative Similarity	38
3.4	Conclusion	40
4	Representation Fusion Through Structural Connection	43
4.1	Introduction	43
4.1.1	Context Words-Glossed Pre-training	44
4.1.2	Unsupervised Definition Embeddings	47
4.1.3	Research Goals & Contributions	48
4.2	DG Embeddings	49
4.2.1	Extraction of Key Features	50
4.2.2	Matrix Factorization & Adaption	51
4.2.3	Evaluation of DG Embeddings	52
4.3	DG-BERT	56
4.3.1	Pre-training & Experimental Settings	56
4.3.2	Pre-training Convergence	57
4.3.3	Preliminary Tests	59
4.3.4	Standard Downstream Tasks	60
4.3.5	Detection of Figurative Semantics	61
4.3.6	Investigation on Collocations of Phrases	62
4.4	Further Explanation on Methodological Advantage	66
4.5	Discussion	67
4.5.1	Further Insight into Research Meaning	68
4.5.2	Limitation	68
4.6	Conclusion	70
5	Conclusion & Future Directions	73
5.1	Conclusion	73
5.2	Future Directions	74
	References	81
	Appendix A Supplementary Information to Chapter 4	97
A.1	Hyperparameters	97
A.2	Psycholinguistic Diagnostics	97

List of figures

1.1	Perceptron [128] with a single neuron cannot address XOR (exclusive or) classification using a linear regression line.	1
1.2	(Non-)Transformer-based auto-encoding and autoregressive models. The visualizations are from Transformer [150], BERT [31] and Word2Vec [106] papers.	2
1.3	Distributed representations are exemplified by word embeddings.	3
1.4	An example is illustrated for the term-document matrix.	4
1.5	Distributed representations of word embeddings are somewhat equivalent to distributional representations after matrix factorization. The visualization is from Yoav Goldberg’s presentation for their paper [88].	5
1.6	An example is shown for the MLM pre-training objective of BERT.	6
2.1	PAN.	9
2.2	PAWS-wiki.	9
3.1	Parameters that need to be updated for each PI, NLI, or STS task. <u>The underlined</u> denotes my work.	18
3.2	Parameter-efficient feature-based transfer for power-efficient continual learning.	20
3.3	Modeling of initial sentence-pair representations.	22
3.4	Mapping mechanism of pre-training.	23
3.5	During inference, each sentence of any sentence pairs can obtain two types of latent representations.	24
3.6	<i>sentence length</i>	27
3.7	<i>absolute difference of sentence length</i>	27
3.8	<i>Jaccard distance</i>	27
3.9	<i>digit-count</i>	27
3.10	Reproducibility test for ‘ <i>with digit-count</i> ’.	29
3.11	Reproducibility test for ‘ <i>without digit-count</i> ’.	29

3.12	Sentence-pair distributions (predicted test dataset of PARADE) at each level of Jaccard distance from 0.70 to 0.80.	38
3.13	Accuracies of paraphrase identification (predicted test dataset of PARADE) at each level of Jaccard distance from 0.70 to 0.80.	39
3.14	Cosine similarities in (non)paraphrase space for sentence pairs (PARADE test dataset) at Jaccard distance 0.78.	39
4.1	The approximation to the meaning of “jubilant” hinges on whether contextual clues are clear or not. If not, dictionaries are used to gloss unclear parts. . .	44
4.2	The embedding layer of BERT is shown for my proposed approach.	45
4.3	Let machines learn how to use the senses represented in DG embeddings. . .	46
4.4	The flow of definition pre-processsing is exemplified by the first entry “subsist” and its semantically related entry “frugality”.	49
4.5	SVD visualization of embeddings from left to right → 1st : DG embeddings; 2nd : word embeddings of Huggingface BERT-base-uncased; 3rd : comparison between 1st and 2nd . In 3rd , the magnitude of DG embeddings is scaled down to a similar magnitude of the word embeddings.	56
4.6	Left : Every 500-step batch loss is shown for the first epoch; Middle : Every 500-step batch loss is shown for the second epoch.; Right : Every 500-step batch loss is shown for the third epoch.	58
4.7	Left : Every 500-step batch loss is shown for the fourth epoch; Middle : The convergence is shown for the first 4 epochs contained in the first 100K pre-training steps. Four loss values are the average of all 500-step batch losses in each epoch; Right : The convergence is shown for every 100K pre-training steps. Ten loss values are the average of all 500-step batch losses in each 100K step.	58
4.8	From left to right → 1st : DG-BERT; 2nd : Vanilla BERT; 3rd : DG-BERT; 4th : Vanilla BERT	63
4.9	From left to right → 1st : DG-BERT; 2nd : Vanilla BERT; 3rd : DG-BERT .	63
4.10	DG, scalar and bias embeddings are removed from pre-trained DG-BERT. .	63
4.11	SVD visualization of token distribution of word embeddings from left to right → 1st : word embeddings of my Vanilla BERT; 2nd : word embeddings of DG-BERT; 3rd : comparison between 1st and 2nd	67

4.12	The convergence speed of preliminary pre-training (every 500-step batch loss in first 10k steps). This is performed to consider if full pre-training is worthy of pursuit. Compared to full pre-training, the setting is scaled down in terms of batch size 32, max-seq-length 128, approximately 20k input sequences, which can run on a single GPU device.	69
5.1	The algorithm of DDPM [63] for text-to-image generators and the generated images are from https://www.imagine.art/dashboard/tool/from-text with the given prompts.	74

List of tables

3.1	Dual weighting scheme.	26
3.2	Discriminative similarities are shown for the spaces pre-trained ‘ <i>with digit-count</i> ’.	28
3.3	Discriminative similarities are shown for the spaces pre-trained ‘ <i>without digit-count</i> ’.	29
3.4	Latent space-related features and lexical overlap features.	30
3.5	The experimental results of the ablation study. The bold values indicate the best performance for both pre-trained types.	32
3.6	Discriminative similarity is not reflected by “no reward” model ‘ <i>with digit-count</i> ’.	32
3.7	Discriminative similarity is not reflected by “no reward” model ‘ <i>without digit-count</i> ’.	33
3.8	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 <u>underlined</u> values indicate better performance than the upper bound.	35
3.9	The experimental results of ten tasks tuned with only 3 training epochs on adapter-BERT. The <u>underlined</u> values indicate decreased performance. The <u>bold underlined</u> values indicate significantly decreased performance. . . .	37
4.1	Three types of embeddings are evaluated at the word and phrase levels. “BERT word embeddings” is the word embeddings from pre-trained Hugging-Face uncased BERT-base [160]. The weight of “Random word embeddings” is initialized by normal distribution with mean 0 and standard deviation 0.02. . . .	54

4.2	The percentage in each interval of internal similarities are scored by three types of embeddings. “BERT word embeddings” is the word embeddings from pre-trained HuggingFace uncased BERT-base [160]. The weight of “Random word embeddings” is initialized by normal distribution with mean 0 and standard deviation 0.02.	55
4.3	A preliminary test is performed for target-word predictions in collocations of two common idiomatic expressions. “DG-BERT w/o DG” denotes the whole package of DG, scalar and bias embeddings is removed from pre-trained DG-BERT.	59
4.4	The experimental results of GLUE benchmark are reported on validation set . The result for MNLI is reported on the average of match and mismatch.	60
4.5	The experimental results are reported on validation set for SQuAD, CoNLL2003, and SWAG tasks. The results of CoNLL2003 are based on label-first-token.	60
4.6	The experimental results are reported on validation set for two paraphrase identification tasks.	60
4.7	The experimental results are reported on test set for four metaphor detection tasks.	61
4.8	The first ten sentences containing idiomatic expressions with confidence score 1.0 are selected from MAGPIE [56] for target-word predictions in collocation of phrases. “DG-BERT w/o DG” denotes the whole package of DG, scalar and bias embeddings is removed from pre-trained DG-BERT. When the expected word is not included in top 5 predictions, - is used to indicate its ranking.	65
4.9	The experimental results of GLUE benchmark are reported on validation set for comparison with the most related previous work. “DG-BERT w/o DG” denotes that the whole package of DG, scalar and bias embeddings is removed from pre-trained DG-BERT. My fine-tuning and Dict-BERT [172] are based on the average of five runs with different random seeds.	67
A.1	The hyperparameters for pre-training and fine-tuning DG-BERT are summarized in this table.	98
A.2	Models are compared on CPRAG-102 word prediction accuracies, and corresponding sensitivity tests: prefer expected completions over inappropriate completions.	98
A.3	Models are compared on ROLE-88 word prediction accuracies, and corresponding sensitivity tests: prefer expected completions over inappropriate completions.	98

A.4	Top word predictions are selected for the CPRAG-102 items by Ettinger [38]. The underlined predictions are the expected ones.	98
A.5	Top word predictions are selected for the CPRAG-102 items by Ettinger [38]. The underlined predictions are the expected ones.	99
A.6	Top word predictions are selected for the CPRAG-102 items by Ettinger [38]. The underlined predictions are the expected ones.	99
A.7	Top word predictions are selected for the ROLE-88 items by Ettinger [38]. The underlined predictions are the expected ones. * denotes inappropriate words.	99
A.8	Top word predictions are selected for the ROLE-88 items by Ettinger [38]. The underlined predictions are the expected ones. * denotes inappropriate words.	99
A.9	Top word predictions are selected for the ROLE-88 items by Ettinger [38]. The underlined predictions are the expected ones. * denotes inappropriate words.	99

Chapter 1

Introduction

1.1 Distributed Representations

Connectionism is the theory contributing to the advent of artificial neural networks that prevail in contemporary Artificial Intelligence (AI). The theory aims to simulate the cognitive process of human's brain in the form of highly interconnected, neuron-like processing units. In mathematical sense, the sum of weighted input values are activated by activation function (e.g., sigmoid, tanh, etc.) with a threshold.

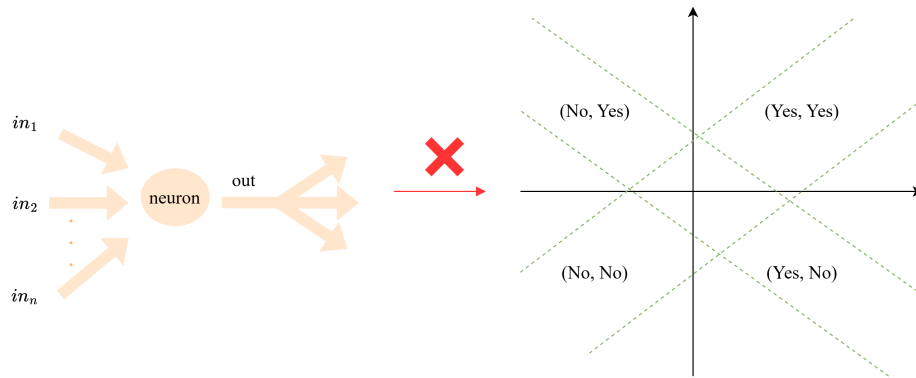


Fig. 1.1 Perceptron [128] with a single neuron cannot address XOR (exclusive or) classification using a linear regression line.

The first neural network can trace back to 1957. At the time, it was named Perceptron devised by Frank Rosenblatt [128]. As shown in the left side of Figure 1.1, Perceptron is composed of only one single neuron to handle multiple input features. The mathematical equation can be simply formulated as $out = f(w_1 \cdot in_1 + w_2 \cdot in_2 + \dots + w_n \cdot in_n)$, where w_i ($i \in [0, n]$) is the weight for each input feature and f is the activation function. Rosenblatt used the linear combination of input features to classify English alphabets. However, later in

1969, Marvin Minsky challenged the functionality of Perceptron by writing a book named Perceptrons [110]. Minsky pointed out that Perceptron cannot address the XOR (Exclusive Or) classification. As shown in the right side of Figure 1.1, when (Yes, No) and (No, Yes) belong to a category, there is no linear regression line can be drawn to distinguish them from (Yes, Yes) and (No, No). Nevertheless, even based on the computation supported by CPU devices nowadays, hidden layers with substantially more neurons can be used to process the output of multiplication between weight matrix and input features; furthermore, the output processed by multiple neurons can be used as input for next hidden layers. In such case, non-linear combinations are achievable for the operation taken on hidden layers, which can be used to address the above-mentioned XOR classification. This is usually known as cascading logistic regression.

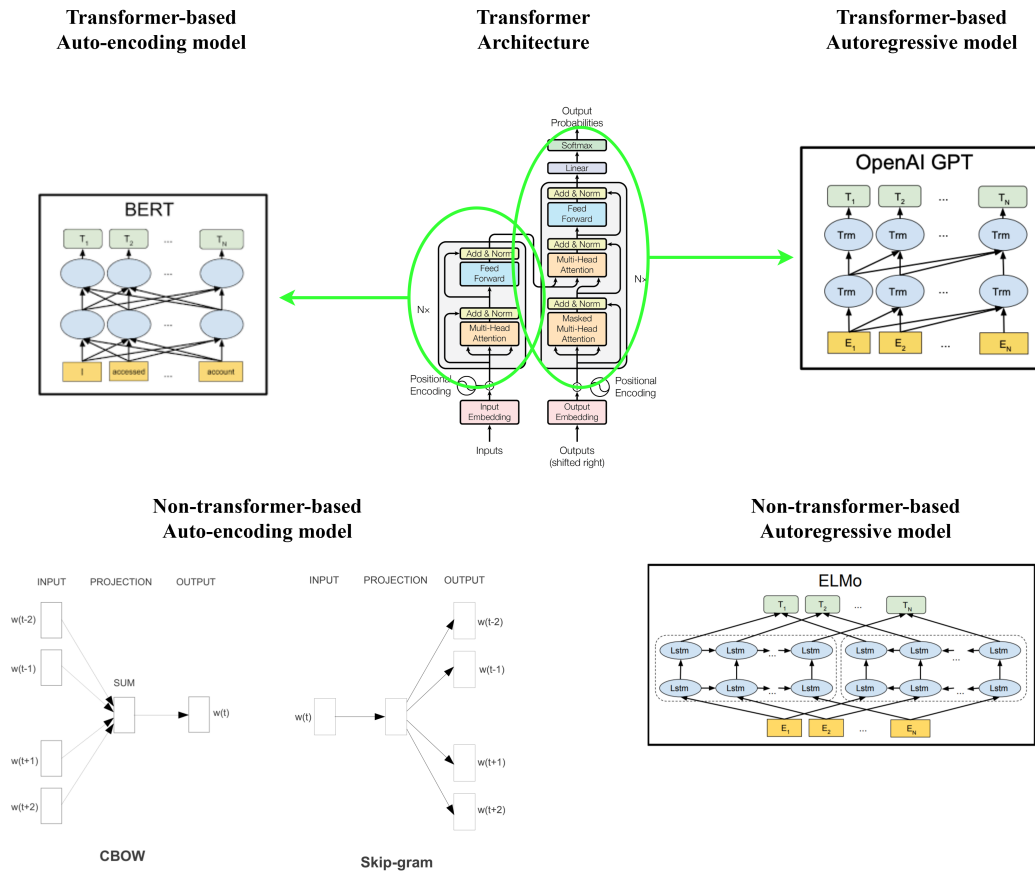


Fig. 1.2 (Non-)Transformer-based auto-encoding and autoregressive models. The visualizations are from Transformer [150], BERT [31] and Word2Vec [106] papers.

With the advancement of computational resources, matrix-related calculations or operations can be powerfully supported by GPU devices with high FLOPS and throughput. For any neural network with billions of model parameters, given a well-designed training objective,

it is trainable via back-propagation of gradient descent. In the Natural Language Processing (NLP) domain that I am dedicated to, Transformer [150] is the mainstream neural network architecture with powerful self-attention mechanism. This contributes to the existing language models such as auto-encoding models like BERT [31] and autoregressive models like GPT [122]. They are widely acknowledged as Large Language Models (LLMs) nowadays that impact not only NLP-related researches and people's daily lives. There also exist other counterparts which are not based on Transformer architectures. For example, Word2Vec embeddings [106] with only two projections matrices to be trained based on Continuous Bag-Of-Words (CBOW) is an auto-encoding model; ELMO [118] based on Long short-term memory (LSTM) architecture that is trained to predict words from left or right context is an autoregressive model. All those models are illustrated in Figure 1.2.

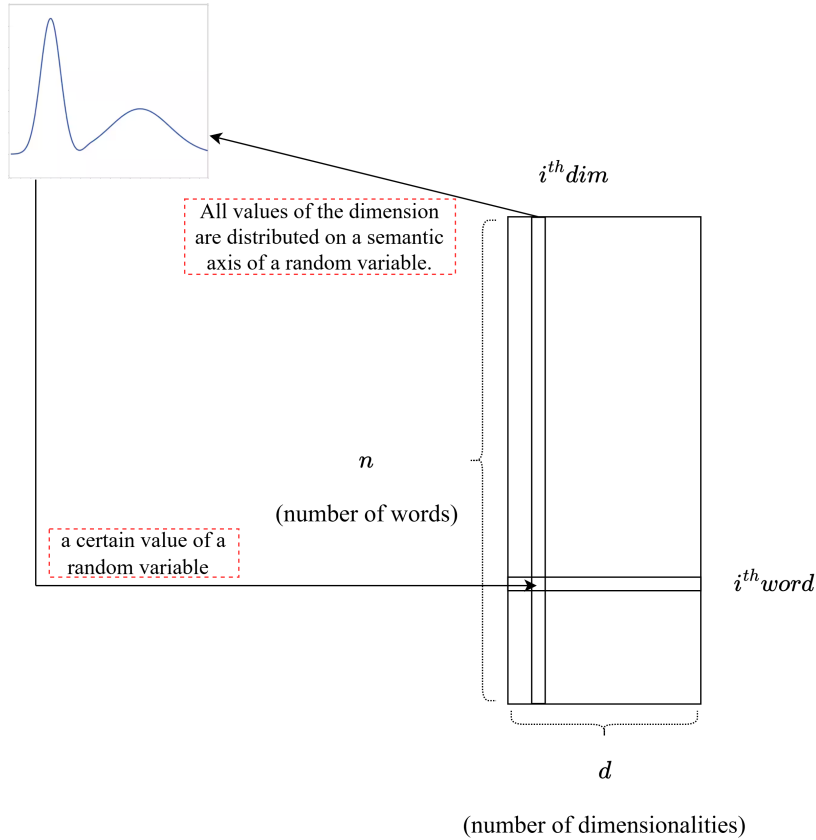


Fig. 1.3 Distributed representations are exemplified by word embeddings.

The semantics representations output from neural models are distributed representations. Distributed representations are dense vector representations (a.k.a. embeddings), where each dimension of all vectors is the semantic axis distributed with numerical values for different words, tokens, or bigger linguistic units like sentences. These pieces of semantic information are distributed across the entire vector for each single word or token. Figure 1.3 shows the

distributed representations exemplified by word embeddings, where each dimensional value in a vector is a float numerical number indicating how strongly this word can represent the dimension (semantic axis). For more observations, see the word embeddings processed by the algorithm of Independent Component Analysis [163].

1.2 Distributional Representations

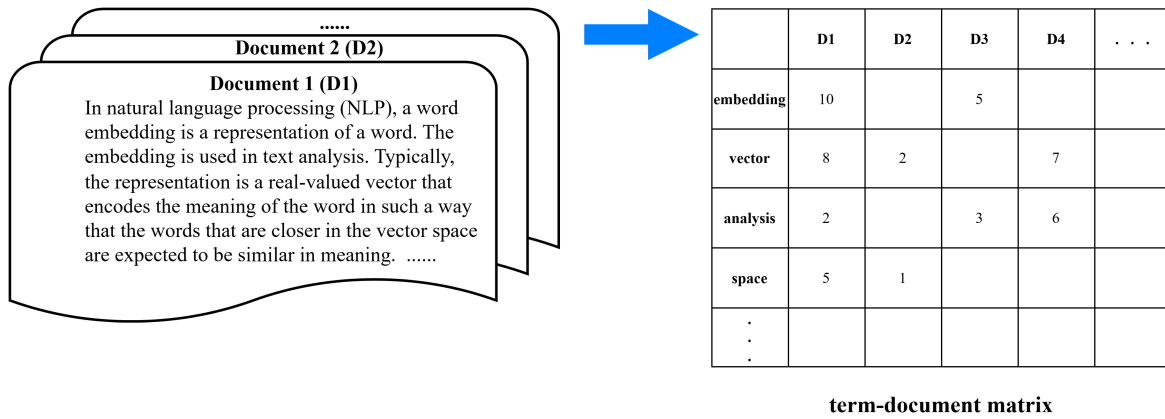


Fig. 1.4 An example is illustrated for the term-document matrix.

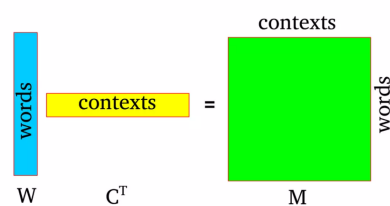
The distributional representations are sparse count vector representations. The method used to implement distributional representations is Vector Space Models (VSMs) [148]. A classical instance of VSMs is term-document matrix shown in Figure 1.4. In the matrix, term denotes all single words comprising different documents, and each term corresponds to a row vector while each document corresponds to a column vector. The idea was conceived for the SMART information retrieval system [130], which pioneered many of the concepts that are used in modern search engines [135]. The transpose of the matrix is document-term matrix. For measuring semantic similarities, words having similar document distributions or documents having similar word distributions tend to achieve high cosine similarities.

The original elements in the vector representation of each document are Bag-of-Words (one-hot encoding). The frequencies of words in a document, to some extent, can determine how relevant the document is to a query. The Bag-of-Words hypothesis is the basis for applying the VSMs to information retrieval [131]. However, the original term-document matrix has some limitations when applied to information retrieval tasks. When the size of the matrix ($M \times N$) is enormous, it will consume a considerable amount of memory. Moreover, if the documents are composed of short texts such as phrases and sentences, the matrix will become sparse. To address this limitation, matrix factorization was introduced, and

one standard approach for this mathematical operation is Singular Vector Decomposition (SVD) [28], which is engaged in the process of LSA (Latent Semantic Analysis) [83]. SVD decomposes the matrix $X (M \times N)$ into the product of three matrices $U (M \times K) \cdot \Sigma (K \times K) \cdot V^T (K \times N)$, in which U is a left singular matrix, Σ is a diagonal matrix of singular values, and V^T is a right singular matrix, where $K \ll M$ or N . From the perspective of mathematics, this is a operation of dimension reduction. On the other hand, from the perspective of computational linguistics, I consider it as a latent semantics detection: U is composed of latent semantics of all words, and V^T is composed of latent semantics of all documents. Proximity in the induced latent space has been shown to correlate with semantic similarity [105]. Within the right singular matrix, each document has k latent semantics, and then one just needs to use some measurements like cosine similarity to judge which documents are semantically similar to a given query, and then display them to users.

1.3 Common Grounds in Both Representations

Reinterpretation



Does this remind you of something?

Very similar to SVD over distributional representation:

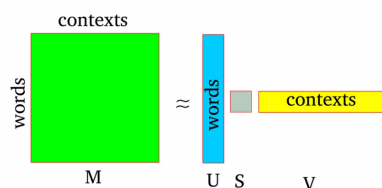


Fig. 1.5 Distributed representations of word embeddings are somewhat equivalent to distributional representations after matrix factorization. The visualization is from Yoav Goldberg's presentation for their paper [88].

The distributional representations after matrix factorization are also distributed representations. This is not only because the factorized representations are dense vectors that are same as word embeddings shown in Figure 1.3. It is theoretically and empirically proved in a published paper [88] that the training of word embeddings is a operation of implicit

matrix factorization. Figure 1.5 shows the comparison of training SGNS (skip-gram with negative-sampling; depicted in Figure 1.2) model of Word2Vec [106] to the matrix factorization algorithm of Singular Value Decomposition (SVD) — the projection matrix of words (trained word embeddings) multiplied by the projection matrix of context (if not discarded after training) approximates the multiplication of decomposed matrices of SVD.

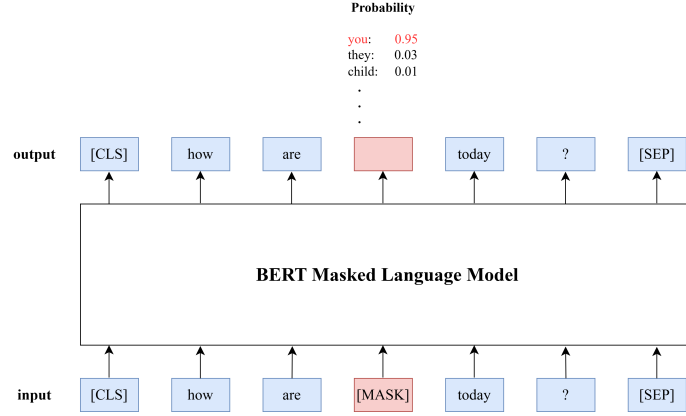


Fig. 1.6 An example is shown for the MLM pre-training objective of BERT.

Another common ground between distributed representations and distributional representations is the distributional hypothesis [57, 44] — “Words that occur in similar contexts tend to have similar meanings.” or “You shall know a word by the company that it keeps.”. For distributional representations, in terms of the term-document matrix, the context is count distributions of documents, and words having similar distributions tend to achieve high cosine similarities. For neural models that generate distributed representations, their training objectives are also based on the hypothesis. In the case of the MLM (Masked Language Modeling) pre-training objective of BERT, Equation 1.1 explains the example of target-word prediction based on context words in Figure 1.6, where $\hat{x}$ is the corrupted input sentence with the masked word $\bar{x}$, and x' is the vocabulary of BERT and θ is trainable model parameters. The $h_{\theta}(\hat{x})$ is the representation of masked word output from BERT via self-attention mechanism. As the weight of word embeddings is tied to the prediction head, *word_embed* in the equation indicates the prediction head.

$$\arg \max_{\theta} P_{\theta}(\bar{x}|\hat{x}) = \arg \min_{\theta} -\log \frac{\exp(\text{word_embed}(\bar{x}) \cdot h_{\theta}(\hat{x}))}{\sum_{x'} \exp(\text{word_embed}(x') \cdot h_{\theta}(\hat{x}))} \quad (1.1)$$

This is the reason why both representations are considered as the representations conveying distributional semantics [36].

1.4 Motivation & Research Goal

There are strength and weakness for both types of representations.

For distributional representations, given a particular knowledge corpus like paraphrase datasets, dictionaries, etc., the transpose of term-document matrix, namely document-term matrix, can be employed to capture semantic relatedness of entries in those corpora. This is an effective approach for information retrieval [130, 135]. The main reason is that the frequency of knowledge entities in any knowledge corpus is equal (only one occurrence), and thus they can be measured straightforwardly according to their count distributions of word features. However, with unlimited large text corpora, the term-document matrix is not only enormous, consuming a significant amount of memory, but also sparse.

For distributed representations, since input word units are decomposed into sub-word (token) units (e.g., “jubilant” composed of “ju”, “##bil”, and “##ant” in BERT vocabulary), token embeddings can be trained to learn overall semantics for large numbers of input words. However, semantic representations of neural models are known to be anisotropic [48, 37]. An indication of this is that the distribution of words is not directionally uniform (occupying a narrow cone) in the vector space. Gao et al. [48] claims that during the pre-training stage of language models, most words are pulled towards a local region of the vector space while low-frequency words are pushed against the region. As knowledge entities (e.g., domain-specific terminologies) are typically low-frequency words, distributed representations cannot decently convey semantic information for the entities. This is the reason why there are many studies endeavored to accomplish K-PLMs (Knowledge-enhanced Pre-trained Language Models) [172, 178, 119].

As discussed above, both types of representations are complementary to each other. The strength of one can be used to address the weakness of another. Herein, it is meaningful to investigate whether their fusion can contribute to the development of NLP. During my PhD study, I prove the feasibility of my idea with two proposed approaches — fusion through mapping and fusion through structural connection. Two approaches can respectively address the problems of computationally expensive continual learning for NLP tasks and anisotropic semantic representations of language models. Also, I write two journal articles for my approaches, which are accepted with peer review [92, 93].

1.5 Dissertation Structure

Apart from this introduction chapter, the rest of my dissertation consists of four chapters concerning proposed approaches for the fusion of distributed representations and distributional

representations. In Chapter 2, I will go through all prior works related to my approaches. In Chapters 3 and 4, thorough descriptions of my proposed methods are provided regarding how I fuse distributed representations and distributional representations. In Chapter 5, I will conclude the contributions of my works for NLP community, and also discuss my future research directions.

Chapter 2

Related Works

In this chapter, the previous works related to my proposed approaches [92, 93] are rephrased and reintroduced. Although most of them are not directly related to the topic of this dissertation regarding the fusion of distributional representations and distributed representations, it is necessary to mention the works in terms of the comparison and similar conceptualization with my approach.

2.1 Textual Similarities for Paraphrase Identification

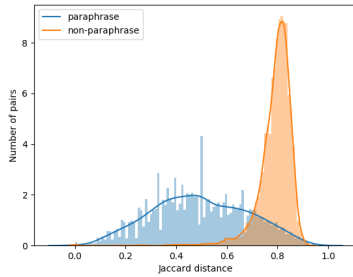


Fig. 2.1 PAN.

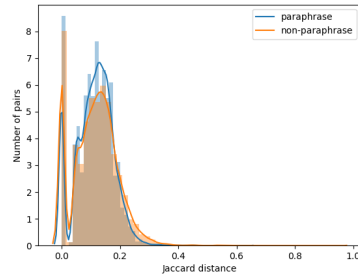


Fig. 2.2 PAWS-wiki.

Lexical overlap features [153] can be combined with the transferred features (the semantic features conveyed in distributed representations) as input to each individual classifier. This technique is not unusual for feature-based transfer; for example, Yin and Schütze [166] combined the machine translation metrics [97] with the flattened features output from their Bi-CNN architecture. As the Jaccard distance¹ illustrated in Figures 2.1 and 2.2 for two

¹All the tokens are converted into small case without stemming or removing stopwords and punctuation.

paraphrase corpora², lexical overlap features are noticeably cost-effective for task datasets like PAN [97] (see Figure 2.1); however, they become unviable when lexical overlaps are indistinguishable between paraphrase and non-paraphrase sentence pairs in task datasets like PAWS-wiki [176] (see Figure 2.2). The combination can have the transferred features enjoy their merits while their demerits can be improved by the transferred features. From another perspective, 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 [127] shown in a recent work [24] for data augmentation, some works still consider lexical overlap features as criteria: Nighojkar and Licato [114] use BLEURT [136] metric to calculate reward for sentence pairs that are mutually implicative but lexically and syntactically disparate; Kadotani et al. [75] use edit distance to decide whether source and target sentences require drastic transformation, so that the training order of curriculum learning [8] can be determined for better performance of paraphrase generation; Jaccard distance is used in Meng et al. [103]’s work to filter generated paraphrase candidates.

Ji and Eisenstein [73] 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 [33] task with this special feature engineering process. However, this approach has the following limitations: (1) the TF-KLD weighting scheme relies on transductive learning [46] 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 [168] can address the first limitation, but is not viable for the second limitation.

2.2 Task-specific DNNs for Paraphrase Identification

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 [108, 116, 107, 117], 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 [14], 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 [166, 58, 169]: using two sub-

²Jaccard distance calculated for two PI corpora take into account training, development (if it exists), and test sets.

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 [139] 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 [33]. As explained in Yin and Schütze [166]’s work, this is due to unavailability of highly accurate parsers for tree structure.

2.3 Transfer Learning Approaches for Paraphrase Identification

Shi et al. [137] has 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 [117] to test its effectiveness. The task-specific architecture (orthogonal transformation matrix) placed prior to embedding models is referred to as retrofitting methods [39, 173, 49]: to incorporate semantic knowledge from external resources into word embeddings. The external resources that Shi et al. [137] used to train their task-specific architecture are the paraphrase sentence pairs from three corpora: PAN [97], Sampled Quora [71], and MRPC [33]. The parameters of the trained architecture then can be used for PI, NLI and STS tasks, which is parameter-efficient.

The adapter module [64] comprises the network layers with comparatively small size of parameters stitched into BERT [32] 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 [30], Yosinski et al. [170] 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 [64], which matches the popular strategy in fine-tuning [66].

2.4 Continual Learning

Continual learning [145], 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; e.g. 20 QA tasks in bAbi corpus [159], permuted handwritten digits recognition [149], text classification tasks with different class [26], and so on. A recent research attempt [142] has managed to handle 5 disparate NLP tasks by following decaNLP [100] to treat all tasks as QA format. The biggest problem for continual learning is catastrophic forgetting [101, 45] — 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) [78], 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 [20] and Progressive Neural Networks [129] to adapter module [65] 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 [68].

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 [26] or generated pseudo-data [142]. With this benefit, a new task can be trained together with the old data in a multi-task learning manner [15], which is considered as the upper bound of continual learning. It is also shown in GEM (Gradient Episodic Memory) [96] that the old real-data can be used to prevent gradient update from being biased towards a new task during optimization.

Curriculum learning [8], 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. [174], which explores the learning order of various image-related tasks; e.g. detecting 3D edges and normal vectors of images first can help learn point matching and reshading effectively.

2.5 Target Words-glossed Pre-training for Language Models

As suggested in applied linguistics [79, 158], when encountering unknown words, people using a dictionary while guessing their meanings from their context may not only learn more words immediately but also recall them better and be able to use them.

In NLP, a previously proposed Dict-BERT [172] matches the human learning process: to append definitions in dictionaries to input sequences of BERT [31] for masked rare words during pre-training. Dict-BERT appears to acquire vocabulary more effectively than BERT, which is reflected by the improvements on standard downstream tasks [155] and domain-specific tasks after domain adaptive pre-training [55]. In other target words-glossed pre-training like the early representative ERNIE [178] and KnowBERT [119], instead of dictionary resources, they use Knowledge Representations (KR) [12, 4] learned from structured knowledge graph [3, 140] to fuse the knowledge conveyed by KR into hidden states of target words. Due to different vector spaces, to address Heterogeneous Information Fusion (HIF) between KR and hidden states, separate self-attention is usually performed before fusion. Their better acquisition of vocabulary leads to the improvements on knowledge-driven tasks.

These studies belong to the category of Knowledge-enhanced Pre-trained Language Models (K-PLMs). When it comes to domain-specific corpora, target words-glossed pre-training is intrinsically advantageous, because the corpora contain denser knowledge entities

than open domain corpora like Wikipedia. The sub-categorical difference is that ERNIE and KnowBERT are in the set of explicit fusion (external KR is structurally connected with BERT layers) while Dict-BERT belongs to implicit fusion (external semantic knowledge is merged into the input sequence of BERT).

There exist other K-PLMs than target words-glossed pre-training ones. For example, in K-BERT [90] and a recent DictBERT [19], KR and definition embeddings do not participate in the MLM objective of BERT, which is only used in fine-tuning. Target words-glossed fine-tuning is also employed by some studies [67, 11] for word sense disambiguation tasks. For another example, in CoLAKE [143] and KEPLER [157], learning KR is integrated into the pre-training process, which is more of regularization than an auxiliary tool. Using glosses for regularization is also adopted by a study called GR-BERT [89] for lexical substitution and unsupervised semantic textual similarity tasks. Despite leveraging glosses for target words in different manners, their strengths further reflect the fact that external semantic information can assist BERT in acquiring better word senses.

2.6 Knowledge Infusing Before the Era of Language Models

Word meaning representations [2] evolve as technologies advance. One invariant technical trend is to inject external semantic knowledge into the representations. The knowledge stems from diverse semantic resources: semantic networks [109], knowledge graphs [3, 140], commonsense reasoning [134], paraphrase datasets [34, 47], dictionaries and glossaries [59]. Besides the studies mentioned in the previous section regarding knowledge injection for language model representations nowadays, there exist three previous stages for other representations.

1st Stage. Vector Space Models (VSMs) [148] such as term-document matrix use distributional counts to measure semantic similarity. Matrix factorizations like SVD [29] and LDA [10] can be further used to decompose sparse count-based representations into dense latent representations. As there is no notion of antonym in VSMs [82], semantic representations of antonyms also tend to have high similarity by this approach. Yih et al. [165] utilize a thesaurus to assign polarity values (1 if synonym and -1 if antonym) to each pair of words. Besides synonyms and antonyms, Chang et al. [18] take hypernyms into consideration by means of WordNet [109], and use three separate matrices to display word relations based on the three notions.

2nd Stage. To incorporate semantic knowledge into word embeddings like word2vec [106] is reminiscent of many interesting previous works. According to Glavaš and Vulić [50]’s work, technically there exist two categories: Joint Specialization Models (modifying pre-training objectives of embedding models by external semantic resources) [171, 146] and Post-Processing Models (injecting external semantic knowledge into pre-trained embedding models) [40, 13]. The latter is also known as retrofitting methods. Two representative studies that utilize dictionaries are CPAE [13] and dict2vec [146]. The CPAE model post-processes the pre-trained word2vec using an LSTM encoder, where inputs are words of concatenated definitions (considering polysemy) represented by word2vec and outputs (last hidden states) are definition embeddings. The post-processing procedure is completed by a decoder to reconstruct the bag-of-words representation of the definitions. The dict2vec method modifies the pre-training objective of word2vec skip-gram model by adding a cost for predicting related words. The relatedness is based on positive sampling, which is determined by words that are either strong or weak pairs (words co-occur in definitions of each other). The unrelatedness based on negative sampling strengthens the cost function.

3rd Stage. The transition from word embeddings to LLMs is contextualized word embeddings called ELMo [118]. Shi et al. [138] discover 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 for evaluation by utilizing paraphrase datasets. The PAR method belongs to the post-process category.

Chapter 3

Representation Fusion Through Mapping

In this chapter, I describe my fusing approach by mapping weighted distributed representations to factorized distributional representations. This approach, with the title “Parameter-efficient feature-based transfer for paraphrase identification” [92], has been accepted by Natural Language Engineering journal. It contributes to the continual learning of paraphrase identification and other sentence-pair tasks with marginal consumption of computational resources and storage.

3.1 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; Semantic Textual Similarity (STS) for scoring semantic similarity of a sentence pair. Many types of approaches have been published in the previous works [17] for PI task and two of its variations (NLI and STS). In this article, I 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, I first introduce the research goal that motivates my proposed method based on the relevant technical background. Then in the second subsection, I describe the benefit for a real-world scenario based on my technical novelty.

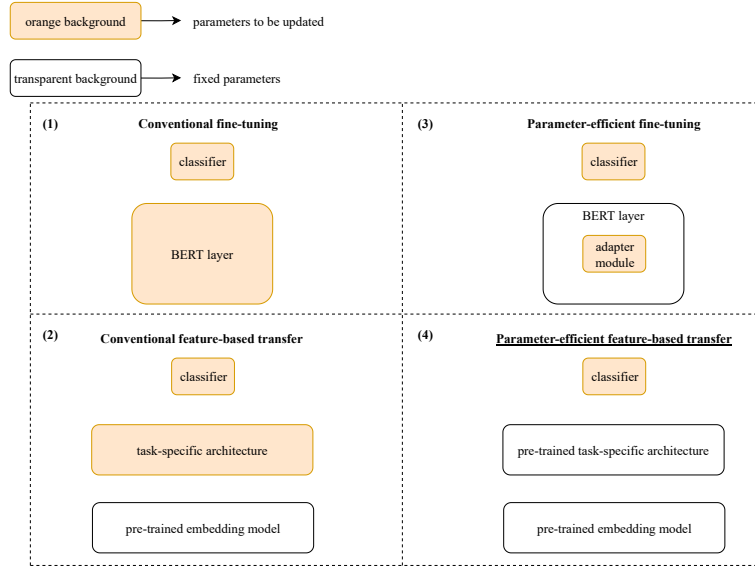


Fig. 3.1 Parameters that need to be updated for each PI, NLI, or STS task. The underlined denotes my work.

3.1.1 Technical Background & 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 [153] and machine translation evaluation metrics [97]; (2) using external lexical knowledge like WordNet [42, 43]; (3) modeling divergence of dependency syntax between two sentences [25]; (4) distributional models with matrix factorization [54, 73]. The traditional approaches mainly consist of unsupervised methods and feature engineering. Their demand for computational resources are 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 [33] in the ACL link¹. In light of that fact, I come up with my research question — *how can I preserve the low computational costs of traditional approaches but yield better task performance?* To seek an approach to it as my research goal, I take a further investigation on current neural network-based transfer learning approaches.

With the advent of various deep neural network models [35, 151, 66, 164], 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

¹[https://aclweb.org/aclwiki/Paraphrase_Identification_\(State_of_the_art\)](https://aclweb.org/aclwiki/Paraphrase_Identification_(State_of_the_art))

language models like BERT [32] need to be fine-tuned. On the other hand, feature-based transfer does not require parameter update of pre-trained embedding models like ELMo [117]; 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 3.1).

Accompanied by the successful performance, there comes one challenge to transfer learning approaches: expensive computational resources [141]. As fine-tuning tends to achieve better performance than feature-based transfer shown in recent works [66, 32], recently proposed resource-lean approaches are mainly based on BERT [32] architecture, focusing on model compression such as network layers repeating [81] and knowledge distillation [133]. 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 remain uninvestigated.

Still, the adapter module proposed by Houlsby et al. [64] 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 3.1). Houlsby et al. [64] applied the adapter module to BERT [32] 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 [27], and also works for autoregressive models like P-tuning [94] for GPT [123].

The efficient usage of architectural parameters also resonates with a current trend² in NLP community, which encourages researchers to empirically justify the model complexity beyond benchmark performance. However, to the best of my knowledge as of writing, direct research attempts regarding parameter-efficiency are rarely made for feature-based transfer. In the last two years, only one indirect attempt was conducted, the PAR (Paraphrase-Aware Retrofitting) method proposed by Shi et al. [137], 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 [145], the parameters of the re-trained network are inclined to forget how to perform previous tasks — catastrophic forgetting [101, 45].

²<https://sites.google.com/view/sustainlp2020/home?authuser=0>

While related works are centered around language model-based fine-tuning, I 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, I 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 3.1), 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 [65] tuned on task A cannot be directly used for task B without any modification. However, in this article, I show that it is feasible to feature-based transfer.

For readability in the rest of this article, I use the acronym PEFBAT (Parameter-Efficient Feature-Based Transfer) to denote my 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 my research goal. For each PI, NLI, or STS task, 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. Such technical novelty is also conducive to a real-world scenario, which is another contribution of PEFBAT described in the next subsection.

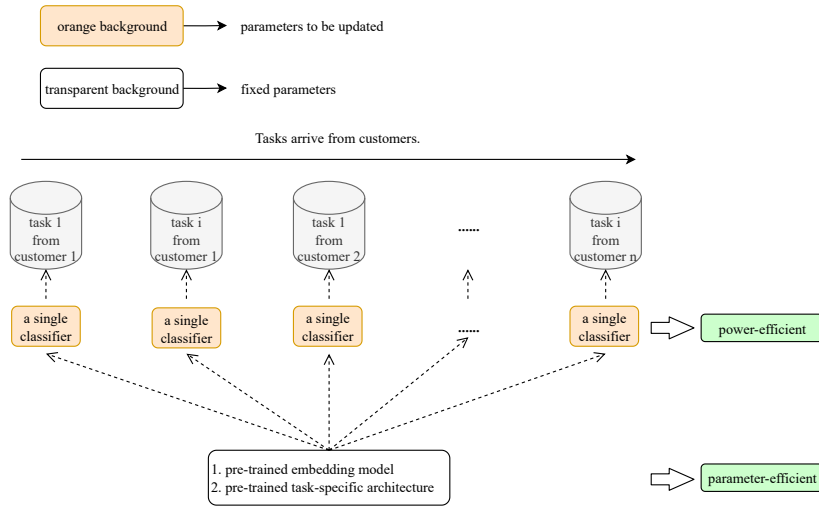


Fig. 3.2 Parameter-efficient feature-based transfer for power-efficient continual learning.

3.1.2 Practical Use in Real-World Scenario

Similar to adapter module [65], 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 3.2). 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)³ with batch size 32 and epochs 100, for small datasets like MRPC [33] (4k training data), training a MLP takes up approximately 1 minute. For large datasets like QQP [71] (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 per classifier), 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.

3.2 Proposed Method

The technical motivation towards PEFBAT is introduced in Section 3.1.1. In this section, I move into the implementation details and mechanism of PEFBAT. I first describe the modeling of initial sentence-pair representations and the objective of my 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 my pre-trained task-specific architecture can output, which are presented together with other features in the last subsection.

3.2.1 Initial Sentence-Pair Representations & 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 3.3). The entity and action components

³The GPU device that I use for my experiments is NVIDIA RTX 2080 TI.

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 3.3). In addition, to facilitate PI, the reward component (details explained in Sections 3.2.2 and 3.2.2) containing (non)paraphrase-like characteristics is combined with the other three components to form the initial sentence representations of a sentence pair. The pre-training objective, therefore, is to have the initial representations generate discriminative latent representations. The latter contains discriminative features, which is conducive to PI.

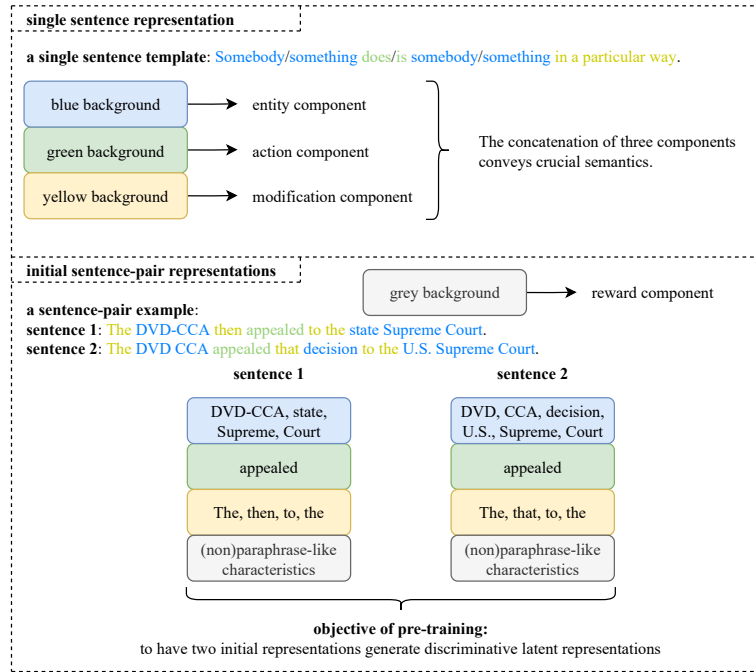


Fig. 3.3 Modeling of initial sentence-pair representations.

3.2.2 Latent Space Pre-training

To generate discriminative sentence latent representations, I refer to Ji and Eisenstein [73]’s work by taking advantage of labeled data, and my concrete strategy in this article is the dual weighting scheme shown in Figure 3.4. 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

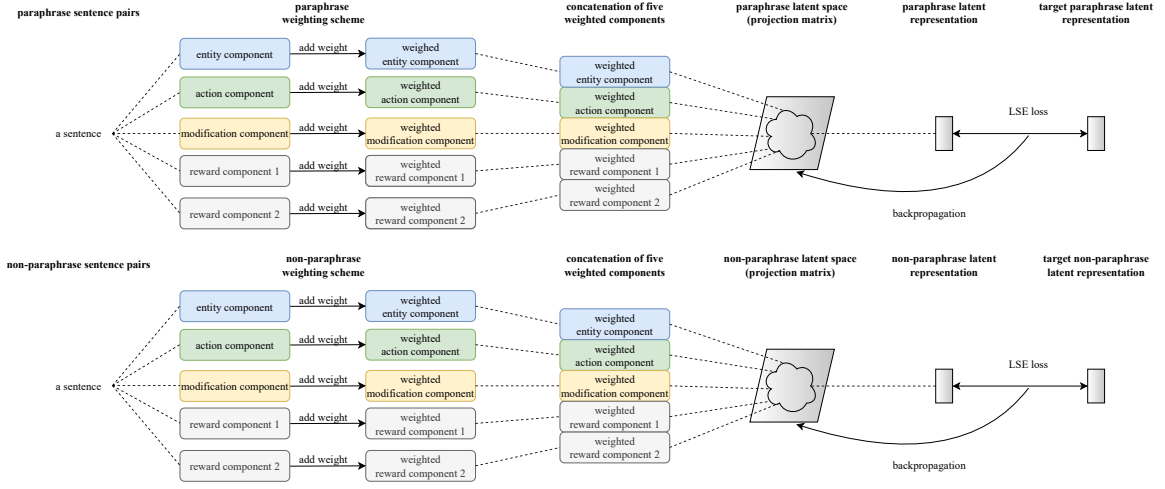


Fig. 3.4 Mapping mechanism of pre-training.

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 3.3, an ablation study is performed for validation.

The two pre-trained latent spaces are namely the task-specific architecture, the fixed parameters of which can be used for multiple PI, NLI, and STS tasks. During inference shown in Figure 3.5, 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 3.2.3.

Before stepping into further implementation details, I introduce the embeddings, toolkits and corpus resource that are employed for pre-training. As I consider the application to low-resource natural languages with limited labeled and unlabeled data [61] as my future work (discussed in the last section), the choice of the resources are based on the consideration of “lightness”: requirement of small training data size and easy availability.

Corpus Resource: The corpus that I use is only the training dataset of Microsoft Research Paraphrase Corpus (MRPC) [33] consisting of 4,076 sentence pairs, in which 2,753 pairs are labeled as paraphrase, which is so far the smallest English paraphrase corpus.

Pre-trained Embeddings: I choose fastText [107] trained with subword information on CommonCrawl as my pre-trained embeddings because it is not deep learning architecture-

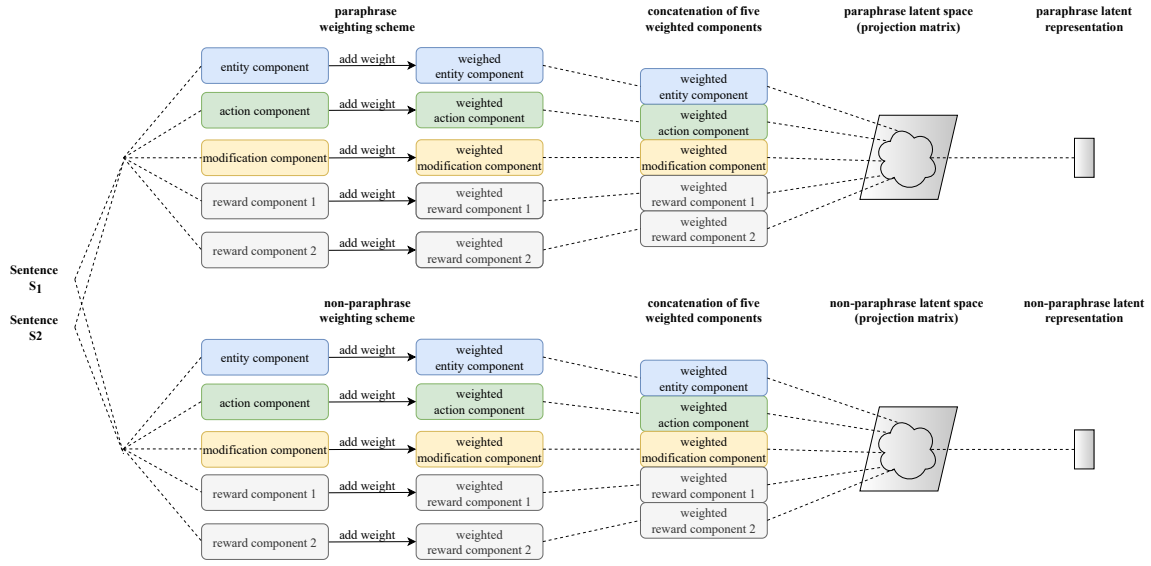


Fig. 3.5 During inference, each sentence of any sentence pairs can obtain two types of latent representations.

based, which is applicable to low-resource natural languages [112]. Moreover, it achieves better performance than GloVe [116] trained on CommonCrawl in SentEval framework [23].

Toolkits: Considering easy availability, I utilize part-of-speech tags (POS-tags) provided by NLTK⁴ for categorizing entity, action and modification components. The scikit-learn tool⁵ is used for distributional models and matrix factorization.

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
- **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

⁴<https://www.nltk.org/>

⁵<https://scikit-learn.org/stable/>

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, my 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 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.

Dual Weighting Scheme

The dual weighting scheme is summarized in Table 3.1, which is utilized to determine whether a component should be strongly or weakly weighted. In the scheme, the 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, the weighting criterion is “occurrence difference”. I utilize the characteristics-based thresholds (see Figures 3.6, 3.7, and 3.8) to determine strong or weak rewards. The main threshold is Jaccard distance (abbreviated as *j-dist* shown in Figure 3.8) [72]. I perform PI on MRPC [33] 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 3.3, which is time-saving. Only for pre-training my latent spaces, I need to ensure every default setting is well set. The sentence length (abbreviated as *sent-len* shown in Figure 3.6) and absolute difference of sentence length (abbreviated as *sent-len-diff* shown in Figure 3.7) are the main factors that can affect Jaccard distance, so I 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.

For the three word-embedding components (entity, action and modification), my weighting criterion is “label”. As adverbs like negation (not) convey discriminative semantics, I 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 I 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 3.9) is the assumption I 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, I also pre-train another two latent spaces without the threshold. In Section 3.3, I conduct the experiments using the latent spaces pre-trained ‘with *digit-count*’ or ‘without *digit-count*’. Below I 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).

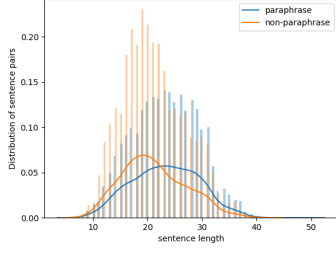
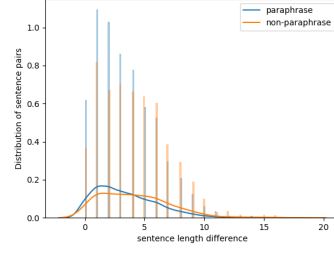
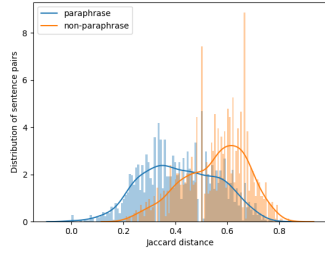
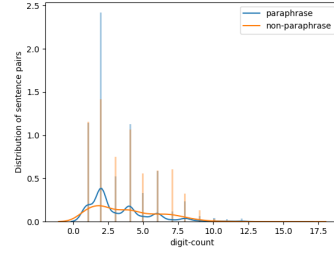
1. Yucaipa owned Dominick’s before selling the chain to Safeway in 1998 for \$2.5 billion.
2. Yucaipa bought Dominick’s in 1995 for \$693 million and sold it to Safeway for \$1.8 billion in 1998.

Table 3.1 Dual weighting scheme.

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

Target Latent Representations

Distributional models with matrix factorization are used to generate latent representations for all sentences in the training dataset of MRPC [33]. For factorization, I follow Guo and Diab [54]’s work by choosing Singular Value Decomposition [28] and the length of 100 for latent dimensionality. Moreover, I normalize the factorized matrix, as the latent representations are

Fig. 3.6 *sentence length*.Fig. 3.7 *absolute difference of sentence length*.Fig. 3.8 *Jaccard distance*.Fig. 3.9 *digit-count*.

the target layer of my 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, I 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.

Hyperparameters of Pre-training & Method of Determining Weights

For each sentence in the training dataset of MRPC [33], the concatenation of its five weighted components is mapped to its latent representation. As mentioned earlier in this subsection shown in Figure 3.4, 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 my pre-training are input vector length (1,500), target vector length (100), activation function (tanh), loss function (LSE), epochs (500), learning rate ($5e-4$), and optimization (vanilla SGD).

Before formal pre-training, I first tune the weights with small training epochs (≤ 50). The determined weights are illustrated in Table 3.1. For both reward components, the strong and weak weights are 1 and 0.2 respectively; 1.5 and 0.5 (I call it “weight pair”) are set for the three word-embedding components. How they are determined is described below.

1. First, I 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 I 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, I observe that the error loss decreases significantly in case of 0.2, and hardly drops when 0.1 is used. Therefore, I decide 0.2 as the weak weight for both reward components.
3. After the decision of the weak weight, I start to tune the “weight pair”. I use only 50 epochs to pre-train my latent spaces, and experiment with different pairs from (1.1, 0.9) to (1.9, 0.1) at the pivot of 1. I 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, I set out to formally pre-train the two latent spaces with the full training epochs of 500 (4 GPU³ hours). I 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 3.3 show that my pre-trained latent spaces are not overfitting and limited to MRPC. As for reproducibility, I 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).

Table 3.2 Discriminative similarities are shown for the spaces pre-trained ‘with digit-count’.

Sentence pairs	Lp(S1) & Lp(S2)	Lp(S1) & Lp(S2)	Lnp(S1) & Lnp(S2)	Lnp(S1) & Lnp(S2)
	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 3.3 Discriminative similarities are shown for the spaces pre-trained ‘without digit-count’.

Sentence pairs	Lp(S1) & Lp(S2)	Lp(S1) & Lp(S2)	Lnp(S1) & Lnp(S2)	Lnp(S1) & Lnp(S2)
	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

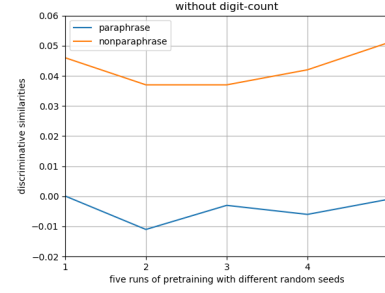
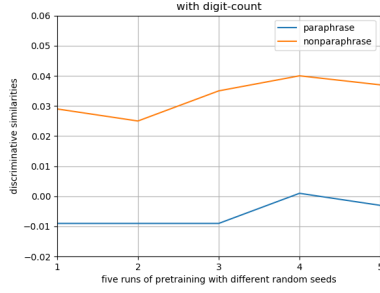


Fig. 3.10 Reproducibility test for ‘with digit-count’. Fig. 3.11 Reproducibility test for ‘without digit-count’.

3.2.3 Discriminative Similarity

As mentioned earlier in Section 3.2.1, the pre-training objective 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 3.5, 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, I calculate the cosine similarities for Lp(S₁) & Lp(S₂) and Lnp(S₁) & Lnp(S₂), using the sentence pairs in the training dataset of MRPC. The results are shown in Tables 3.2 and 3.3.

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). I 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 my feature set as the primary features. All the features are collectively presented in the next subsection.

Table 3.4 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)

Furthermore, I use the discriminative similarities (the examples with concrete figures are shown in the parentheses of Tables 3.2 and 3.3) to measure the reproducibility of my latent space pre-training. I additionally pre-train both types of latent spaces for 5 times respectively, using the same hyperparameter setting mentioned in Section 3.2.2 with different random seeds. The results are shown in Figures 3.10 and 3.11. 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 shown in Tables 3.2 and 3.3, 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 my pre-training is robust for random seeds.

3.2.4 Features

My latent space-related features (1-12) and lexical overlap features (13-19) are summarized in Table 3.4. The features serve as input to multiple independent classifiers for different task datasets. Inspired by Ji and Eisenstein [73], 1-4 are concatenated as basic features. Although 9-12 are not directly related to my 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) [153], I also enrich the granularity level by 13 (sentence level) and

19 (character level) [120]. Both types of features are mutually complementary to each other, and I delve deeper into this by performing a concrete analysis in Section 3.3.

3.3 Experimental Results

The experimental results presented in this section is to verify what I claimed in the previous sections. First, to test the effectiveness of two weighted reward components, the ablation study is performed. Then I 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 Section 3.2.4.

3.3.1 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 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 my 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 my 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, I 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.

Table 3.5 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

Table 3.6 Discriminative similarity is not reflected by “no reward” model ‘with digit-count’.

Sentence pairs	Lp(S1) & Lp(S2)	Lp(S1) & Lp(S2)	Lnp(S1) & Lnp(S2)	Lnp(S1) & Lnp(S2)
	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

- **baseline**: my baseline is pure feature engineering: the lexical overlap features (13-19) in Table 3.4.
- **origin**: the two latent spaces are pre-trained based on the implementation details described in Section 3.2.2.
- **no j-dist**: the two latent spaces are pre-trained based on the implementation details described in Section 3.2.2 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 3.2.2 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 3.2.2 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*, I include the thresholds of the lexical overlap features (13-19) in Table 3.4 to weight two reward components together. The 600 dimensionality

Table 3.7 Discriminative similarity is not reflected by “no reward” model ‘without digit-count’.

Sentence pairs	Lp(S1) & Lp(S2)	Lp(S1) & Lp(S2)	Lnp(S1) & Lnp(S2)	Lnp(S1) & Lnp(S2)
	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

is evenly distributed by the thresholds. Accordingly, this model is the two latent spaces pre-trained based on the implementation details described in Section 3.2.2 except that the two reward components are weighted by multiple thresholds during pre-training.

As for evaluation, I perform PI on MRPC [33] using the SVM⁵ with linear kernel as the classifier, and metric is accuracy and F1 score⁶. 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 3.4 as input to the classifier. The experimental results are shown in Table 3.5, from which I 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 my expectation, because using lots of characteristics-based thresholds 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 DNNs¹. While performance level is similar, my 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.

1. The pre-trained type ‘with digit-count’ is the assumption I 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

⁶Unlike other tasks like VUA-18 [86] and VUA-20 [85] 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 my experiments, I use the two metrics only.

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 3.3.2.

2. 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 3.2 and 3.3, 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 3.6 and 3.7, 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 3.3.3, I delve deeper into the utility of discriminative similarity.

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. I also take a probe into the replacement; for example, I replace *j-dist* with the Levenshtein distance (feature 13 in Table 3.4) to weight one reward component, but the results do not outperform “origin”.

3.3.2 Pre-trained Latent Spaces with Manually Adjustable Thresholds

The experimental results shown in this subsection is to verify three of my claims: (1) the pre-trained latent spaces are not limited to MRPC [33]; (2) PEFBAT can also handle NLI and STS tasks other than PI tasks; (3) PEFBAT is capable of power-efficient continual learning. I 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 [65] is considered as my 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, I underline that all the tasks are tested with the fixed parameters of pre-trained latent spaces without task-specific re-training. My open-source code is available under this link⁷.

Metrics & adjusted thresholds: I experiment with six PI tasks, two NLI and STS tasks respectively, and the results are summarized in Table 3.8. For the PI tasks including MRPC

⁷<https://github.com/ryuliuxiaodong/latentspace>

Table 3.8 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	77.6/84.4	93.3/93.3	81.2/74.5	88.3/69.3	74.1/78.6	<u>74.5/71.4</u>	82.1/76.7	72.3/72.0	<u>85.6</u>	79.3
Adjusted thresholds 'with digit-count'	-	93.5/93.5	81.4/74.6	88.4/70.1	74.7/79.6	<u>75.0/71.1</u>	-	-	-	79.4
Default thresholds 'without digit-count'	78.2/84.8	93.5/93.4	81.5/74.7	88.3/68.6	74.5/79.9	<u>73.6/68.4</u>	82.3/76.9	72.3/71.8	<u>85.2</u>	79.4
Adjusted thresholds 'without digit-count'	-	93.6/93.5	81.3/75.1	88.3/69.1	74.8/80.4	<u>73.7/68.6</u>	-	-	-	79.8
Trainable parameters	< 0.001M	≈ 0.48M	≈ 0.48M	≈ 0.48M	≈ 0.48M	≈ 0.48M	≈ 0.48M	≈ 0.48M	≈ 0.48M	≈ 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	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M
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

[33], PAN [97], QQP⁸ [71], Twitter-URL [80], and PARADE [60], accuracy/F1 scores are reported; except for PAWS-wiki [176], I report accuracy/AUC scores complying with the metric presented in the original paper. Accuracy scores are reported for two NLI tasks: SICK-E [98] and SciTail [77]. Pearson/Spearman correlations are reported for two STS tasks: SICK-R [98] and STS-B [16].

The “default thresholds” are namely the thresholds shown in Table 3.1, which are determined by the characteristics of the training dataset of MRPC [33]. The “adjusted thresholds” indicate that I 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 *j-dist* are adjusted to 8 and 0.8 respectively for PARADE [60] 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 [77] whose sentence pairs are labeled with entails or neutral.

Classifier settings & adapter-BERT implementation: For small task datasets like MRPC containing less than 10k sentence pairs, I make attempts on both the SVM⁵ 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 my measurement to tune classifiers. When tuning the SVM, I 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, I use the widely-adopted hyperparameters for fine-tuning BERT [32]; the small

⁸Due to the unavailable labels of test dataset constrained by the policy of GLUE benchmark [154], the task performance of QQP is reported on the development dataset.

difference is that my 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. My 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 my open-source code⁷). As reflected in Yin et al. [169]’s work, seven linguistic features like the number of hypernyms in a sentence pair are particularly useful for NLI tasks. I append them to my 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, I decide to use the simplest classifier setting to test the effectiveness of PEFBAT for a pure academic demonstration.

I use bert-for-tf2⁹ implementation for adapter-BERT. In light of adapter’s performance on GLUE benchmark [154], I choose the adapter size 64 based on BERT-base ($\approx$ 3M trainable parameters), and follow the recommended hyperparameters presented in the original paper. For each task, I perform fine-tuning with four different training epochs {3, 10, 20, 50}, and report the one that achieves optimal performance. An exception is QQP [71], because 3 epochs are sufficient to process its large size of data.

Discussion on performance: I discuss my 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 [60] 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 is 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 [176] 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

⁹<https://github.com/kpe/bert-for-tf2>

learning, tasks from customers definitely vary to some extent — some resemble PARADE while some not.

The adjusted thresholds can improve corresponding tasks to some extent but not significantly, which reflects the fact that my latent spaces pre-trained with default thresholds are already useful for various tasks, and not only limited to MRPC [33]. This is further validated 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 I recommend it for practical use. To make the reuse of PEFBAT convenient, I have coded programming interfaces including the function of adjusting thresholds, which can be found in my open-source code⁷.

Table 3.9 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 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
adapter-BERT	<u>74.3/83.0</u>	94.2/94.1	84.7/79.6	<u>87.3/72.2</u>	<u>57.7/58.4</u>	<u>71.0/70.4</u>	<u>58.9/53.5</u>	<u>69.1/65.2</u>	<u>56.9</u>	88.6
Trainable parameters	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M	≈ 3M
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

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 are substantially light. As shown in Table 3.8, 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 [141]. It is obvious that tuning adapter-BERT [65] on the small tasks is more expensive than fine-tuning original BERT [32]. 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, I perform additional experiments, and the results are presented in Table 3.9. 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.

3.3.3 Analysis of Discriminative Similarity

Two experimental results are quite surprising to us. Firstly, PARADE [60] 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) [64] but also BERT-large (73.6/70.9) [32] and SciBERT (74.1/72.3) [5] in terms of accuracy. Secondly, PAWS [176] is created to measure models' sensitivity to word order and syntactic structure. Supposedly, my performance level should not be satisfactory, but when compared to the experimental results reported in the 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, I perform statistical analysis to examine the effectiveness of my discriminative similarity, namely the features 5-6 in Table 3.4. 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 my discriminative similarity enhance the ability of differentiation? The visualization to illustrate the analysis is presented in Figures 3.12, 3.13 and 3.14.

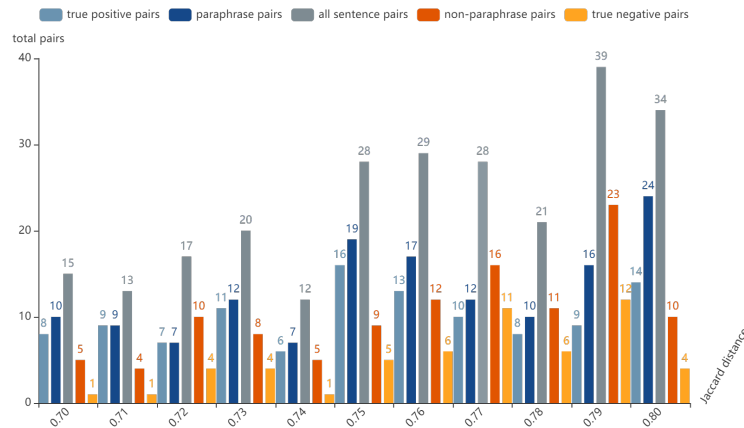


Fig. 3.12 Sentence-pair distributions (predicted test dataset of PARADE) at each level of Jaccard distance from 0.70 to 0.80.

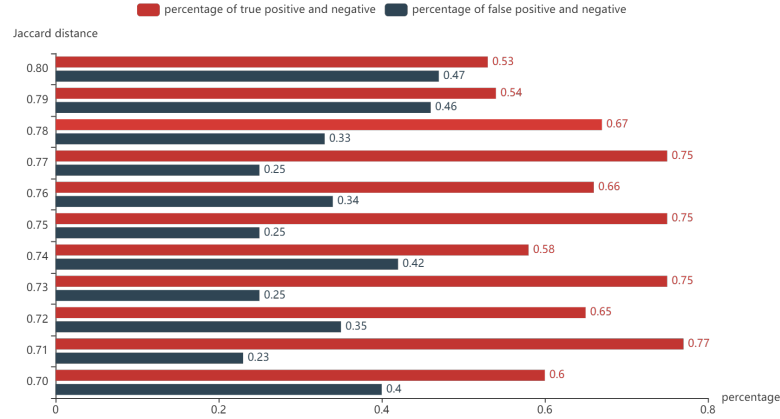


Fig. 3.13 Accuracies of paraphrase identification (predicted test dataset of PARADE) at each level of Jaccard distance from 0.70 to 0.80.

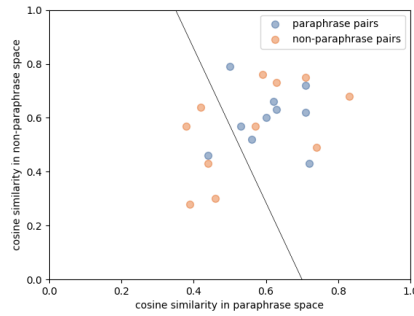


Fig. 3.14 Cosine similarities in (non)paraphrase space for sentence pairs (PARADE test dataset) at Jaccard distance 0.78.

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 3.12, I 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 3.13) 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 3.14 corresponding to the features 5-6 in Table 3.4. In Figure 3.14, 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 3.12. 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. I believe that it is my design of the features 7-12 (explained in Section 3.2.4) that augments the features 5-6.

To respond to what I mentioned in Section 3.2.4, my 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 [97] (see Figure 2.1). Both types of features are mutually complementary to each other.

3.4 Conclusion

For PI task, I 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 3.1 and 3.2 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 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 3.3, I 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. I also performed the ablation study and technical analysis to help further understand the mechanism of PEFBAT.

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, I 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 [33], 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 [112]’s work that fastText [107] 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 NLTK⁴ can be implemented using HMMs (Hidden Markov Models)¹⁰, 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.

¹⁰<http://www.nltk.org/api/nltk.tag.html?highlight=hmm>

Chapter 4

Representation Fusion Through Structural Connection

In this chapter, I describe my fusing approach by structurally connecting factorized distributional representations with word embeddings of language models. This approach, with the title “DG Embeddings: The unsupervised definition embeddings learned from dictionary and glossary to gloss context words of Cloze task” [93], has been accepted by Knowledge-Based Systems journal. It contributes to the better acquisition of vocabulary for language models, which is directly reflected by improved word embeddings suffering less from anisotropy, improved various downstream tasks, and more accurate attention scores for collocations of phrases.

4.1 Introduction

To acquire vocabulary, it is effective to learn words from context while using dictionaries as an auxiliary tool. For humans, it has been shown in the previous linguistic studies [79, 1, 74, 158] that using the auxiliary tool on either target words to be learned from context (target words-glossed) or words comprising context (context words-glossed) is an effective approach. In Natural Language Processing (NLP) for machines, previous studies are mainly focused on the former, and thus context words-glossed explorations remain sparse. In this paper, I investigate the latter by proposing a simple yet effective approach to infusing context words with semantic features of glosses for the MLM objective of BERT [31], which is considered as the Cloze task [144] for machines to learn words from bidirectional context words.

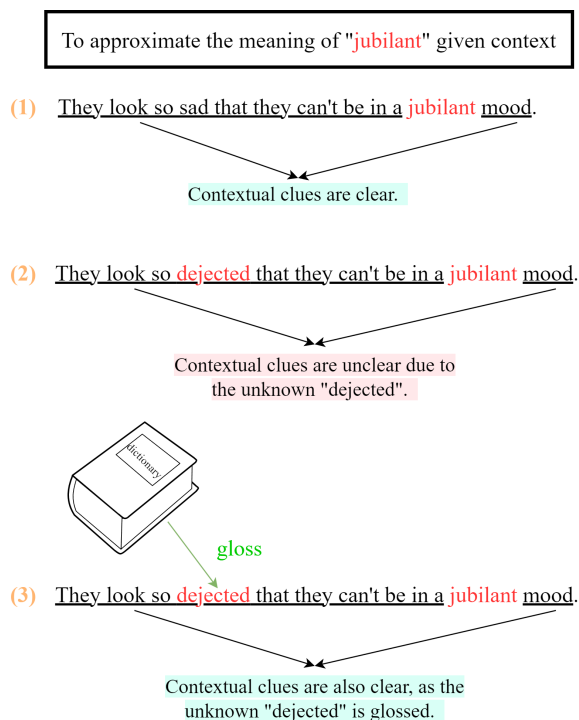


Fig. 4.1 The approximation to the meaning of “jubilant” hinges on whether contextual clues are clear or not. If not, dictionaries are used to gloss unclear parts.

4.1.1 Context Words-Glossed Pre-training

It is a common linguistic phenomenon that when humans try to approximate the meaning of an unknown word given its context, a good approximation is difficult to reach if some words in the context are also unknown. The phenomenon suggests that it is a vital prerequisite to understand contextual clues. An example is demonstrated in Figure 4.1. As shown in the previous linguistic studies [1, 74], language learners tend to acquire vocabulary better after reading comprehension (incorporating the task of guessing word sense from context) if some unintelligible context words (e.g., domain-specific terminologies, idiomatic expressions, etc.) are glossed.

To the best of my knowledge as of writing, the only previous NLP study that can be barely counted as context words-glossed pre-training is SenseBERT [87]. It infuses both context and target words with supersense (e.g., chocolate belongs to noun.food). During pre-training, apart from the MLM objective for predicting target words, the objective of classifying supersense of target words is computed simultaneously. Stringently, supersense (i.e., semantic category) cannot be deemed as semantic information for glosses, which are normally represented by definitions or synonyms.

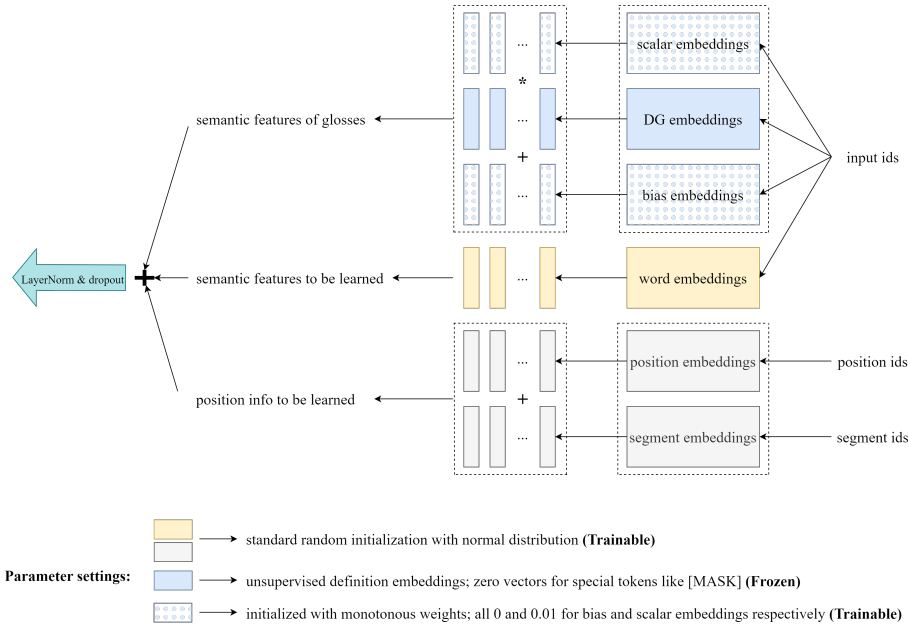


Fig. 4.2 The embedding layer of BERT is shown for my proposed approach.

Inspired by the solution to the linguistic phenomenon, I am motivated to investigate the potentiality of context words-glossed pre-training for the MLM objective of BERT [31]. Unlike the approaches for target words-glossed pre-training mentioned in the previous subsection, my strategy is focused on the embedding layer, which is illustrated in Figure 4.2. Except the position and segment embeddings, the roles of the embeddings related to semantic information are described as follows.

Word embeddings: They are the regular embeddings whose weight is tied to the decoder [69, 121] for predicting target words. In HuggingFace [160] implementation of BERT that I use, its weight is initialized with normal distribution of mean 0.0 and standard deviation 0.02¹. During pre-training², semantic features are optimized and stored in these embeddings by learning words from context via self-attention mechanism [150].

DG embeddings: To infuse context words with semantic features of glosses, I propose DG embeddings to serve as auxiliary embeddings — the unsupervised definition embeddings learned from dictionaries and glossaries. My idea is to simulate the scenario that BERT is holding a dictionary for context words while doing the Cloze task. Thus, like dictionary entries, words and tokens covered in DG embeddings should convey multiple senses. For example, the DG embedding of “cake” consists of literal

¹This is the link to HuggingFace BERT modeling.

²For clarity, the NSP pre-training objective is omitted, where the quality of sentence-level predictions should also depend on the quality of acquired vocabulary.

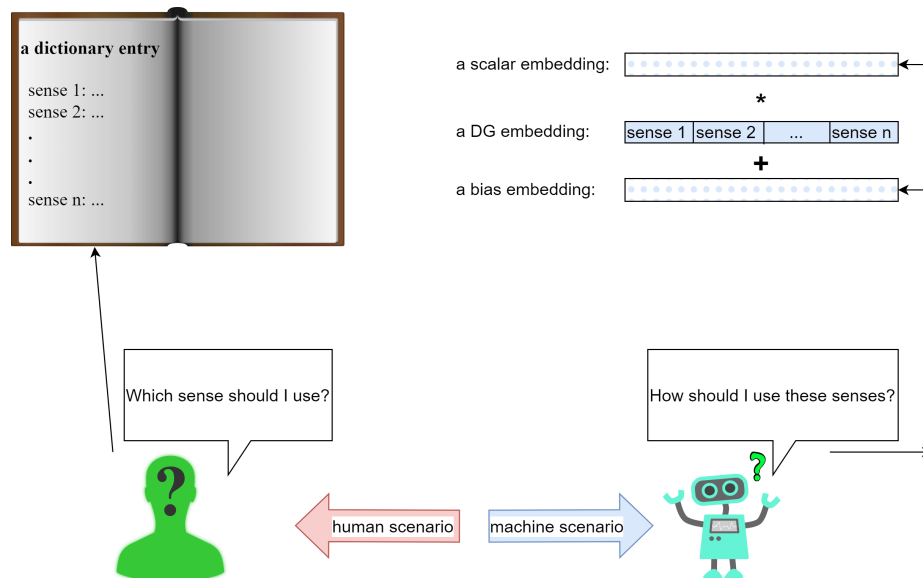


Fig. 4.3 Let machines learn how to use the senses represented in DG embeddings.

meaning from the word and figurative meaning from “a piece of cake” (reflected by a certain dimensionality pattern in its embedding vector representation). For tokens like “##ant”, its DG embedding is composed of semantics from “jubilant” (ju, ##bil, ##ant), “intolerant” (into, ##ler, ##ant), and tokens in phrases if exist. Zero vectors are generated for special tokens like “[MASK]”. This is desirable because in the context words-glossed setting, target words should not be glossed.

Scalar and bias embeddings: For context words or tokens in different input sequences, to allow machines to learn how to use the senses represented in DG embeddings, the scalar and bias embeddings are created for element-wise multiplication and addition (the idea is shown in Figure 4.3). This is consistent with the human practice of looking up words in dictionaries. When I encounter polysemy, which sense of a word is useful for us is case by case, and should be chosen by ourselves. Through pre-training, two embeddings learn a general distribution of their dimensional values. As to which value should be amplified or diminished more, it can be adjusted during fine-tuning for different tasks. Since the biggest absolute value in the weight matrix of DG embeddings created using my method is 2.72, to stay similar magnitude as the weight of word embeddings, the weight initialization of scalar embeddings is filled with 0.01.

Through my approach, all context words of any input sequence can obtain DG embeddings by their input ids. Although it is viable to follow two previous works [161, 172] by setting a static or dynamic threshold of word frequency in corpora to highlight rare words

only, any word can be potentially rare in context words-glossed pre-training. For example, almost every idiomatic expression consists of active vocabulary (e.g., “clean house”, “a piece of cake”, etc.) that tend to have high frequency in any corpus, but they can be regarded as rare words due to their metaphorical attribute [21]. If DG embeddings are used to gloss them only when their figurative senses occur, there will be a concern that scalar and bias embeddings are learned to completely lean towards their figurative semantics.

4.1.2 Unsupervised Definition Embeddings

To employ unsupervised learning for DG embeddings is based on three considerations.

First, based on the distributional hypothesis [57, 44], by utilizing any probabilistic auto-encoding model with self-supervised learning such as early CBOW and skip-gram of word2vec [106] or the transformer encoder [150] of BERT nowadays, learning words from context is applicable for machines. As DG embeddings used in my study is an auxiliary tool to assist the learning process, a technical duplicate will occur if the creation is related to those models with self-supervised learning. For example, if I use previous definition embeddings like CPAE [13] (a post-processed word2vec), dict2vec [146] (adding regularization to the skip-gram objective of word2vec), or the recent DictBERT [19] (based on pre-trained BERT), the learning process will be “a model created for learning words from context” → “further learning word senses with definitions based on the model” → “using the learned word sense to assist a new model in learning words from context”. This clearly suggests that the auxiliary tool for learning words from context originates from learning words from context. Another neural language model that can embed words with their definitions is tailored for reverse dictionaries [62], which is not felicitous for my study. I need to underline that this consideration is based on appropriateness rather than quality.

Second, DG embeddings as an auxiliary tool are not meant for independent existence — they are not created to address downstream tasks by means of feature-based transfer [167, 92]. Basic semantic relatedness of words and phrases is attainable via unsupervised learning based on the bag-of-words hypothesis [132], which is computationally inexpensive and therefore beneficial to future expansion. Creating definition embeddings based on the bag-of-words hypothesis is also in accordance with a real-world scenario of dictionary use: headword search in thesauruses like Longman Language Activator [113]. An example is listed below given the headword “strange”.

- quirky → (of an aspect of somebody’s personality or behaviour) a little strange
- grotesque → strange in a way that is unpleasant or offensive

- eerie → strange, mysterious and frightening

These dictionary entries are mainly defined by multiple underlined key features, where the headword is the key feature that they have in common while their nuance is determined by other key features. The headword recognized as active vocabulary is self-explanatory, which belongs to a linguistic property called reflexivity [102]. For humans, it seems that this approach is only feasible for thesauruses with conceptual categories. For machines, I devise an algorithm to extract key features (incorporating polysemy) for all dictionary entries (see Section 4.2.1). Extracted key features for each entry are represented by bag-of-words — for machines to process semantic relatedness of entries via key features, word order does not matter.

Third, unsupervised learning can allow phrases to have independent vector representations, as there is no need of one-hot encodings for supervisory signals. This is helpful for the following operation. Suppose the embedding for the phrase “a piece of cake” is $\vec{v}_1$, and the embedding for the word “cake” is $\vec{v}_2$. Then a combined embedding containing both literal and figurative semantics can be generated for the word “cake” by dint of $\frac{1}{2}(\frac{1}{4}\vec{v}_1 + \vec{v}_2)$ (similar for the words “a”, “piece”, and “of”). This operation can considerably preserve internal similarity of phrases. Further details are provided in Sections 4.2.2 and 4.2.3.

4.1.3 Research Goals & Contributions

I apply my approach to uncased BERT-case for an evaluation, and refer to my model as DG-BERT. To validate whether machines can achieve better acquisition of vocabulary via learning words from glossed context, the primary research goal is to observe if **fast pre-training convergence** for DG-BERT can be accomplished. An indication of this is the fast and effective acquisition of vocabulary for machines. Subsequently, I will further investigate the effectiveness by testing if **good understanding of literal and figurative semantics** can be achieved by DG-BERT. For literal semantics, standard downstream tasks are used for a test. For figurative semantics, I apply DG-BERT as a backbone model to MeBERT [21] for a test. In addition, as I mentioned in the previous subsection that DG embeddings can preserve internal similarities of phrases to a large extent, it will be investigated whether DG-BERT can absorb critical semantic features from DG embeddings so that **sensitive target-word predictions in the collocation of phrases** and **accurate self-attention for phrases in input sequences** are attainable. My evaluation criterion is to compare DG-BERT with my vanilla BERT: both models are pre-trained under the exact same condition. All details and empirical evidences are provided in Section 4.3.

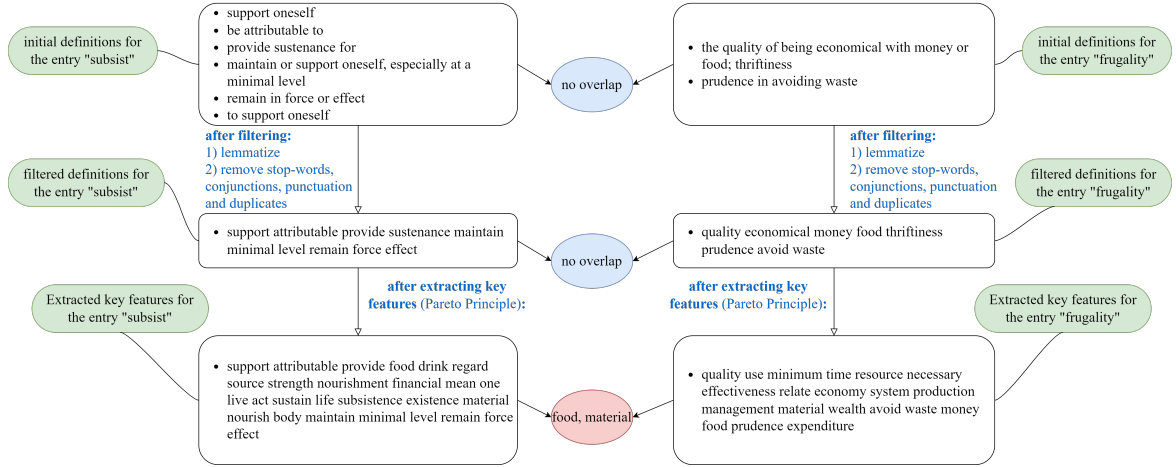


Fig. 4.4 The flow of definition pre-processing is exemplified by the first entry “subsist” and its semantically related entry “frugality”.

Definitions from dictionaries and glossaries are the external semantic knowledge that I draw on for my research purpose. The accomplishment of my research goals benefits from the utilization of the knowledge, which leads to the contributions of this paper enumerated as follows.

1. I discover that by glossing context words to predict target words for auto-encoding models, a better acquisition of vocabulary can be achieved by machines.
2. I show that unsupervised definition embeddings based on the bag-of-words hypothesis can help language modeling without using any cutting-edge technologies and can lead to computationally inexpensive future expansion.

4.2 DG Embeddings

The creation of DG embeddings is not complex, which involves two critical steps: extraction of key features in Section 4.2.1 and adaption to the wordpiece vocabulary of BERT in Section 4.2.2. I present an evaluation criterion in Section 4.2.3, which is also beneficial for future expansion and quality improvement.

For unsupervised learning based on the bag-of-words hypothesis, the term-document matrix of VSMs [148] is employed to create DG embeddings. Documents are indicated by dictionary and glossary entries (words and phrases) while terms are represented by words as features comprising definitions. The dictionary resources are released by a previous work [70], including Oxford, Wiktionary, and WordNet [109]. I found one glossary resource

provided by PARADE dataset [59], which contains computer science terminologies. After assembling all resources, there are 305,518 unique entries containing 189,137 phrases.

Algorithm 1 Extraction of Key Features

```

 $X \leftarrow \text{entry\_definition\_map}$ 
 $Y \leftarrow \text{top\_20\%\_key\_feature\_list}$ 
 $Z \leftarrow \text{word\_is\_identified\_as\_key\_feature}$ 
 $K \leftarrow \text{already\_identified\_word\_list}$ 
procedure IDENTIFY( $word, X, Y, Z, K$ )
  if  $word \in K$  then
    return
  end if
   $K.add(word)$ 
  if  $word \in Y$  and  $word \notin Z$  then
     $Z.add(word)$ 
  else
     $definition = X[word]$ 
     $words = \text{word\_tokenize}(definition)$ 
    for  $word \in words$  do
      IDENTIFY( $word, X, Y, Z, K$ )
    end for
  end if
end procedure

```

4.2.1 Extraction of Key Features

As exemplified in Figure 4.4, the initial definition for each entry are concatenated by multiple definitions (including polysemy if exists) from different dictionary and glossary resources. All words are lemmatized first, then duplicates and unnecessary words are removed after filtering. As a result, each entry consists of multiple unique words as features, which make the term-document matrix comprise the values of 1 and 0 only. Unlike information retrieval [132] relying on distributional counts to determine semantic relatedness of documents, for dictionary entries introduced in Section 4.1.2, it should depend on how many key features they have in common. This is consistent with two previous works [165, 18] seeking to use the matrix for the generation of synonyms and antonyms. After filtering, the total unique features are 54,861; correspondingly, the matrix size is $305,518 \times 54,861$.

To extract key features, the concrete method is to employ Pareto Principle, also well known as 80/20 rule — roughly 80% of consequences come from 20% of causes. In my case, roughly 80% semantic relatedness are determined by top 20% words as key features. I count

the frequencies of those 54,861 words based on their occurrences in all definitions, and select 10,748 words of high frequency as top 20%. For the rest 44,113 words, I devise a recursive algorithm to search their definitions for replacement (see Algorithm 1 for details). As a result shown in Figure 4.4, the refined definition of “subsist” contains more key features like food, drink, material, etc. More robust semantic relatedness to other entries is achievable; e.g., overlapped features are generated between the definitions of “subsist” and “frugality” after this process. Meanwhile the matrix size is scaled down to $305,518 \times 10,748$. With more key features added to each entry and reduced dimensionality, the matrix become less sparse.

4.2.2 Matrix Factorization & Adaption

The size of wordpiece vocabulary and length of hidden states are 30,522 and 768 respectively in the HuggingFace uncased BERT-base [160]. Correspondingly, there are two steps to convert the matrix size from $305,518 \times 10,748$ to $30,522 \times 768$. The first step is to scale down the matrix from $305,518 \times 10,748$ to $305,518 \times 768$ by matrix factorization. As my matrix does not rely on different distributional counts to distinguish semantics, such as a centered matrix for computing covariance like PCA [99], I directly use the basic algorithm truncated SVD for matrix factorization.

SVD decomposes a matrix X into the product of three matrices UDV^T , where U and V are in orthogonal column form with unit length: $U^T U$ and $V^T V$ are identity matrices and D is a diagonal matrix of singular values. If X is of rank r , then D is also of rank r . Let D_k , where $k < r$, be the diagonal matrix formed from the top k ($k = 768$ in my case) singular values, and let U_k and V_k be the matrices produced by selecting the corresponding columns from U and V . Among all matrices $\hat{X} = U_k D_k V_k^T$ of rank k , to select one that best approximates the original matrix X , the Frobenius norm [52] $\|\hat{X} - X\|_F$ is used to minimize the approximation errors. After obtaining best approximated $\hat{X}$, the U_k in $\hat{X}$ is the factorized matrix with the size of $305,518 \times 768$ for my purpose. I address the latent vector representations in this matrix for all entries as unsupervised definition embeddings.

The next step to scale down the factorized matrix from $305,518 \times 768$ to $30,522 \times 768$ is called matrix adaption (see Algorithm 2). I seek to allow words or tokens in BERT vocabulary to encompass multi-senses. For example, suppose there are only two definition embeddings related to the word “cake”: the definition embedding $\vec{v}_1$ for “a piece of cake” and the definition embedding $\vec{v}_2$ for the word itself. Then according to Algorithm 2, the DG embedding generated for the word “cake” is $\frac{1}{2}(\frac{1}{4}\vec{v}_1 + \vec{v}_2)$. As each word or token contained in a phrase is an inseparable component to form the complete phrase, computationally, semantic features conveyed by the phrase should be equally shared ($\frac{1}{4}\vec{v}_1$ in the example). To avoid large dimensional values, the average is calculated for each adaption ($\frac{1}{2}$ in the example). Zero

Algorithm 2 Adaption to BERT Vocabulary

```

 $X \leftarrow \text{BERT\_vocabulary}$ 
 $Y \leftarrow \text{entry\_list}$ 
 $Z \leftarrow \text{definition\_embed}$ 
 $K \leftarrow \text{DG\_embed}$ 
procedure ADAPT( $X, Y, Z, K$ )
  for  $\text{BERT\_token} \in X$  do
     $\text{count} = 0$ 
     $\text{temp\_embed} = \vec{0}$ 
    for  $\text{entry} \in Y$  do
       $\text{entry\_tokens} = \text{BERT\_tokenizer}(\text{entry})$ 
       $\text{length} = \text{len}(\text{entry\_tokens})$ 
      if  $\text{BERT\_token} \in \text{entry\_tokens}$  then
         $\text{temp\_embed} = \text{temp\_embed} + \frac{1}{\text{length}} \cdot Z(\text{entry})$ 
         $\text{count} = \text{count} + 1$ 
      end if
    end for
    if  $\text{count} \neq 0$  then
       $\text{temp\_embed} = \frac{1}{\text{count}} \cdot \text{temp\_embed}$ 
       $K.\text{add}(\text{temp\_embed})$ 
    else
       $K.\text{add}(\text{temp\_embed})$ 
    end if
  end for
end procedure

```

vectors are generated for 5,940 special tokens such as “[MASK]” and foreign vocabulary like Chinese characters. In this paper, I focus on a monolingual scenario, and treat foreign vocabulary as special tokens. As for “[MASK]”, even if there existed a definition embedding for it, I would also modify it to a zero vector, because it represent target words and in context words-glossed pre-training, target words should not be glossed. The matrix after adaption with the size of $30,522 \times 768$ is the final DG embeddings.

4.2.3 Evaluation of DG Embeddings

DG embeddings are evaluated at word and phrase levels, plus the potentiality of their vector space to assist the one of word embeddings (see the last paragraph of this section).

For evaluation at word and phrase levels, the evaluation criterion is considered for to what degree DG embeddings can preserve semantic features after matrix adaption. In the hypothetical example mentioned in the previous subsection, the DG embedding generated for

the word “cake” is $\frac{1}{2}(\frac{1}{4}\vec{v}_1 + \vec{v}_2)$. Although multi-senses exist in the DG embedding, this can decrease its cosine similarity to other literally close words (e.g., “bread”). Also for the phrase “a piece of cake”, the internal similarity (see Equations 4.1 and 4.2, where “wt” denotes word or token) of this phrase before matrix adaption is 100% ($\frac{1}{4}\vec{v}_1$ for each word), but after adaption, the similarity will inevitably decrease, because the DG embeddings for each word are composed of multiple definition embeddings.

$$phrase = \{wt_0, wt_1, \dots, wt_n\} \quad (4.1)$$

$$InternalSim(phrase) = \frac{1}{n+1} \sum_{i=0}^n CosineSim(embed(wt_i), \sum_{\substack{j=0 \\ j \neq i}}^n embed(wt_j)) \quad (4.2)$$

For the evaluation at the word level, I choose the TOEFL synonym selection task³ [84] for a straightforward comparison. An example is “enormously $\rightarrow$ {appropriately, uniquely, tremendously, decidedly}”, where “tremendously” is the semantically closest word among four candidates. The dataset contains 80 such questions. When a word consists of tokens, the addition of their embeddings form a complete embedding (e.g., $embed(pan) + embed(##cake)$ to represent $embed(pancake)$). The highest cosine similarity between embeddings is used for decision. For the evaluation at the phrase level, the internal similarities are calculated based on Equations 4.1 and 4.2 for all 189,137 phrases. I compare DG embeddings with another two embeddings: the word embeddings from pre-trained HuggingFace uncased BERT-base [160]; the embeddings whose weight is initialized by normal distribution with mean 0 and standard deviation 0.02 (introduced in Section 4.1.1) as my baseline.

The results are presented in Tables 4.1 and 4.2. There is a prerequisite information for the evaluation at the word level: when the matrix size is $305,518 \times 10,748$, the performance for TOEFL task is 77.2; after matrix factorization scaled down to $305,518 \times 768$, the performance is 74.7 due to inevitable information loss. The performance is based on total 79 samples, as there is one out-of-vocabulary sample, which are also the performance of three embeddings based on. Therefore, the upper bounds for DG embeddings at word and phrase levels are 74.7 and 100.0 (average internal similarities) respectively. As shown in both tables,

³The task dataset should be requested via a portal, which is available at [https://aclweb.org/aclwiki/TOEFL_Synonym_Questions_\(State_of_the_art\)](https://aclweb.org/aclwiki/TOEFL_Synonym_Questions_(State_of_the_art)). The samples in this dataset might change with time. For example, the sample “rug $\rightarrow$ {sofa, ottoman, carpet, hallway}” mentioned in Faruqui et al. [40]’s work does not exist in the dataset that I have obtained.

Table 4.1 Three types of embeddings are evaluated at the word and phrase levels. “BERT word embeddings” is the word embeddings from pre-trained HuggingFace uncased BERT-base [160]. The weight of “Random word embeddings” is initialized by normal distribution with mean 0 and standard deviation 0.02.

Embeddings	TOEFL task (%)	Average internal similarities of phrases (%)
Random word embeddings	22.8	0.4
BERT word embeddings	76.0	39.1
DG embeddings	65.8	59.6

although the performance naturally drops, the overall performance is comparable to BERT word embeddings, considering $(65.8 + 59.6) > (76.0 + 39.1)$. Especially at the phrase level, DG embeddings account for bigger percentile in high intervals. Since DG embeddings are outperformed at the word level, it can be worried that DG embeddings can assist context words of BERT at the early stage of pre-training, but they might inflict side effect at the late stage. I will empirically show in Sections 4.3.3 and 4.4 that there is no side effect.

I use an extreme but easy to understand example to further explain the performance at word and phrase levels. Suppose there are only five definition embeddings: $\vec{v}_1$ for “a piece of cake”; $\vec{v}_2$ for “a”; $\vec{v}_3$ for “piece”; ; $\vec{v}_4$ for “of”; ; $\vec{v}_5$ for “cake”. Then according to Algorithm 2, the DG embeddings generated for these tokens are itemized as follows.

- $\text{DG_embed}(\text{“a”}) = \frac{1}{2}(\frac{1}{4}\vec{v}_1 + \vec{v}_2)$
- $\text{DG_embed}(\text{“piece”}) = \frac{1}{2}(\frac{1}{4}\vec{v}_1 + \vec{v}_3)$
- $\text{DG_embed}(\text{“of”}) = \frac{1}{2}(\frac{1}{4}\vec{v}_1 + \vec{v}_4)$
- $\text{DG_embed}(\text{“cake”}) = \frac{1}{2}(\frac{1}{4}\vec{v}_1 + \vec{v}_5)$

The internal similarity of “a piece of cake” is high because $\frac{1}{4}\vec{v}_1$ is merged into their DG embeddings. On the other hand, as multi-senses are incorporated into DG embeddings for each token, their performance at word level naturally drops. However, this is what I seek to achieve — DG embedding for each token encompasses multi-senses, and allow scalar and bias embeddings to learn which sense to use during pre-training (illustrated in Figure 4.3). Actually, the decrease within 10% (from 74.7 to 65.8 described in the previous paragraph) is acceptable for the trade-off between gaining multi-senses and losing performance on single sense.

Table 4.2 The percentage in each interval of internal similarities are scored by three types of embeddings. “BERT word embeddings” is the word embeddings from pre-trained Hugging-Face uncased BERT-base [160]. The weight of “Random word embeddings” is initialized by normal distribution with mean 0 and standard deviation 0.02.

Intervals	Random word embeddings (%)	BERT word embeddings (%)	DG embeddings (%)
(0.0, 0.1]	49.4	0.2	0.5
(0.1, 0.2]	1.1	2.8	2.5
(0.2, 0.3]	0.3	15.1	5.4
(0.3, 0.4]	0.1	32.4	8.9
(0.4, 0.5]	0.1	36.9	12.5
(0.5, 0.6]	0.0	12.3	16.6
(0.6, 0.7]	0.0	0.3	19.2
(0.7, 0.8]	0.0	0.0	18.8
(0.8, 0.9]	0.0	0.0	13.3
(0.9, 1.0]	0.0	0.0	2.3

I also make an attempt on the matrix without key feature extraction, namely the matrix with the size of $305,518 \times 54,861$ (by filtering process only in Figure 4.4). The performance on TOEFL task is the same (77.2), and decreases to 73.4 with the size of $305,518 \times 768$ after matrix factorization. However, after matrix adaption, the performance on TOEFL task increases to 69.6, while the internal similarities of phrases decrease to 43.4. As a result, the overall performance is lower than one with key feature extraction: $(69.6 + 43.4) < (65.8 + 59.6)$. The result further validates the effectiveness of key feature extraction.

Another way to evaluate DG embeddings is to see if it can assist word embeddings in learning better semantics. The crucial criterion is to examine whether the anisotropic vector space [48, 37] of word embeddings can be alleviated. As shown in Figure 4.5, although the distribution of DG embeddings is not directionally uniform, the aperture size of the cone is noticeably larger than the one of word embeddings (the comparison demonstrated in the 3rd sub-figure). As DG embeddings are structurally connected with word embeddings at the embedding layer, extrapolating from the comparison, it can be expected that the distribution of word embeddings in the enclosed area of two lines can become more dense after pre-training, which indicates the enlarged aperture size for word embeddings. The corresponding evidence will be provided in Section 4.4.

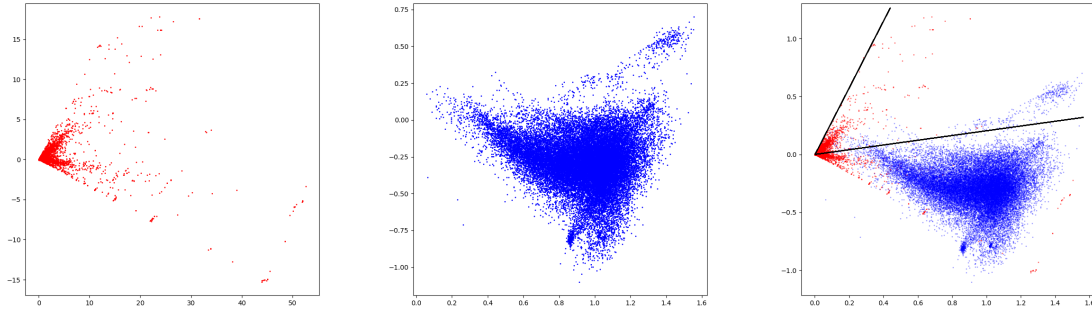


Fig. 4.5 SVD visualization of embeddings from left to right → **1st**: DG embeddings; **2nd**: word embeddings of Huggingface BERT-base-uncased; **3rd**: comparison between **1st** and **2nd**. In **3rd**, the magnitude of DG embeddings is scaled down to a similar magnitude of the word embeddings.

4.3 DG-BERT

In this section, I experiment with my vanilla BERT (uncased BERT-base) and DG-BERT (uncased BERT-base with DG, scalar and bias embeddings). For a fair comparison, both models are pre-trained on *the same pre-processed text files with the exact same masked tokens*⁴ and hyperparameters.

4.3.1 Pre-training & Experimental Settings

I follow the original hyperparameter settings [31] to pre-train my vanilla BERT and DG-BERT, which are listed in Appendix A.1. The loss is also computed by the addition of NSP and MLM pre-training objectives. Three settings — max-seq-length 448, no 10% random tokens to replace [MASK], and pre-training corpus — slightly vary from the origin due to three special reasons listed below.

1. My pre-training server — its RAM (for loading pre-processed text files) and total GPU memory (for batch size 256) — physically cannot hold max-seq-length 512. Due to the physical constraint, I have to scale the max length from 512 down to 448.
2. Among 15% randomly-selected masked tokens, the distribution proposed by Devlin et al. [31] is 80% for [MASK], 10% to remain original tokens, and 10% to be replaced

⁴I perform three basic steps for text file pre-processing: remove HTML tags, concatenate short sentences, and convert texts to input format of BERT model. I saved the pre-processed text files (a Python dictionary of input_ids, attention_masks, segment_ids, MLM_labels and NSP_labels) to a Pytorch file. It can be loaded anytime for pre-training both models.

by random tokens. In my case, the distribution is 80%, 20%, and 0%. As random tokens (in most cases except special tokens) also can obtain their non-zero-vector DG embeddings via their input ids, there is a concern that corresponding scalar and bias embeddings will be improperly optimized if I keep using random tokens.

3. As stated in the GitHub page of BERT, BookCorpus [179] is not available for public download, I use OpenWebText [51] and Wikipedia dumps. Due to the physical constraint mentioned above, text files with the size of roughly 7 GB (3.5 GB from each) are used for pre-training. The corpus size is comparatively small in modern language model pre-training, but it is an affordable size for pre-training in academic environment.

My model implementation is based on HuggingFace transformer [160], and coding multi-GPU pre-training with mixed precision (fp16) [104] is based on Accelerate [53].

In the following subsections, a variety of experiments are conducted to evaluate DG-BERT from different perspectives. For those experiments where fine-tuning is involved, two points need to be underlined: 1) existing fine-tuning techniques are not utilized; 2) each task is run with five different random seeds⁵, and the average of five runs is reported based on optimal hyperparameter settings (details in Appendix A.1).

In this section, I consistently keep the comparison between DG-BERT and my vanilla BERT pre-trained under the exact same condition. I leave the comparison with the most related work in Section 4.4, where I further explain the intrinsic strengths of my approach based on this section.

4.3.2 Pre-training Convergence

I first show that DG-BERT converges faster than my vanilla BERT in pre-training, which is the indication of acquiring vocabulary in a fast and effective manner. From my understanding, the pre-training process can be generally divided into three phases, and the corresponding pre-training losses for convergence are summarized in Figures 4.6 and 4.7.

The first phase happens in the first epoch. Until the end of this phase, as DG-BERT has not seen all input sequences, scalar and bias embeddings are still under-fitting. Accordingly, what DG embeddings can provide for context words is overall semantic features of glosses. Nevertheless, the pre-training convergence is generally faster than vanilla BERT (see the left sub-figure of Figure 4.6).

The second phase is the refining process of scalar and bias embeddings. After the first epoch seeing all input sequences, scalar and bias embeddings gradually refine their

⁵I leave random seed generation to the nondeterministic behavior of PyTorch [115].

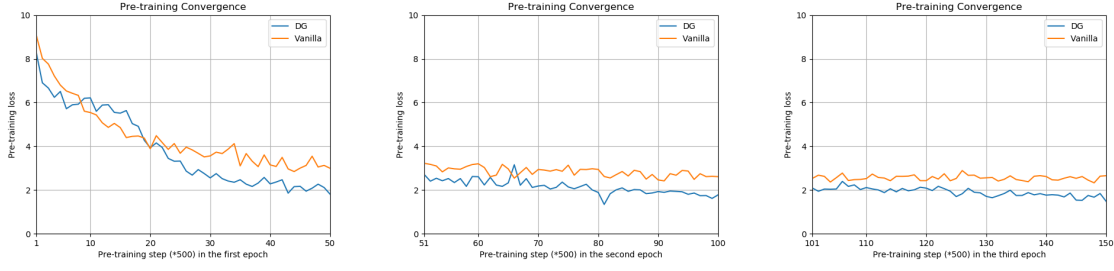


Fig. 4.6 **Left:** Every 500-step batch loss is shown for the first epoch; **Middle:** Every 500-step batch loss is shown for the second epoch.; **Right:** Every 500-step batch loss is shown for the third epoch.

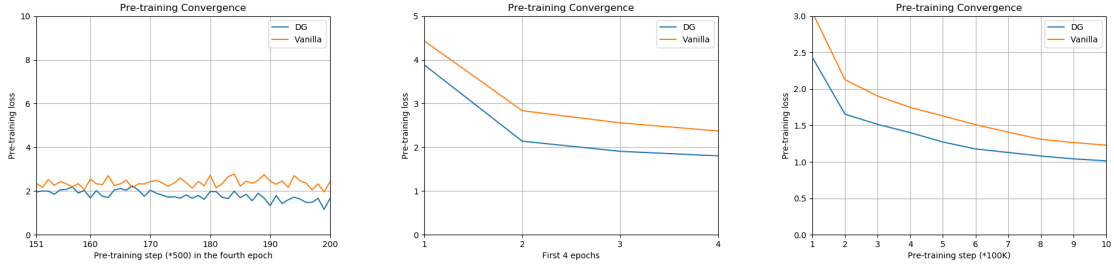


Fig. 4.7 **Left:** Every 500-step batch loss is shown for the fourth epoch; **Middle:** The convergence is shown for the first 4 epochs contained in the first 100K pre-training steps. Four loss values are the average of all 500-step batch losses in each epoch; **Right:** The convergence is shown for every 100K pre-training steps. Ten loss values are the average of all 500-step batch losses in each 100K step.

dimensional values to learn how to use dimensionality patterns of DG embeddings. In this phase, the fast pre-training convergence is more obvious than the first phase, because the semantic features provided by DG embeddings for context words are more rationally adjusted by scalar and bias embeddings, and further reinforced by the semantic features of word embeddings learned through predicting target words. As shown from the middle and right sub-figures of Figure 4.6 to the left sub-figure of Figure 4.7, the loss is consistently lower than the vanilla BERT (except a noticeable outlier in the second epoch). The summary of the first 4 epochs (namely the first 100K pre-training steps) is illustrated in the middle sub-figure of Figure 4.7. As shown in this figure, the loss gap becomes larger from the second epoch on, and keeps the trend.

The third phase is where the refinement ends and marginal adjustment begins. It can be inferred that this phase starts from the 600K pre-training step shown in the right sub-figure of Figure 4.7. The loss of pre-training DG-BERT drops not as significantly as previous steps,

Table 4.3 A preliminary test is performed for target-word predictions in collocations of two common idiomatic expressions. “DG-BERT w/o DG” denotes the whole package of DG, scalar and bias embeddings is removed from pre-trained DG-BERT.

Input sequences	Top 5 predictions & softmax scores	Models
[CLS] They left on the [MASK] of the moment. [SEP]	spur, heat, beat, power, scale 0.9817, 0.0052, 0.0021, 0.0007, 0.0006	DG-BERT
	spur, heat, beat, power, edge 0.9587, 0.0142, 0.0034, 0.0022, 0.0011	DG-BERT w/o DG
	spur, heat, eve, basis, face 0.5943, 0.0358, 0.0267, 0.0250, 0.0181	Vanilla BERT
[CLS] The task is a piece of [MASK]. [SEP]	cake, art, work, paper, history 0.3945, 0.2898, 0.0607, 0.0305, 0.0256	DG-BERT
	cake, art, work, paper, software 0.2623, 0.1981, 0.1141, 0.0423, 0.0314	DG-BERT w/o DG
	paper, work, cake, machinery, wood 0.1550, 0.0528, 0.0449, 0.0317, 0.0307	Vanilla BERT

and the loss gap becomes gradually smaller. However, the gap is still noticeable, as the final loss of the vanilla BERT is somewhere between 500K and 600K of DG-BERT.

4.3.3 Preliminary Tests

I perform a preliminary test for target-word predictions in the collocation of two common idiomatic expressions (see Table 4.3). Since both DG-BERT and my vanilla BERT are pre-trained on the same conditions (emphasized at the beginning of Section 4.3), the word embeddings of DG-BERT are optimized better than the ones of my vanilla BERT. As shown in “DG-BERT w/o DG”, although the softmax scores for two target words “spur” and “cake” are lower than the ones of DG-BERT (0.9587 to 0.9817 and 0.2623 to 0.3945), they are substantially higher than the ones of my vanilla BERT (0.9587 to 0.5943 and 0.2623 to 0.0449). Later in Section 4.3.6, I present experimental results on a published dataset for a further evaluation.

I also perform the TOEFL test of Section 4.2.3 for three types of embeddings: word embeddings of DG-BERT; word embeddings plus scalar, bias and DG embeddings of DG-BERT; word embeddings of my vanilla BERT. The results are 74.7, 77.2, and 73.4 respectively, where the word embeddings of DG-BERT (74.7) is better than the one of my vanilla BERT (73.4). From 74.7 increased to 77.2, it can be concluded that although DG embeddings are moderate on the performance at the word level (65.8 for the TOEFL test), they do not stifle the optimization of word embeddings in pre-training but rather assist and improve it. This is also validated by the faster pre-training convergence presented in this subsection. Furthermore, the better performance of DG embeddings at the phrase level

Table 4.4 The experimental results of GLUE benchmark are reported on **validation set**. The result for MNLI is reported on the average of match and mismatch.

Models	MNLI (Acc.)	QQP (Acc.)	QNLI (Acc.)	SST2 (Acc.)	COLA (Matthews)	STSB (Pearson)	MRPC (Acc.)	RTE (Acc.)	Avg.
Vanilla BERT	83.92	90.90	91.20	91.92	58.50	89.59	86.76	69.34	82.77
DG-BERT	85.09	91.13	91.40	93.00	60.14	90.07	88.09	73.94	84.11

Table 4.5 The experimental results are reported on **validation set** for SQuAD, CoNLL2003, and SWAG tasks. The results of CoNLL2003 are based on label-first-token.

Models	SQuAD (EM/F1)	SQuAD2.0 (EM/F1)	CoNLL2003 (F1)	SWAG (Acc.)
Vanilla BERT	80.47/87.76	72.26/75.59	94.50	72.46
DG-BERT	82.81/89.80	74.63/77.75	94.94	75.56

shown in Section 4.2.3 contributes to more sensitive target-word predictions in collocations of phrases. As mentioned above, I will further validate it in Section 4.3.6.

4.3.4 Standard Downstream Tasks

The tasks in this subsection include sequence classification (GLUE benchmark 155), question answering (SQuAD 126, 125), token classification (CoNLL2003 147), and multiple choice (SWAG 175). The selection of those tasks follows the original BERT paper [31]. The experimental results are presented in Tables 4.4 and 4.5 respectively.

As shown in both tables, DG-BERT consistently outperforms my vanilla BERT. Besides, the increments on the two question answering tasks are noticeable. Extrapolating from the performance on the EM metric, it seems that DG-BERT can capture more accurate self-attention for collocation of phrases. An additional analysis for a further verification is described in Section 4.3.6.

Table 4.6 The experimental results are reported on **validation set** for two paraphrase identification tasks.

Tasks	Models	Acc.	Prec.	Rec.	F1
PARADE	Vanilla BERT	73.32	74.31	69.58	71.84
	DG-BERT	75.53	76.53	72.24	74.25
PAWS-wiki	Vanilla BERT	92.60	88.79	95.31	91.93
	DG-BERT	93.61	90.31	95.85	93.00

Table 4.7 The experimental results are reported on test set for four metaphor detection tasks.

Tasks	Backbone Models	Prec.	Rec.	F1
VUA18	Vanilla BERT	78.07	76.82	77.43
	DG-BERT	77.36	78.68	78.01
VUA20	Vanilla BERT	72.83	68.51	70.60
	DG-BERT	72.53	70.23	71.37
MOH-X (zero-transfer)	Vanilla BERT	72.80	78.48	75.53
	DG-BERT	71.76	82.97	76.52
TroFi (zero-transfer)	Vanilla BERT	52.56	72.21	60.84
	DG-BERT	51.93	74.09	60.99

As the definitions of computer science terminologies from PARADE [59] participate in the creation of DG embeddings, and some terminologies appear in definitions of other terminologies, I fine-tune PARADE paraphrase identification task to validate the effectiveness of using the glossary resource. Also to validate if position embeddings of DG-BERT is well pre-trained, I fine-tune the paraphrase identification for PAWS [177] task. This task is meant to measure model’s sensitivity to word order and syntactic structure, which disables zero-transfer from other task datasets. Both tasks have high lexical overlap on non-paraphrases other than paraphrases. The results are provided in Table 4.6. Since the two task datasets contain fairly evenly-distributed paraphrases and non-paraphrases (46.9% & 53.1% for PARADE and 44.2% & 55.8% for PAWS-wiki), two metrics are crucial: accuracy (overall performance) and precision (implying the degree of correctly predicted non-paraphrases). As the results show, DG-BERT gains noticeable improvement. The increment on PARADE task is comparable to SciBERT [6] reported in the original paper, and DG-BERT can retain comparatively high recall score.

4.3.5 Detection of Figurative Semantics

MelBERT [21] designs two task-specific layers above BERT encoder for detecting metaphorical expressions in input sequences. The SPV (Selectional Preference Violation) layer is designed for a theory that a metaphorical word is identified by the semantic difference from its surrounding words; the MIP (Metaphor Identification Procedure) layer is designed for a theory that a metaphorical word is identified by the gap between the contextual and literal meaning of a word. The metaphorical expressions covered in this study comprise single words with different syntactic roles. For instance, in the sentence “Finally, the debate sharpened.”, “sharpend” as a verb conveys figurative semantics. To verify if DG-BERT can

grasp figurative semantics better than vanilla BERT, I apply them as backbone models to MeIBERT for comparison.

The experimental results are shown in Table 4.7 (I follow the source code to report results on test set). VUA18 [86] and VUA20 [85] are two big major task datasets, but with extremely unevenly-distributed labels (approximately 90% literal words and 10% figurative words). Thus, following the original paper, the accuracy metric is trivial and omitted in the table. Two tasks with small datasets MOH-X [111] and TroFi [9] are performed by zero-transfer, namely tested by models fine-tuned on VUA20. The evaluation criterion is the same as the original paper: while the performance gap between models is much small in terms of precision, DG-BERT is better than vanilla BERT in terms of recall, which means DG-BERT as a backbone model can capture more metaphors.

4.3.6 Investigation on Collocations of Phrases

From another perspective, also as a way to observe if DG-BERT can capture more accurate self-attention for collocation of phrases, I perform analyses on raw pre-trained models without fine-tuning via visualization [152] on self-attention. Given a sentence containing an idiomatic expression, I traverse all 12 encoder layers and their 12 heads, to search if there exists such a head that can:

- when the start word of an expression is selected as a pivot (the dot product between its query and keys of other words), whether semantically related words in the collocation can be given proper attention to it;
- semantically unrelated words do not or marginally attend to the start token.

The first ten sentences (containing idiomatic expressions with confidence score 1.0) are selected from MAGPIE [56] for comparison. For idiomatic expressions contained in these ten sentences, there are four of them whose definitions participate in the creation of DG embeddings. DG-BERT outperforms my vanilla BERT by six samples, and achieve four on-par performances. I show two samples (“off the beaten track” with participation and “chapter and verse” without participation) involving all analytic factors in Figures 4.8, 4.9, and 4.10.

I first compare the 1st and 2nd sub-figures of Figure 4.8. Semantically unrelated words are not or marginally given attention in both models. When it comes to allocating proper attention to the collocation, DG-BERT outperforms the vanilla BERT by giving more attention to “beaten” and “track” than “the”, because in this case, it can be distinguishable from other expressions like “off the road/track/etc.”.

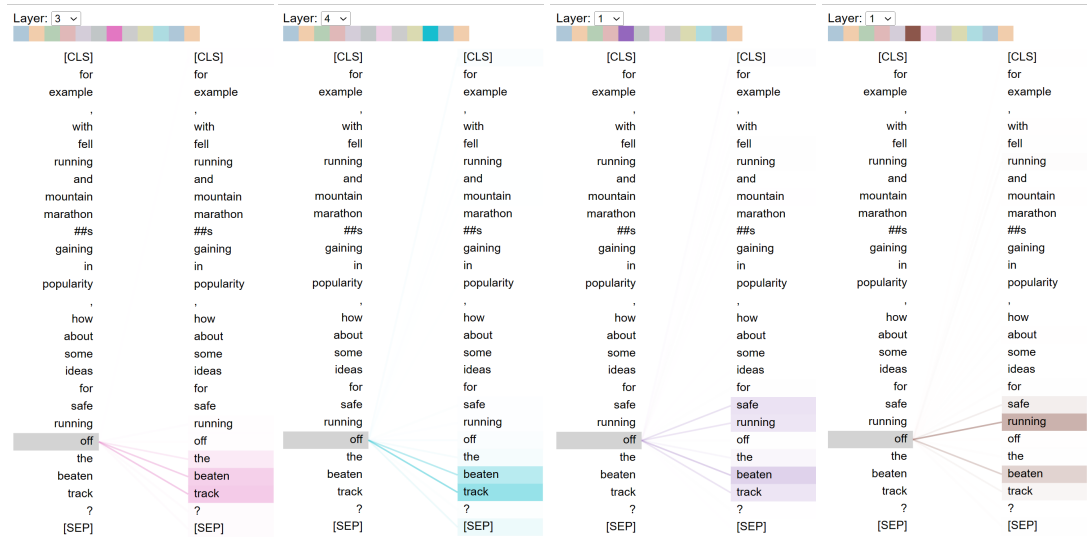


Fig. 4.8 From left to right → **1st: DG-BERT; 2nd: Vanilla BERT; 3rd: DG-BERT; 4th: Vanilla BERT**

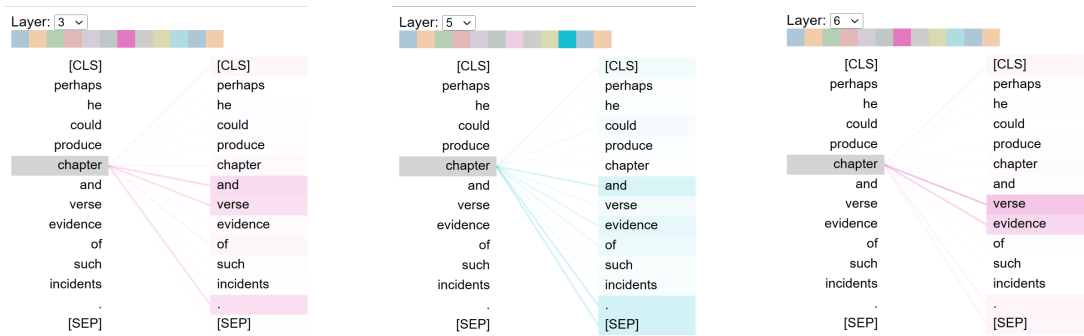


Fig. 4.9 From left to right → **1st: DG-BERT; 2nd: Vanilla BERT; 3rd: DG-BERT**

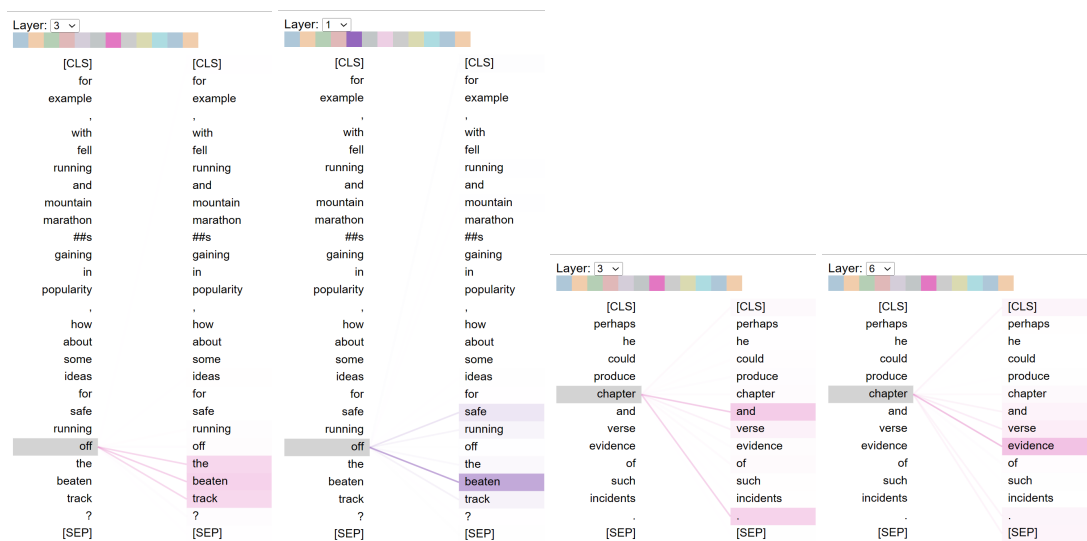


Fig. 4.10 DG, scalar and bias embeddings are removed from pre-trained DG-BERT.

It can be argued that if the combination of “safe” and “running” should also be considered as semantically related words modified by the idiomatic expression. I find one head in DG-BERT (the 3rd sub-figure of Figure 4.8) where both words are given clear attention while proper attention to the collocation is further improved (“beaten” > “track” > “the”, considering another expression “dangerous running off the track” is both syntactically and semantically justifiable), but fail to find one in my vanilla BERT (“safe” and “the” are weak in the 4th sub-figure of Figure 4.8).

From the 1st sub-figure of Figure 4.9, although DG-BERT has a flaw by giving a clear attention to “.”, compared to my vanilla BERT (the 2nd sub-figure of Figure 4.9), it knows that “and” and “verse” are the words collocated with “chapter”. In another head for DG-BERT (the 3rd sub-figure of Figure 4.9), “evidence” as a word modified by the idiomatic expression and “verse” as a key word collocated with “chapter” are given clear attentions. Despite the fact that the attention to “and” is marginal, considering it is a common conjunction (e.g., “chapter and paragraph” is also a legitimate expression), “verse” is comparatively crucial for the form of this collocation. There is no such a head in my vanilla BERT.

I also perform the visualization after removing DG, scalar and bias embeddings from pre-trained DG-BERT, and the results are shown in Figure 4.10. Compared to the ones before removing, the attentions become weaker. A proper allocation to the collocation words disappears in comparison between the 1st sub-figure of Figure 4.8 and the 1st sub-figure of Figure 4.10; the attentions to “track”, “safe” and “running” in the 2nd sub-figure of Figure 4.10 are not as strong as the ones in the 3rd sub-figure of Figure 4.8; the attention to the key collocation word “verse” in the 3rd and 4th sub-figures of Figure 4.10 are not highlighted compared to the ones in the 1st and 3rd sub-figures of Figure 4.9. But they are still relatively better than my vanilla BERT; e.g., “safe” and “running” are preserved in the 2nd sub-figure of Figure 4.10 in comparison to the 4th sub-figure of Figure 4.8, and semantically unrelated words are given less attention in the 3rd sub-figure of Figure 4.10 than the ones in the 2nd sub-figure of Figure 4.9. I search other heads after removing, but do not find ones comparable to the originals. The results justify the important roles of my auxiliary embeddings. Especially for “chapter and verse” whose definition does not participate in the creation of DG embeddings, DG-BERT manages to learn the knowledge that three words form a collocation.

To delve deeper behind the scene, I make use of the ten samples for testing target-word predictions in collocation of phrases. The results are summarized in Table 4.8. In each sample, the word that is a comparatively critical part of a phrase is masked. Except in the second sample where “creeps” is not a single-word vocabulary of BERT, I mask “gave” instead. The results are generally consistent with the preliminary test presented in Table 4.3.

Table 4.8 The first ten sentences containing idiomatic expressions with confidence score 1.0 are selected from MAGPIE [56] for target-word predictions in collocation of phrases. “DG-BERT w/o DG” denotes the whole package of DG, scalar and bias embeddings is removed from pre-trained DG-BERT. When the expected word is not included in top 5 predictions, - is used to indicate its ranking.

Input sequences	Ranking in top 5 predictions (softmax score)	Models
[CLS] For example, with fell running and mountain marathons gaining in popularity, how about some ideas for safe running off the [MASK] track? [SEP]	1 (0.4514)	DG-BERT
	1 (0.6255)	DG-BERT w/o DG
	1 (0.2965)	Vanilla BERT
[CLS] He [MASK] me the creeps, so I looked round, hmm hmm. [SEP]	2 (0.1604)	DG-BERT
	3 (0.0940)	DG-BERT w/o DG
	3 (0.0368)	Vanilla BERT
[CLS] ‘He’s done us [MASK], as well,’ says Granville. [SEP]	- (0.0051)	DG-BERT
	- (0.0054)	DG-BERT w/o DG
	- (0.0009)	Vanilla BERT
[CLS] People quickly embraced formal democracy, but the tolerance and compromise that is at the heart of the democratic process took time to take [MASK]. [SEP]	2 (0.2119)	DG-BERT
	2 (0.1732)	DG-BERT w/o DG
	3 (0.1465)	Vanilla BERT
[CLS] Indeed, the Kursk crisis may provide him with an opportunity to further clean [MASK] in the military. [SEP]	4 (0.0194)	DG-BERT
	- (0.0107)	DG-BERT w/o DG
	- (0.0002)	Vanilla BERT
[CLS] A man has made medical [MASK] by having four organ transplants. [SEP]	1 (0.7121)	DG-BERT
	1 (0.6521)	DG-BERT w/o DG
	1 (0.3540)	Vanilla BERT
[CLS] So I went all the [MASK] down to the door slightly irritated, and what did I see? [SEP]	1 (0.9999)	DG-BERT
	1 (0.9998)	DG-BERT w/o DG
	1 (0.9997)	Vanilla BERT
[CLS] Perhaps he could produce chapter and [MASK] evidence of such incidents. [SEP]	1 (0.7475)	DG-BERT
	1 (0.6853)	DG-BERT w/o DG
	4 (0.0219)	Vanilla BERT
[CLS] By this time, some are up to 14 — but because the original outlay on stock didn’t break the [MASK], Ken can offer them remarkably reasonably — priced. [SEP]	1 (0.6242)	DG-BERT
	1 (0.5768)	DG-BERT w/o DG
	4 (0.0384)	Vanilla BERT
[CLS] I’m still heading for the [MASK]. [SEP]	- (0.0180)	DG-BERT
	- (0.0162)	DG-BERT w/o DG
	- (0.0061)	Vanilla BERT

In comparison between “DG-BERT w/o DG” and my vanilla BERT, the results show that the word embeddings of DG-BERT are better optimized than the ones of my vanilla BERT, given the same pre-training conditions. On the other hand, extrapolating from the results of first and third samples, the semantic features provided by DG embeddings might contain noise, and thus DG-BERT is outperformed by “DG-BERT w/o DG”. Despite minorities, there is still room for improvement, which will be discussed in Section 4.5.

4.4 Further Explanation on Methodological Advantage

I compare my results on GLUE benchmark with the most related work. The results are presented in Table 4.9, and I further explain the intrinsic advantage of my approach.

Although better performance is achieved, it can be argued that DG-BERT achieves the performance with the advantage of additional parameters of DG, scalar and bias embeddings. “DG-BERT w/o DG” shown in Table 4.9 clearly suggests that the performance is marginally diminished after removal. This is also consistent with the results regarding target-word predictions given context words presented in Tables 4.3, 4.8, and the psycholinguistic diagnostics in Appendix A.2, where “DG-BERT w/o DG” is slightly lower than DG-BERT, but noticeably higher than my vanilla BERT. Therefore, better performance is not caused by more parameters. Instead, the word embeddings of DG-BERT are comparatively well optimized with my approach. This can be proved by two observations and one additional visualization. First, in Section 4.3.2 describing pre-training convergence, DG-BERT is noticeably faster than vanilla BERT. Since the weight of decoder is tied to word embeddings, the acquisition of vocabulary for DG-BERT is more effective. Second, semantic information provided by my word embeddings contributes to better self-attention. When comparing “DG-BERT w/o DG” with my vanilla BERT in Figures 4.8, 4.9 and 4.10, “DG-BERT w/o DG” achieves better self-attention than my vanilla BERT (detailed descriptions are given in Section 4.3.6). In language models, the input layer provides fundamental semantic information for attention layers. This can also be validated by the comparison between “DG-BERT w/o DG” and DG-BERT in those figures, as the only difference between them is in the input layer. Especially for collocations of phrases, the high phrasal internal similarities of DG embeddings shown in Tables 4.1 and 4.2 help my model to gain the advantage. Thus, the word embeddings of DG-BERT learn useful semantic information from context words glossed by DG embeddings. Third, word embeddings of language models are known to be anisotropic [48], but the problem is alleviated by my approach. As shown in Figure 4.11, the aperture size of the cone is comparatively increased — in the angle depicted in the **3rd** sub-figure, noticeably more tokens are directionally distributed for the word embeddings of DG-BERT.

Table 4.9 The experimental results of GLUE benchmark are reported on **validation set** for comparison with the most related previous work. “DG-BERT w/o DG” denotes that the whole package of DG, scalar and bias embeddings is removed from pre-trained DG-BERT. My fine-tuning and Dict-BERT [172] are based on the average of five runs with different random seeds.

Models	MNLI (Acc.)	QQP (Acc.)	QNLI (Acc.)	SST2 (Acc.)	COLA (Matthews)	STSB (Pearson)	MRPC (Acc.)	RTE (Acc.)	Avg.	Δ Avg.
Vanilla BERT	84.12	90.75	90.69	92.52	58.59	89.39	86.17	68.67	82.65	-
Dict-BERT	84.34	90.81	91.20	92.65	61.68	89.68	87.21	72.89	83.80	+1.15
Vanilla BERT	83.92	90.90	91.20	91.92	58.50	89.59	86.76	69.34	82.77	-
DG-BERT	84.85	91.00	91.22	93.25	58.70	90.01	87.75	74.01	83.88	+1.11
w/o DG										
DG-BERT	85.09	91.13	91.40	93.00	60.14	90.07	88.09	73.94	84.11	+1.34

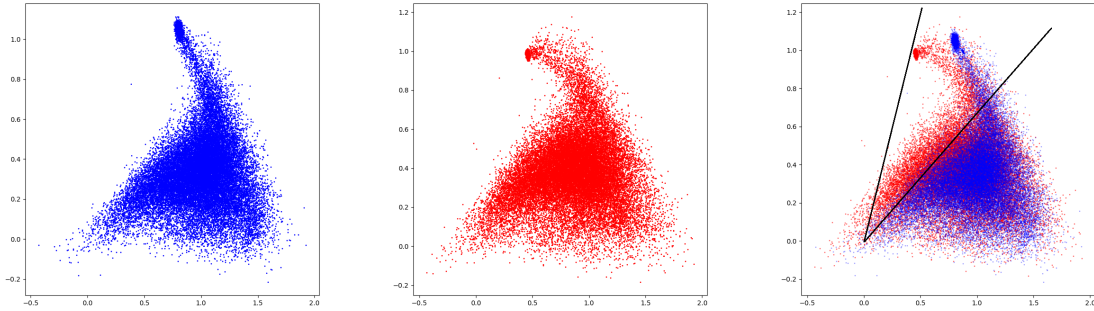


Fig. 4.11 SVD visualization of token distribution of word embeddings from left to right → **1st**: word embeddings of my Vanilla BERT; **2nd**: word embeddings of DG-BERT; **3rd**: comparison between **1st** and **2nd**.

This validates the assumption mentioned in Section 4.2.3. The distribution could be further improved if I adopt the regularization proposed by Gao et al. [48]. In this paper, I focus on the improvements made by my approach illustrated in Figure 4.3.

4.5 Discussion

Based on the experimental results, I conclude that my approach is relatively simple yet effective. If only to further improve task performance, this can be done by using more pre-training corpora and more careful pre-training schema like RoBERTa [95]. This can also be achieved by augmenting self-attention mechanism such as using more encoder layers like BERT-large, or more advanced encoder structure like pre-layer normalization [156, 162]. Additionally, ensembling methods can be considered, as my technical novelty does not

conflict with any previous study. Recently, I have been planning to further pre-train DG-BERT by max sequence length 512 once the physical constraint is resolved, for future works that would like to apply DG-BERT to longer-sequence tasks, or to ensemble it with other models. I will upload a brief technical report after this research activity is completed. My source code and current pre-trained model is publicly available⁶.

4.5.1 Further Insight into Research Meaning

There are two contributions of my work stated in Section 4.1.3. Still, one might argue that the unsupervised definition embeddings mentioned in the second contribution are technically weak among modern technical trends, even though it is proved in this paper that they can assist context words for auto-encoding models. There might be other methods to create DG embeddings, as long as the creation of the embeddings is not subject to the technical duplicate mentioned in Section 4.1.2. However, in NLP domain, my approach offers a witness for the successful cooperation between two general AI research directions: Symbolism and Connectionism. Pre-training BERT belongs to Connectionism, which is based on deep learning of neural networks and big data. On the other hand, the creation of DG embeddings can be considered as an implementation of Symbolism. The extracted key features described in Section 4.2.1 are knowledge symbols. For dictionary entries having similar symbols in the term-document matrix, similar semantics are obtained in their vector representations. The outcome of their cooperation, DG-BERT, yields intrinsic advantages such as better target-word predictions given context words (especially for collocations of phrases) and alleviated anisotropic vector space of word embeddings, as well as better performance on downstream tasks than BERT.

4.5.2 Limitation

Following the convention of NLP community, I would also like to discuss the potential limitations of my approach, and how they can be mitigated.

DG embeddings. The method to create DG embeddings is inspired by the headword search in thesauruses, which is introduced in Section 4.1.2. Through my method described in Section 4.2.1, as shown in Figure 4.4, the key features extracted for each dictionary entry can generate overlap to represent similar semantics. However, some unnecessary key features are also extracted like “regard”, “relate” and so on. This could cause noise for DG embeddings, albeit there is no big negative impact inflicted on my later experimental settings. For future expansion, two solutions are conceivable for quality improvement. First, to scale down the

⁶<https://github.com/ryuliuxiaodong/DG-BERT>

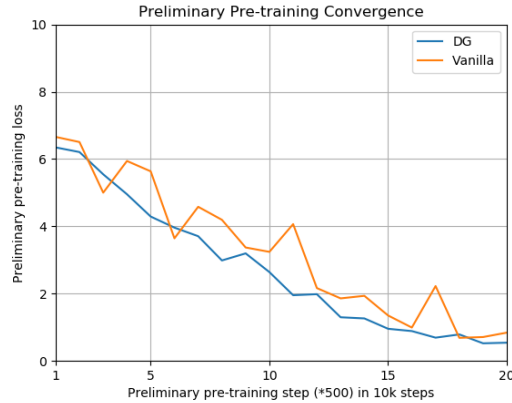


Fig. 4.12 The convergence speed of preliminary pre-training (every 500-step batch loss in first 10k steps). This is performed to consider if full pre-training is worthy of pursuit. Compared to full pre-training, the setting is scaled down in terms of batch size 32, max-seq-length 128, approximately 20k input sequences, which can run on a single GPU device.

threshold of extraction is an option, as in some other real-world scenarios, the 80/20 rule can be more effective after it is scaled down to 82/18. Namely, only top 18% features are considered as key features for extraction to avoid those unnecessary features. Second, as the implementations of Symbolism are typically rule-based (e.g., the 80/20 rule in my case), it can be investigated if there exists such a knowledge base that can be used to filter out those unnecessary key features.

Extensibility. DG embeddings are created for the MLM pre-training objective of BERT — glossing bidirectional context words for target-word prediction. Although there is no guarantee that DG embeddings can also assist the pre-training objectives of other language models, the advantages of DG embeddings ensures a pursuit worthy of exploration: 1) it is inexpensive for the creation (only single CPU device is enough); 2) DG embeddings can be removed from pre-trained models with performance slightly decreased (no occupation of additional memory resource). This is conducive to two scenarios of future expansion enumerated as follows.

1. One attempts to create DG embeddings using different dictionary resources.
2. One seeks to apply DG embeddings to different language models.

For the second research attempt, a preliminary pre-training (small batch size and corpora run on a single GPU) can be conducted to observe if convergence is faster than vanilla models. If so, full pre-training can be performed accordingly. This is what I had done before I performed full pre-training of DG-BERT. As shown in Figure 4.12, in the preliminary pre-training,

DG-BERT is faster than vanilla BERT. Based on the observation, it can be expected that DG embeddings can assist BERT in acquiring vocabulary more effectively, and then full pre-training is performed. As a matter of fact, Figures 4.6 and 4.7 in Section 4.3.2 confirms that full pre-training meets my expectation. After full pre-training, the whole package of DG, and ancillary scalar and bias embeddings can be removed — no extra burden on memory.

4.6 Conclusion

In this paper, I investigated context words-glossed pre-training for the MLM pre-training objective of BERT, which is considered as the Cloze task for machines to learn words from bidirectional context words. I proposed DG embeddings to infuse context words with semantic features of glosses, and utilized scalar and bias embeddings to allow machines to learn how the semantic features represented in DG embeddings should be used. My motivation, insights, and main claims were stated in Section 4.1. The details regarding DG embeddings were provided in Section 4.2. I conducted a wide range of experiments in Section 4.3 to evaluate DG-BERT (vanilla BERT with DG, scalar and bias embeddings), and compared it with my vanilla BERT pre-trained based under the exact same condition. I also compared DG-BERT with the most related work in Section 4.4, and further explained the intrinsic advantage my approach. I discussed my further insight based on the experimental results in Section 4.5, along with the potential limitations of my approach and how they can be addressed.

Ethical Statement

I follow professional practice for NLP technology and the use of data proposed by Bender and Friedman [7]. Above all, this paper is written for purely academic demonstration, and the pre-training data that I use is limited when compared to modern language model pre-training. Nevertheless, there are potential concerns that such data could still be associated with gender bias, behavior patterns, bot masquerading as humans and so on. Although no explicit attempts have been made to identify those cases, pre-processing methods [124] and manual labor can be employed to resolve the concerns if one seeks to apply my model to real-world use. In addition, the dictionary, glossary, and other evaluation datasets involved in this paper are publicly available and widely-used in NLP community. Except for the TOEFL dataset only available via portal request, I have signed an agreement with the dataset provider to use it only in not-for-profit academic researches. I will not leak it to public.

The images contained in Figures 4.1 and 4.3 are cited from <https://publicdomainvectors.org/en/>, where uploaders have waived all copyright to their vector images. There is no issue of using the images for non-profit academic papers.

Chapter 5

Conclusion & Future Directions

5.1 Conclusion

In this dissertation, I described my PhD study regarding the fusion of distributional representations and distributed representations. Distributional representations are advantageous for knowledge corpora, but disadvantageous for word representations, while distributed representations show the opposite attributes. They are complementary to each other. My exploration started from two perspectives: mapping and structural connection.

As for mapping, I build distributional semantic representations for paraphrase and non-paraphrase sentences, belonging to the smallest English paraphrase dataset MRPC [34] as one type of knowledge corpora. Then concatenations of words embeddings (distributed representations) of sentences with my designed weighting scheme are mapped to the factorized distributional representations by two projection matrices for training. After training, the matrices are frozen during inference, and any sentence pairs from any datasets can obtain their factorized distributional representations via the matrices. As a result, only different classifiers need to be trained for different sentence-pair tasks. This contributes to power-efficient continual learning of paraphrase identification and other sentence-pair tasks. 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.

As for structural connection, I use the dictionary knowledge corpus to build distributional semantic representations for dictionary entries. The inspiration stems from a real-word dictionary use called headword search. After factorization and adaption to the same shape as word embeddings of BERT, the distributional representations are used to gloss context words for target-word predictions in the MLM pre-training objective of BERT. This contributes to the better acquisition of vocabulary for BERT, in terms of faster pre-training convergence,

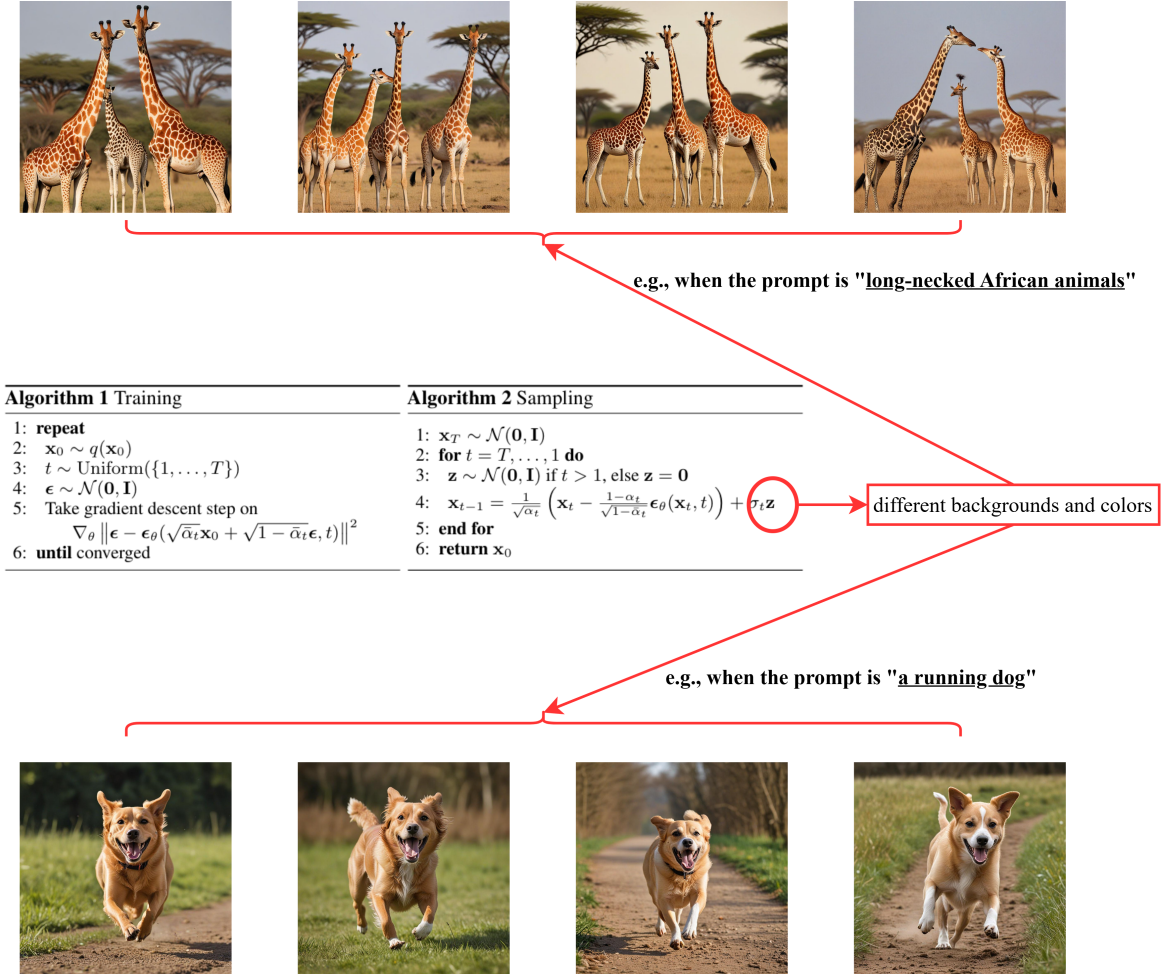


Fig. 5.1 The algorithm of DDPM [63] for text-to-image generators and the generated images are from <https://www.imagine.art/dashboard/tool/from-text> with the given prompts.

better task performance on literal and figurative semantics, more sensitive target-word predictions in the collocation of phrases, and more accurate self-attention for phrases in input sequences. All these are attributed to the improved word embeddings of BERT.

5.2 Future Directions

Studies on semantic representations will remain in my future research directions. Besides exploring other potential fusions of distributional representations and distributed representations, geometry in embedding vector space, transparency of embeddings, and a new language acquisition framework for computational linguistics will be my main focus.

Regarding geometry in embedding vector space, recently, I have completed one research regarding improving sentence embeddings of language models, and the title of my paper

is “DefSent+: Improving sentence embeddings of language models by projecting definition sentences into a quasi-isotropic or isotropic vector space of unlimited dictionary entries” [91]. I have submitted this paper to arXiv as preprint, and this paper is under review by an AI journal. Sentence embeddings improved by my approach have achieved state-of-the-art performance on measuring semantic similarities of sentence pairs, among the approaches without using manually labeled datasets to improve language models.

Regarding transparency of embeddings, this belongs to an important AI research direction called explainable AI. My research about this topic will be centered around text-to-image generators. I am particularly interested in to what degree semantic representations of texts can impact the generations of different images. Generally, according to DDPM (Denoising Diffusion Probabilistic Models) [63], the variances used during generation (the mathematical term with red circle in Figure 5.1) are the main reason why images are generated with different backgrounds or colors; e.g., when the prompt is “a running dog”, four images are generated with different backgrounds and colors, but all of them match the description “a running dog”. However, the question is that why the dog is not a Husky, Shiba inu, or Border Collie? The question is still valid when the prompt is “long-necked African animals”: why are the images of giraffes generated only? what about ostriches or vultures also with long necks? My research goal is to devise an approach that can be used to explain to what extents semantic representations of texts can result in broader differences of generated images.

Regarding a new language acquisition framework for computational linguistics, it can be argued that language model pre-training nowadays is actually based on second language acquisition — directly feeding enormous text data to language model (as a form of learning by reading). On the other hand, for first language acquisition of humans, the priority of learning a natural language is **Listening** (babies listen to phonetic patterns before they can speak) → **Speaking** (children start to mimic what they hear) → **Reading** (children attend to school lessons) → **Writing** (after accumulation of lessons, youngsters begin to write diaries, essays, etc. in a prescriptive manner). As for Speaking, Reading and Writing, I have a mature conceptualization of how they can be implemented with existing technologies. However, regarding the crucial fundamental Listening step, I have only a preliminary conceptualization as to how it can be implemented by introducing a new mathematical tool — knowledge representations acquired by Listening for machines to use. Empirical research attempts will be made to test if my idea is feasible.

Acknowledgement

During my PhD study in the last 3.5 years, I have been able to accomplish my research topic described in this dissertation. The success is attributed to not only my continuous passion, but also the supports from multiple sources for my passion to continue. In this section, I would like to express my deepest gratitude to those supports.

Firstly, I really appreciate the DX fellowship of Hokkaido University (Grant Number JPMJSP2119). The monthly allowance enables me to focus on my PhD studying without worrying extra financial burden on my daily life. Especially, the second journal publication of mine involves language model pre-training, which is expensive, albeit the language models pre-trained for my research purpose is not comparatively large among LLMs nowadays. The research funds supported by the DX fellowship helped me to complete this research in time, and moreover, extra funds requested by me was also permitted. Many thanks to your support.

Next, my wholehearted thanks go to Prof. Kenji Araki and Associate Prof. Toshihiko Ito. Prof. Kenji Araki was my supervisor until April 2024, the time he retired from Hokkaido University. I really appreciate the opportunity he gives to me to study in our Language Media Lab, and the free research topic that he allows me to focus on. After the retirement of Prof. Kenji Araki, Associate Prof. Toshihiko Ito begins to take on the responsibility to handle relevant documents of my PhD graduation. This is the first time he deals with bureaucratic procedures, which can be time-consuming. Many thanks to your continuous investigations during your precious research time.

In addition, I would also like to say thanks to Assistant Prof. Rafal Rzepka. I am thankful to him for patiently proofreading my papers before submission for typo-checking and sharing some writing skills. Also, as I live far away from the campus, some bureaucratic documents are handled by him so that I can leave lots of time to focus on my research.

Special thanks to the members of the review committee: Prof. Yuji Sakamoto, Prof. Miki Haseyama, Prof. Yoshinori Dobashi and Associate Prof. Toshihiko Ito. Thank you for your time in constructing feedback that would help further refine this dissertation.

Last but not least, it cannot be more grateful to my family. Thank you very much for understanding and supporting my choice of PhD studying in Japan, which is a long-term exhausting exploration. I will keep my passion as an academic researcher.

List of Publications

Two papers listed below have been accepted by journals as articles after peer review.

1. **Xiaodong Liu**, Rafal Rzepka, and Kenji Araki. "Parameter-Efficient Feature-Based Transfer for Paraphrase Identification." *Natural Language Engineering* 29, no.4(2023): 1066-96. <https://doi.org/10.1017/S135132492200050X>.
2. **Xiaodong Liu**, Rafal Rzepka, and Kenji Araki, "DG Embeddings: The unsupervised definition embeddings learned from dictionary and glossary to gloss context words of Cloze task," *Knowledge-Based Systems*, Volume 296, 2024, 111883, ISSN 0950-7051, <https://doi.org/10.1016/j.knosys.2024.111883>.

One paper listed below has been recently submitted to arXiv as preprint. As of writing this dissertation, it is under review.

1. **Xiaodong Liu**. DefSent+: Improving sentence embeddings of language models by projecting definition sentences into a quasi-isotropic or isotropic vector space of unlimited dictionary entries. arXiv preprint arXiv:2405.16153. (2024)

References

- [1] AbuSeileek, A. F. (2011). Hypermedia annotation presentation: The effect of location and type on the efl learners' achievement in reading comprehension and vocabulary acquisition. *Computers Education*, 57(1):1281–1291.
- [2] Apidianaki, M. (2023). From Word Types to Tokens and Back: A Survey of Approaches to Word Meaning Representation and Interpretation. *Computational Linguistics*, pages 1–58.
- [3] Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada. Association for Computational Linguistics.
- [4] Balazevic, I., Allen, C., and Hospedales, T. (2019). TuckER: Tensor factorization for knowledge graph completion. 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)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.
- [5] Beltagy, I., Lo, K., and Cohan, A. (2019a). 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)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- [6] Beltagy, I., Lo, K., and Cohan, A. (2019b). 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)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- [7] Bender, E. M. and Friedman, B. (2018). Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.
- [8] 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*, page 41–48, New York, NY, USA. Association for Computing Machinery.

- [9] Birke, J. and Sarkar, A. (2006). A clustering approach for nearly unsupervised recognition of nonliteral language. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 329–336, Trento, Italy. Association for Computational Linguistics.
- [10] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.
- [11] Blevins, T. and Zettlemoyer, L. (2020). Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1006–1017, Online. Association for Computational Linguistics.
- [12] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.
- [13] Bosc, T. and Vincent, P. (2018). Auto-encoding dictionary definitions into consistent word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532, Brussels, Belgium. Association for Computational Linguistics.
- [14] 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.
- [15] Caruana, R. (1998). *Multitask Learning*, pages 95–133. Springer US, Boston, MA.
- [16] 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)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.
- [17] Chandrasekaran, D. and Mago, V. (2020). Evolution of semantic similarity - A survey. *CoRR*, abs/2004.13820.
- [18] Chang, K.-W., Yih, W.-t., and Meek, C. (2013). Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1602–1612, Seattle, Washington, USA. Association for Computational Linguistics.
- [19] Chen, Q., Li, F.-L., Xu, G., Yan, M., Zhang, J., and Zhang, Y. (2022). Dictbert: Dictionary description knowledge enhanced language model pre-training via contrastive learning. In *IJCAI*, pages 4086–4092.
- [20] Chen, T., Goodfellow, I., and Shlens, J. (2016). Net2net: Accelerating learning via knowledge transfer.

- [21] Choi, M., Lee, S., Choi, E., Park, H., Lee, J., Lee, D., and Lee, J. (2021). MelBERT: Metaphor detection via contextualized late interaction using metaphorical identification theories. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1763–1773, Online. Association for Computational Linguistics.
- [22] Chow, W.-Y., Smith, C., Lau, E., and Phillips, C. (2016). A “bag-of-arguments” mechanism for initial verb predictions. *Language, Cognition and Neuroscience*, 31(5):577–596.
- [23] 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).
- [24] 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)*, pages 301–308, Held Online. INCOMA Ltd.
- [25] 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*, pages 468–476, Suntec, Singapore. Association for Computational Linguistics.
- [26] 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., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- [27] de Wynter, A. and Perry, D. J. (2020). Optimal subarchitecture extraction for bert.
- [28] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990a). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.
- [29] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990b). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.
- [30] 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*, pages 248–255. Ieee.
- [31] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019a). 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

- [32] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019b). 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- [33] Dolan, B., Quirk, C., and Brockett, C. (2004a). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356, Geneva, Switzerland. COLING.
- [34] Dolan, B., Quirk, C., and Brockett, C. (2004b). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356, Geneva, Switzerland. COLING.
- [35] 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)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.
- [36] Emerson, G. (2020). What are the goals of distributional semantics? In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7436–7453, Online. Association for Computational Linguistics.
- [37] Ethayarajh, K. (2019). How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.
- [38] Ettinger, A. (2020). What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.
- [39] Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015a). 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*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.
- [40] Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015b). 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*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.

- [41] Federmeier, K. D. and Kutas, M. (1999). A rose by any other name: Long-term memory structure and sentence processing. *Journal of memory and Language*, 41(4):469–495.
- [42] Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.
- [43] 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*, pages 45–52.
- [44] Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.
- [45] French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135.
- [46] Gamberman, A., Vovk, V., and Vapnik, V. (1998). Learning by transduction. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 148–155. Morgan Kaufmann Publishers Inc.
- [47] Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2013). PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.
- [48] Gao, J., He, D., Tan, X., Qin, T., Wang, L., and Liu, T. (2019). Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*.
- [49] Glavaš, G. and Vulić, I. (2018a). Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45, Melbourne, Australia. Association for Computational Linguistics.
- [50] Glavaš, G. and Vulić, I. (2018b). Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45, Melbourne, Australia. Association for Computational Linguistics.
- [51] Gokaslan, A. and Cohen, V. (2019). Openwebtext corpus. <http://Skylion007.github.io/OpenWebTextCorpus>.
- [52] Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, USA.
- [53] Gugger, S., Debut, L., Wolf, T., Schmid, P., Mueller, Z., Mangrulkar, S., Sun, M., and Bossan, B. (2022). Accelerate: Training and inference at scale made simple, efficient and adaptable. <https://github.com/huggingface/accelerate>.
- [54] 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*, pages 864–872. Association for Computational Linguistics.

- [55] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. (2020). Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.
- [56] Haagsma, H., Bos, J., and Nissim, M. (2020). MAGPIE: A large corpus of potentially idiomatic expressions. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 279–287, Marseille, France. European Language Resources Association.
- [57] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.
- [58] 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*, pages 1576–1586, Lisbon, Portugal. Association for Computational Linguistics.
- [59] He, Y., Wang, Z., Zhang, Y., Huang, R., and Caverlee, J. (2020a). 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)*, pages 7572–7582, Online. Association for Computational Linguistics.
- [60] He, Y., Wang, Z., Zhang, Y., Huang, R., and Caverlee, J. (2020b). 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)*, pages 7572–7582, Online. Association for Computational Linguistics.
- [61] 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*, pages 2545–2568, Online. Association for Computational Linguistics.
- [62] Hill, F., Cho, K., Korhonen, A., and Bengio, Y. (2016). Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.
- [63] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.
- [64] Hounsby, N., Giurigu, 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., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.
- [65] Hounsby, N., Giurigu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019b). Parameter-efficient transfer learning for nlp.

- [66] 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)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.
- [67] Huang, L., Sun, C., Qiu, X., and Huang, X. (2019). GlossBERT: BERT for word sense disambiguation with gloss knowledge. 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)*, pages 3509–3514, Hong Kong, China. Association for Computational Linguistics.
- [68] 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., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- [69] Inan, H., Khosravi, K., and Socher, R. (2017). Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*.
- [70] Ishiwatari, S., Hayashi, H., Yoshinaga, N., Neubig, G., Sato, S., Toyoda, M., and Kitsuregawa, M. (2019). Learning to describe unknown phrases with local and global contexts. 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)*, pages 3467–3476, Minneapolis, Minnesota. Association for Computational Linguistics.
- [71] Iyer, S., Dandekar, N., and Csernai, K. (2017). First Quora Dataset Release: Question Pairs. <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>.
- [72] Jaccard, P. (1912). The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50.
- [73] 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*, pages 891–896.
- [74] Jung, J. (2016). Effects of glosses on learning of l2 grammar and vocabulary. *Language Teaching Research*, 20(1):92–112.
- [75] Kadotani, S., Kajiwar, 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*, pages 229–234, Online. Association for Computational Linguistics.
- [76] Ke, G., He, D., and Liu, T.-Y. (2021). Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*.
- [77] Khot, T., Sabharwal, A., and Clark, P. (2018). SciTail: A textual entailment dataset from science question answering. In *AAAI*.

- [78] 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*, 114(13):3521–3526.
- [79] Knight, S. (1994). Dictionary use while reading: The effects on comprehension and vocabulary acquisition for students of different verbal abilities. *The Modern Language Journal*, 78(3):285–299.
- [80] 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)*, pages 1235–1245. Association for Computational Linguistics.
- [81] 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.
- [82] Landauer, T., Laham, D., and Foltz, P. (1997). Learning human-like knowledge by singular value decomposition: A progress report. In Jordan, M., Kearns, M., and Solla, S., editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press.
- [83] Landauer, T. K. and Dumais, S. T. (1997a). A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.
- [84] Landauer, T. K. and Dumais, S. T. (1997b). A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.
- [85] 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*, pages 18–29, Online. Association for Computational Linguistics.
- [86] 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*, pages 56–66, New Orleans, Louisiana. Association for Computational Linguistics.
- [87] Levine, Y., Lenz, B., Dagan, O., Ram, O., Padnos, D., Sharir, O., Shalev-Shwartz, S., Shashua, A., and Shoham, Y. (2020). SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.
- [88] Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

- [89] Lin, Y., An, Z., Wu, P., and Ma, Z. (2022). Improving contextual representation with gloss regularized pre-training. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 907–920, Seattle, United States. Association for Computational Linguistics.
- [90] Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., and Wang, P. (2020). Kbert: Enabling language representation with knowledge graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2901–2908.
- [91] Liu, X. (2024). Defsent+: Improving sentence embeddings of language models by projecting definition sentences into a quasi-isotropic or isotropic vector space of unlimited dictionary entries. *arXiv preprint arXiv:2405.16153*.
- [92] Liu, X., Rzepka, R., and Araki, K. (2022). Parameter-efficient feature-based transfer for paraphrase identification. *Natural Language Engineering*, page 1–31.
- [93] Liu, X., Rzepka, R., and Araki, K. (2024). Dg embeddings: The unsupervised definition embeddings learned from dictionary and glossary to gloss context words of cloze task. *Knowledge-Based Systems*, 296:111883.
- [94] Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. (2021). Gpt understands, too.
- [95] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.
- [96] 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., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- [97] 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*, pages 182–190, Montréal, Canada. Association for Computational Linguistics.
- [98] 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)*, pages 1–8, Dublin, Ireland. Association for Computational Linguistics.
- [99] Maćkiewicz, A. and Ratajczak, W. (1993). Principal components analysis (pca). *Computers Geosciences*, 19(3):303–342.
- [100] McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730.

- [101] McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109 – 165. Academic Press.
- [102] McGregor, W. B. (2015). *Linguistics: an introduction*. Bloomsbury Publishing.
- [103] 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*, pages 2551–2562, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- [104] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. (2018). Mixed precision training. In *International Conference on Learning Representations*.
- [105] Mihalcea, R., Corley, C., Strapparava, C., et al. (2006). Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780.
- [106] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [107] Mikolov, T., Grave, E., Bojanowski, P., Puhersch, 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).
- [108] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- [109] Miller, G. A. (1998). *WordNet: An electronic lexical database*. MIT press.
- [110] Minsky, M. and Papert, S. A. (2017). *Perceptrons, reissue of the 1988 expanded edition with a new foreword by Léon Bottou: an introduction to computational geometry*. MIT press.
- [111] Mohammad, S., Shutova, E., and Turney, P. (2016). Metaphor as a medium for emotion: An empirical study. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 23–33, Berlin, Germany. Association for Computational Linguistics.
- [112] Mohiuddin, T. and Joty, S. (2020). Unsupervised Word Translation with Adversarial Autoencoder. *Computational Linguistics*, 46(2):257–288.
- [113] Nesi, H. (2012). Dictionary use. In *The encyclopedia of applied linguistics*. Wiley.
- [114] Nighojkar, A. and Licato, J. (2021). Improving paraphrase detection with the adversarial paraphrasing task. *CoRR*, abs/2106.07691.

- [115] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- [116] 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)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- [117] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). 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)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- [118] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). 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)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- [119] Peters, M. E., Neumann, M., Logan, R., Schwartz, R., Joshi, V., Singh, S., and Smith, N. A. (2019). Knowledge enhanced contextual word representations. 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)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.
- [120] Popović, M. (2015). chrF: character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.
- [121] Press, O. and Wolf, L. (2017). Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.
- [122] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding with unsupervised learning.
- [123] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.
- [124] Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. (2021). Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

- [125] Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.
- [126] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- [127] 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)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.
- [128] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.
- [129] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks.
- [130] Salton, G. et al. (1971). The smart system—experiments in automatic document processing.
- [131] Salton, G., Wong, A., and Yang, C.-S. (1975a). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.
- [132] Salton, G., Wong, A., and Yang, C. S. (1975b). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.
- [133] 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.
- [134] Sap, M., Le Bras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., Roof, B., Smith, N. A., and Choi, Y. (2019). Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'19/IAAI'19/EAAI'19*. AAAI Press.
- [135] Schütze, H., Manning, C. D., and Raghavan, P. (2008). Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference*, page 260.
- [136] Sellam, T., Das, D., and Parikh, A. P. (2020). BLEURT: learning robust metrics for text generation. *CoRR*, abs/2004.04696.
- [137] Shi, W., Chen, M., Zhou, P., and Chang, K.-W. (2019a). 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)*, pages 1198–1203, Hong Kong, China. Association for Computational Linguistics.
- [138] Shi, W., Chen, M., Zhou, P., and Chang, K.-W. (2019b). 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)*, pages 1198–1203, Hong Kong, China. Association for Computational Linguistics.
- [139] 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*, pages 801–809.
- [140] Speer, R., Chin, J., and Havasi, C. (2017). Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI’17*, page 4444–4451. AAAI Press.
- [141] 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*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.
- [142] Sun, F.-K., Ho, C.-H., and Lee, H.-Y. (2020a). LAMOL: LAnguage MOdeling for Lifelong Language Learning. In *International Conference on Learning Representations*.
- [143] Sun, T., Shao, Y., Qiu, X., Guo, Q., Hu, Y., Huang, X., and Zhang, Z. (2020b). CoLAKE: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3660–3670, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- [144] Taylor, W. L. (1953). “cloze procedure”: A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.
- [145] Thrun, S. (1998). *Lifelong Learning Algorithms*, pages 181–209. Springer US, Boston, MA.
- [146] Tissier, J., Gravier, C., and Habrard, A. (2017). Dict2vec : Learning word embeddings using lexical dictionaries. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Copenhagen, Denmark. Association for Computational Linguistics.
- [147] Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.
- [148] Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.
- [149] van de Ven, G. M. and Tolias, A. S. (2019). Three scenarios for continual learning.

- [150] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017a). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- [151] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017b). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- [152] Vig, J. (2019). A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.
- [153] 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*, pages 131–138.
- [154] 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*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- [155] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019a). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.
- [156] Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D. F., and Chao, L. S. (2019b). Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, Florence, Italy. Association for Computational Linguistics.
- [157] Wang, X., Gao, T., Zhu, Z., Zhang, Z., Liu, Z., Li, J., and Tang, J. (2021). KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.
- [158] Wege, M. V. D. (2018). Teaching strategies for independent vocabulary development. *The TESOL Encyclopedia of English Language Teaching*, pages 1–7.
- [159] 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.
- [160] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

- [161] Wu, Q., Xing, C., Li, Y., Ke, G., He, D., and Liu, T.-Y. (2021). Taking notes on the fly helps language pre-training. In *International Conference on Learning Representations*.
- [162] Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. (2020). On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.
- [163] Yamagiwa, H., Oyama, M., and Shimodaira, H. (2023). Discovering universal geometry in embeddings with ICA. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4647–4675, Singapore. Association for Computational Linguistics.
- [164] 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., editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.
- [165] Yih, W.-t., Zweig, G., and Platt, J. C. (2012). Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12*, page 1212–1222, USA. Association for Computational Linguistics.
- [166] 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*, pages 901–911, Denver, Colorado. Association for Computational Linguistics.
- [167] Yin, W. and Schütze, H. (2015b). 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*, pages 901–911, Denver, Colorado. Association for Computational Linguistics.
- [168] Yin, W. and Schütze, H. (2015c). 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*, pages 1368–1373, Denver, Colorado. Association for Computational Linguistics.
- [169] 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.
- [170] 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., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

- [171] Yu, M. and Dredze, M. (2014). Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 545–550, Baltimore, Maryland. Association for Computational Linguistics.
- [172] Yu, W., Zhu, C., Fang, Y., Yu, D., Wang, S., Xu, Y., Zeng, M., and Jiang, M. (2022). Dict-BERT: Enhancing language model pre-training with dictionary. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1907–1918, Dublin, Ireland. Association for Computational Linguistics.
- [173] 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*, pages 43–51, Austin, TX. Association for Computational Linguistics.
- [174] 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)*.
- [175] Zellers, R., Bisk, Y., Schwartz, R., and Choi, Y. (2018). SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.
- [176] Zhang, Y., Baldridge, J., and He, L. (2019a). 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)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.
- [177] Zhang, Y., Baldridge, J., and He, L. (2019b). PAWS: Paraphrase Adversaries from Word Scrambling. In *Proc. of NAACL*.
- [178] Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., and Liu, Q. (2019c). ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.
- [179] Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

Appendix A

Supplementary Information to Chapter 4

A.1 Hyperparameters

Pre-training and fine-tuning hyperparameters are listed in Table A.1. For fine-tuning, I follow previous works [161, 76] to set 10 epochs for small training sets. As for the batch size used in those sets, compared to 8 and 16 for my vanilla BERT, 12 can produce more stable performance for DG-BERT. Max sequence length of 384 is used for long-sequence question answering tasks. When I apply my models as backbone models to MelBERT [21], I use their default sequence length 150.

A.2 Psycholinguistic Diagnostics

To verify if DG-BERT is also sensitive to target-word predictions in the context without collocation, I perform two tests utilized in the psycholinguistic diagnostics proposed by Ettinger [38]. CPRAG-102 stems from an influential human study [41] to test how brains respond to different types of context completions. ROLE-88 originates from a human experiment [22] for testing the brain’s sensitivity to role reversals. For the single-word vocabulary of BERT, 34 contexts from CPRAG-102 and 44 context pairs from ROLE-88 are selected by Ettinger [38].

The results are presented from Tables A.2 to A.9. In Tables A.2 and A.3, the value under the column of “preferring good” is calculated based on the softmax scores where expected word is higher than inappropriate words (good completion), regardless of top 5 predictions; the “w/ 0.01 threshold” indicates that the difference of softmax scores in good completion is greater than 0.01. The overall performance is consistent with the target-word predictions in the context with collocation shown in the previous subsection: DG-BERT > “DG-BERT

Table A.1 The hyperparameters for pre-training and fine-tuning DG-BERT are summarized in this table.

Hyperparameters	Pre-training	Fine-tuning
Total Steps	1M	-
Total Epochs	40	{3, 10}
Learning Rate	1e-4	{2e-5, 3e-5, 4e-5}
Batch Size	256	{12, 32}
Warm-up Ratio	0.01	-
Max Sequence Length	448	{128, 384}
Learning Rate Decay	Linear	Linear
Adam ϵ	1e-6	1e-6
Adam (β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)
Dropout	0.1	0.1
Weight Decay	0.01	0.01
Mixed Precision	fp16	fp16

Table A.2 Models are compared on CPRAG-102 word prediction accuracies, and corresponding sensitivity tests: prefer expected completions over inappropriate completions.

Models	Percentage of correct prediction (Top 1 / Top 5)	Percentage of preferring good	w/ 0.01 threshold
Vanilla BERT	11.8 / 38.2	64.7	35.3
DG-BERT	17.7 / 47.1	67.7	47.1
DG-BERT w/o DG	14.7 / 41.2	64.7	44.1

Table A.3 Models are compared on ROLE-88 word prediction accuracies, and corresponding sensitivity tests: prefer expected completions over inappropriate completions.

Models	Percentage of correct prediction (Top 1 / Top 5)	Percentage of preferring good	w/ 0.01 threshold
Vanilla BERT	12.5 / 29.5	70.5	34.1
DG-BERT	14.8 / 36.4	81.8	40.9
DG-BERT w/o DG	14.8 / 30.7	75.0	36.4

Table A.4 Top word predictions are selected for the CPRAG-102 items by Ettinger [38]. The underlined predictions are the expected ones.

Context	Vanilla BERT predictions
Pablo wanted to cut the lumber he had bought to make some shelves. He asked his neighbor if he could borrow her ____.	lumber, furniture, wood, house, equipment
The snow had piled up on the drive so high that they couldn't get the car out. When Albert woke up, his father handed him a ____.	gun, key, hug, <u>shovel</u> , ride
At the zoo, my sister asked if they painted the black and white stripes on the animal. I explained to her that they were natural features of a ____.	zoo, animal, dog, bird, lion

Table A.5 Top word predictions are selected for the CPRAG-102 items by Ettinger [38]. The underlined predictions are the expected ones.

Context	DG-BERT predictions
Pablo wanted to cut the lumber he had bought to make some shelves. He asked his neighbor if he could borrow her ____.	house, car, money, truck, tractor
The snow had piled up on the drive so high that they couldn't get the car out. When Albert woke up, his father handed him a ____.	<u>shovel</u> , package, bag, letter, gun
At the zoo, my sister asked if they painted the black and white stripes on the animal. I explained to her that they were natural features of a ____.	deer, cat, <u>zebra</u> , spider, tiger

Table A.6 Top word predictions are selected for the CPRAG-102 items by Ettinger [38]. The underlined predictions are the expected ones.

Context	DG-BERT w/o DG predictions
Pablo wanted to cut the lumber he had bought to make some shelves. He asked his neighbor if he could borrow her ____.	house, car, tractor, lumber, money
The snow had piled up on the drive so high that they couldn't get the car out. When Albert woke up, his father handed him a ____.	<u>shovel</u> , package, gun, bag, phone
At the zoo, my sister asked if they painted the black and white stripes on the animal. I explained to her that they were natural features of a ____.	cat, spider, human, <u>zebra</u> , deer

Table A.7 Top word predictions are selected for the ROLE-88 items by Ettinger [38]. The underlined predictions are the expected ones. * denotes inappropriate words.

Context	Vanilla BERT predictions
the camper reported which bear the girl had ____.	eaten, shot, killed, bitten, become
the camper reported which girl the bear had ____.	bitten, *, taken, attacked, kidnapped
the restaurant owner forgot which waitress the customer had ____.	ordered, asked, met, hired, been
the restaurant owner forgot which customer the waitress had ____.	been, <u>served</u> , met, seen, asked

Table A.8 Top word predictions are selected for the ROLE-88 items by Ettinger [38]. The underlined predictions are the expected ones. * denotes inappropriate words.

Context	DG-BERT predictions
the camper reported which bear the girl had ____.	killed, eaten, <u>seen</u> , hunted, shot
the camper reported which girl the bear had ____.	killed, *, <u>eaten</u> , hit, bitten
the restaurant owner forgot which waitress the customer had ____.	hired, chosen, been, booked, requested
the restaurant owner forgot which customer the waitress had ____.	been, <u>served</u> , chosen, become, had

Table A.9 Top word predictions are selected for the ROLE-88 items by Ettinger [38]. The underlined predictions are the expected ones. * denotes inappropriate words.

Context	DG-BERT w/o DG predictions
the camper reported which bear the girl had ____.	killed, eaten, <u>seen</u> , hunted, hit
the camper reported which girl the bear had ____.	killed, bitten, *, <u>eaten</u> , hit
the restaurant owner forgot which waitress the customer had ____.	been, hired, chosen, booked, wanted
the restaurant owner forgot which customer the waitress had ____.	been, <u>served</u> , chosen, become, had

w/o DG” > vanilla BERT. DG-BERT achieve comparatively good predictions on the difficult samples selected by Ettinger [38] shown from Tables A.4 to A.9. It can be concluded that DG-BERT is comparatively sensitive to context words for target-word predictions. The performance on ROLE-88 further explains the reason why DG-BERT can achieve better results on the PAWS-wiki task (see Table 4.6).