



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Ph.D. DISSERTATION

Contextualized Language Representations
with Deep Neural Networks for
Unsupervised Learning

비지도 학습을 위한 딥 뉴럴 네트워크 기반 문맥화된
언어 표현 연구

BY

SHIN JOONGBO

FEBRUARY 2021

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Contextualized Language Representations
with Deep Neural Networks for
Unsupervised Learning

비지도 학습을 위한 딥 뉴럴 네트워크 기반 문맥화된
언어 표현 연구

BY

SHIN JOONGBO

FEBRUARY 2021

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Contextualized Language Representations with Deep Neural Networks for Unsupervised Learning

비지도 학습을 위한 딥 뉴럴 네트워크 기반 문맥화된
언어 표현 연구

지도교수 정 교 민
이 논문을 공학박사 학위논문으로 제출함

2021년 2월

서울대학교 대학원






전기 정보 공학부

신 중 보

신중보의 공학박사 학위 논문을 인준함

2021년 2월

위 원 장:	이 상 구
부위원장:	정 교 민
위 원:	윤 성 로
위 원:	김 건 희
위 원:	임 성 수

(인) 
(인) 
(인) 
(인) 
(인) 

Abstract

In natural language processing, deep neural networks are powerful language learners since they are able to incorporate context information from raw text data in a flexible way. Language representations learned in an unsupervised manner on a large corpus provide a source for deep neural networks to understand human language better. Natural language understanding has been made remarkable progress with pre-training contextualized language representations using language modeling, which is a representative unsupervised learning or self-supervised learning technique. In contextualized language representation learning, autoregressive language modeling and masked language modeling are two major learning objectives, and state-of-the-art pre-training methods are based on these two tasks. This dissertation presents a novel language modeling task called bidirectional language autoencoding that takes advantage of both of the previous learning objectives. The proposed learning objective enables a model to understand a text in a deep and bidirectional way like masked language modeling, and at the same time, to extract contextualized language representations without fine-tuning like autoregressive language modeling. To learn bidirectional language autoencoding, this dissertation introduces a novel network architecture of a deep bidirectional language model. The presented architecture allows the bidirectional language model to learn useful language representations rather than simply copying and allows each word to have a contextualized representation. The main contribution of this dissertation is the verification that the proposed bidirectional language autoencoding can be a better approach than the previous language modeling tasks when extracting contextualized language representations for natural language understanding tasks. Experimental results are presented on N -best list re-ranking, semantic textual similarity, word sense disambiguation, and text classification, demonstrating the advantages of the advanced unsupervised representation learning over previous language modelings.

keywords: deep neural networks, language modeling, unsupervised learning,
contextualized language representations

student number: 2014-21625

Contents

Abstract	i
Contents	iii
List of Tables	vii
List of Figures	ix
1 Introduction	1
1.1 Overview	1
1.2 Contributions and Outline of This Dissertation	6
2 Background: Language Representation Models	8
2.1 Non-contextualized Word Representations: Word Embeddings	9
2.2 ALM-based Language Representation Models	12
2.2.1 ELMo	13
2.2.2 GPT	14
2.3 MLM-based Language Representation Models	16
2.3.1 BERT	16
2.3.2 Other Language Representation Models	19
2.4 Base Language Model Architecture: SAN	21

3	Masked Language Modeling for Sentence Scoring	26
3.1	Accurate Bidirectional LMs	26
3.1.1	Overview	26
3.1.2	Contributions	27
3.2	Related Works	29
3.2.1	Bidirectional LMs in NLP	29
3.2.2	Bidirectional LMs for ASR	29
3.3	Methodology	30
3.3.1	Architecture of SAN-based LMs	30
3.3.2	Sentence Scoring with SANLMs	32
3.3.3	Re-ranking the N-best List with SANLMs	36
3.4	Experiments	37
3.4.1	Acoustic Model Setups	37
3.4.2	Language Model Setups	38
3.4.3	Results: Re-ranking the N-best List	39
3.4.4	Analysis: Misrecognized Position	41
3.5	Summary of MLM for Sentence Scoring	43
4	Bidirectional Language Autoencoding for Sentence Scoring	44
4.1	Fast and Accurate Bidirectional LMs	44
4.1.1	Overview	44
4.1.2	Contributions	45
4.2	Related Works	47
4.2.1	Bidirectional LMs for Unsupervised Tasks	47
4.2.2	Consideration of Inference Speed	47
4.3	Methodology	48
4.3.1	Baselines: ALM for UniLM and MLM for BiLM	48
4.3.2	BLA: New Language Modeling Objective	49
4.3.3	T-TA: New Deep Bidirectional Language Model	51

4.3.4	Verification of the T-TA Architecture	55
4.3.5	Comparison T-TA with BERT	56
4.4	Experiments	57
4.4.1	Language Model Setups	57
4.4.2	Analysis: Runtime Comparison	59
4.4.3	Settings: Re-ranking the N-best List	61
4.4.4	Results: Re-ranking the N-best List	65
4.4.5	Analysis: Re-ranking and Language models	68
4.5	Summary of BLA for Sentence Scoring	72
5	Bidirectional Language Autoencoding for Feature Extraction	73
5.1	Extracting Contextualized Language Representations	73
5.1.1	Overview	73
5.1.2	Contributions	74
5.2	Related Works	76
5.2.1	Contextualization in Language Representations	76
5.2.2	Word-level VS Sentence-level Representations	76
5.3	Experiments on Unsupervised Learning Tasks	78
5.3.1	Language Model Setups	78
5.3.2	Settings: Unsupervised STS	78
5.3.3	Results: Unsupervised STS	80
5.3.4	Settings: Unsupervised WiC	83
5.3.5	Results: Unsupervised WiC	84
5.4	Experiments on Supervised Learning Tasks	86
5.4.1	Language Model Setups	86
5.4.2	Settings: Text Classification Tasks	87
5.4.3	Results: Feature Extraction	88
5.4.4	Results: Fine-tuning Approach	90
5.5	Summary of BLA for Feature Extraction	92

6 Conclusions and Future Works	93
6.1 Future Works	94
Abstract (In Korean)	109
Acknowledgement	111

List of Tables

1.1	Comparison of each language modeling	5
3.1	Oracle WERs of the 100-best lists on LibriSpeech	38
3.2	WERs for unidirectional and bidirectional SANLMs interpolated with the baseline model on LibriSpeech	40
4.1	Accuracy of each language model for each training task. Plain texts of the test-clean set of LibriSpeech ASR Corpus are used in this experiment.	58
4.2	Oracle WERs of the 50 best lists on LibriSpeech from <i>Seq2Seq_{ASR}</i>	63
4.3	Oracle WERs of the 50 best lists on LibriSpeech from <i>Seq2Seq_{ASR+}</i>	64
4.4	Oracle BLEU scores of the 50 best lists on WMT13	64
4.5	WERs after re-ranking with each language model on LibriSpeech	66
4.6	BLEU scores after re-ranking with each language model on WMT13	67
4.7	(pseudo)Perplexities and corresponding WERs of the language models on LibriSpeech.	69
4.8	WERs after re-ranking with each large-size language model on LibriSpeech	70
5.1	Pearson’s $r \times 100$ results on the STS-B dataset	78
5.2	Pearson’s $r \times 100$ results on the STS-B dataset	81
5.3	Pearson’s $r \times 100$ results on the SICK dataset	82
5.4	Accuracy on the WiC dataset	84

5.5	Accuracy on each pre-training task.	87
5.6	Accuracies on text classification tasks.	89
5.7	Accuracies on text classification tasks.	91
6.1	Comparison of bidirectional language modeling objectives.	94

List of Figures

2.1	Representing a word from one-hot encoding to a dense vector.	9
2.2	Architectures of (a) CBOW and (b) skip-gram.	10
2.3	Schematic diagrams of a (a) forward and (b) backward RNNLMs. . .	13
2.4	Schematic diagram of ELMo.	14
2.5	Schematic diagram of SANLM.	15
2.6	Schematic diagram of MLM. Dashed arrows denote not used during MLM training.	17
2.7	Illustration of input representations of BERT.	18
2.8	Schematic diagram of PLM. Dashed arrows denote that the only posi- tion information without token information is forwarded to the upper layer. Note that the sequence order is not permuted, and input and output of PLM is the same instead of delayed. To achieve permutation of the factorization order $3 \rightarrow 2 \rightarrow 4 \rightarrow 1$, the model should rely on a proper attention mask in Transformer.	20
2.9	Architecture of the Transformer model.	22
2.10	(left) Scaled dot-product attention and (right) multi-head attention. . .	24
3.1	Architectures of (a) the self-attention network language model and (b) the scaled dot-product attention.	31
3.2	Schematic diagram of the SANLMs.	33
3.3	Example of procedure for re-ranking using BERT on ASR.	36

3.4	Error count by word position.	41
4.1	Schematic diagram of SAN-based (a) uniLM and (b) biLM.	48
4.2	Schematic diagram of a one-layer language model for BLA.	50
4.3	Example of loosing self-unknown property with a multi-layer language model for BLA.	51
4.4	Architecture of our T-TA. The highlighted box and dashed arrows are the innovations presented in this dissertation.	52
4.5	Diagonal masking of the scaled dot-product attention mechanism. The highlighted box and dashed arrow represent the innovations reported in this dissertation.	53
4.6	Schematic diagram of T-TA for BLA.	54
4.7	Average runtimes of each model according to the number of words on the re-ranking task.	59
4.8	Runtimes according to the number of words for the uniLM and T-TA.	60
4.9	Runtimes according to the number of words in the GPU-augmented environment.	60
4.10	Example of procedure for re-ranking using BERT on ASR.	61
5.1	Language autoencoding with (a) sequence-to-sequence model and (b) bidirectional encoder.	76
5.2	Example of procedure for unsupervised STS.	80
5.3	Example of procedure for unsupervised WiC.	83
5.4	Feature extraction with language models for text classification tasks. .	89
5.5	Fine-tuning approach for text classification tasks.	90
6.1	Conditional MLM score for text evaluation.	95
6.2	Generator with MLM for pre-training discriminator.	96
6.3	Illustration of BLA for long texts.	96

Chapter 1

Introduction

1.1 Overview

In natural language processing (NLP), extracting useful features from raw text data is one of the most fundamental techniques and, at the same time, one of the most challenging tasks. Among others, word embeddings is the most popular technique for representing words as feature vectors in a continuous space, and it has been established as a more efficient and effective input representation rather than one-hot encoding or term frequency-inverse document frequency. Up to this day, word embeddings has been a basic input unit of state-of-the-art deep learning models for various NLP tasks.

In particular, word embeddings has received great attention in that word features can be trained with unsupervised learning or self-supervised learning by using co-occurrence information of words appearing in a large text corpus [1, 2]. Transferring pre-trained word embeddings to downstream tasks has achieved performance improvements showing unsupervised learning helps the generalization ability of networks. Furthermore, pre-trained word embeddings can capture semantic by themselves, as in the following examples.

Male-Female: $x_{king} - x_{man} + x_{woman} \approx x_{queen}$

Verb Tense: $x_{swimming} - x_{swam} + x_{walked} \approx x_{walking}$

Country-Capital: $x_{Seoul} - x_{Korea} + x_{Beijing} \approx x_{China}$

Like this, unsupervised representation learning in NLP is helpful to not only improve performance in downstream tasks but also summarize and understand the meaningful features of text data or natural language.

However, word embeddings has a major limitation: words with multiple meanings are compressed into a single representation. In other words, word embeddings alone cannot reflect the varying sense of words, and it does not distinguish the meaning of words in context. For example, an “apple” can be a fruit or a company name, depending on a given context. Similarly, a “bat” can be an animal or a baseball tool, but it cannot be distinguished without any context.

In this light, deep neural networks are popular language learners because they are able to incorporate context information on top of word embeddings in a flexible way. Since making good use of context information boosts the performance in natural language processing tasks, various kinds of neural networks such as fully-connected neural networks (FNNs) [3, 4, 1], recurrent neural networks (RNNs) [5, 6, 7], convolutional neural networks (CNNs) [8], and self-attention networks (SANs) [9, 10, 11] have been proposed. As each network has pros and cons in a relative sense, there have been many studies on designing model architectures to compensate for each other. For example, RNN is good to encode global context, whereas CNN is good to encode local context. Therefore, stacking CNNs followed by RNNs may be helpful to capture local-to-global information by taking advantage of both networks [12]. Indeed, it has been a ubiquitous approach to customize model architectures to obtain better contextual language representations in many NLP tasks [13].

However, contextualized representation learning in a supervised manner has a major limitation in that it requires a lot of labeled samples. Otherwise, language representations are vulnerable to new patterns of contexts that have not been encountered

during training. Therefore, it is desirable that contextualized language representations can be trained with unsupervised learning on a large corpus.

Recently, NLP has taken a new turn with the success of pre-training contextualized language representations in an unsupervised manner. The success has been accelerated by the use of unsupervised learning called *language modeling* based on deep neural networks. In language modeling, *autoregressive language modeling* (ALM) and *masked language modeling* (MLM) are two main learning objectives.

At first, ALM is to predict the next word in a sentence given previous words. More formally, given a text sequence $\mathbf{x} = [x_1, \dots, x_T]$, ALM is the task of assigning the likelihood by forward product $p(\mathbf{x}) = \prod_{t=1}^T p(x_t|\mathbf{x}_{<t})$ or backward product $p(\mathbf{x}) = \prod_{t=T}^1 p(x_t|\mathbf{x}_{>t})$. The output of the neural network language model that learns ALM is transferred to downstream tasks in general. Embeddings from Language Models (ELMo) is the representative model in this learning objective. In ELMo, bidirectional RNNs are pre-trained with the forward and backward language modelings, respectively, and contextual word representations are extracted by concatenating hidden representations of each language model for each word. In this way, ELMo provides contextual word representations instead of (or in addition to) non-contextual word embeddings and achieves performance improvements in many language understanding tasks. However, it has been pointed out that ALM is less effective in deep bidirectional context modeling because it only utilizes unidirectional context during pre-training.

On the other hand, MLM is another famous learning objective that can learn deep bidirectional language representations during pre-training. In the MLM task, some (usually 15%) words in a given text are randomly replaced with a special token [MASK], and the model predicts the original words at the masked positions in the corrupted input. Specifically, MLM is the task of computing probabilities of masked words $\{p(x_t|\hat{\mathbf{x}})_{m_t}\}_{m_t=1}$ with the assumption that masked words are independent of each other given the corrupted input text $\hat{\mathbf{x}}$, where $m_t = 1$ indicates x_t is masked. In this learning objective, Bidirectional Encoder Representations from Transformers (BERT)

is the representative model. BERT and its variants have achieved huge successes on various downstream tasks when transferring pre-trained models and fine-tuning the entire network. In this pre-training and fine-tuning approach, MLM has been a fundamental learning objective on state-of-the-art pre-training techniques that have been explored continually. Indeed, after BERT was recently proposed, pre-training contextualized language representations on a large corpus is in a spotlight [11, 14, 15, 16, 17, 18, 19, 20].

However, the use of word-level contextualized language representations for unsupervised learning tasks has stayed less mature despite its importance. In other words, much research on extracting contextualized word representations using deep bidirectional language models as generic linguistic features has not been sufficiently explored. As ELMo did, studies of extracting contextualized word representations without fine-tuning the pre-trained network and customizing the task-specific model on top of extracted language features deserve further exploration.

As mentioned above, deep neural networks learned with ALM are known to be insufficient to use deep contextualized word representations due to their unidirectional nature. In comparison, MLM is the pre-training objective that enables deep and bidirectional language understanding. However, this MLM has several limitations to be used for unsupervised learning or few-shot learning tasks, which do not have a lot of labeled samples. At first, it is necessary to construct an artificial noise with [MASK] in the input to learn MLM. When the pre-trained model is used as the feature extractor, there is a training-inference discrepancy that [MASK] does not exist in input text at all during inference. The severer limitation of this input corruption is that training signals occur only at the 15% masked positions, and nothing happens at the rest of 85% unmasked positions. Specifically, contextual word representations are computed at masked positions to predict the original word, but the hidden representations at unmasked positions are computed not for themselves but for predicting original word ids of masked tokens. These points make it hard to guarantee which representation

will be computed at a specific position for an input text without [MASK]. Therefore, pre-trained networks with MLM appear to have unstable performance when utilized as feature extraction.

Table 1.1: Comparison of each language modeling

Property	ALM	MLM	BLA
Understanding of Bidirectional Context	△	○	○
Application on Supervised Learning Tasks	△	○	△
Application on Unsupervised Learning Tasks	○	△	○

This dissertation deals with an approach for extracting fully contextualized word representations with deep bidirectional language models learned with self-supervised learning. This dissertation introduces a new learning objective called *bidirectional language autoencoding* (BLA) and discusses its effectiveness with differences from the aforementioned learning objectives. Like MLM, BLA allows deep bidirectional language understanding during (pre-)training. Like ALM and word embeddings, the proposed learning objective allows the model to extract useful linguistic features at the word-level and be used for unsupervised learning tasks. In retrospect, the MLM-trained model seems similar to the denoising autoencoder in the image domain, but there is a critical difference that the bottleneck to prevent copy is not in the model but only in the input. As mentioned before, the fact that learning only occurs at masked positions in the MLM task makes it difficult for the model to be called (denoising) autoencoder. The proposed BLA, on the other hand, follows the traditional autoencoder that has a bottleneck in the model and reconstructs the input as its name. Like an image autoencoder, the encoder of the model that learns the proposed learning objective could be used for extracting fully contextualized word representations.

1.2 Contributions and Outline of This Dissertation

The majority of the recent deep learning research in NLP is focused on developing pre-training algorithms for fine-tuning the whole networks on supervised learning tasks. More precisely, their fine-tuning oriented unsupervised or self-supervised learning algorithms are not effective when used as feature extraction. In contrast, BLA we propose in this dissertation has opened a new phase in deep contextualized word representations, enabling deeper exploration into the future usability of feature extraction. Since unsupervised linguistic feature learning in NLP helps boost task performance and uncover interesting patterns, I decided to investigate further the space of extracting deep bidirectional language representations with deep neural networks.

To this end, this dissertation compares the pros and cons of utilizing two popular language modeling objectives, ALM and MLM, for unsupervised learning tasks. This dissertation points out that ALM lacks in the bidirectionality for text understanding. While MLM is better for bidirectional language understanding than ALM, MLM suffers the training-inference discrepancy when extracting contextualized word representation. To overcome the limitation of MLM, this dissertation presents BLA, which is fully bidirectional and does not suffer from the training-inference discrepancy. Therefore, a major contribution of this dissertation is to provide a better language modeling framework for unsupervised learning tasks that need bidirectional language understanding. The outline of this dissertation is below:

Firstly, Chapter 2 briefly reviews language representation models ranging from the basic word embedding models to the recent bidirectional language representation models like BERT. While this chapter lightly mentions advanced pre-training algorithms such as permutation language modeling [16], sequence-to-sequence language modeling [17, 19, 20, 21], and multilingual or multimodal pre-training [15, 22], all they are based on masked language modeling and/or autoregressive language modeling, which is out of the scope of this dissertation.

Chapter 3 presents a method to use MLM for sentence scoring, which is to mea-

sure how natural a given sentence is grammatically and semantically [23]. This task is the most direct application of language models that compute the (pseudo-)likelihood of a sentence. When applied to the N -best list re-ranking, the proposed MLM-based sentence scoring method outperforms the ALM-based one in terms of accuracy. This is the first empirical demonstration that bidirectional language models outperform unidirectional ones for the re-ranking task on the LibriSpeech ASR Corpus [24].

In Chapter 4, however, this dissertation argues that the MLM-based method suffers a problem of growth in computational complexity since it requires mask-and-predict repetition as many as the number of words in a sequence [25]. This chapter introduces BLA and compares it with other language modeling objectives ALM and MLM to solve this limitation. To learn the proposed BLA, this chapter presents a novel deep bidirectional language model called Transformer-based text autoencoder (T-TA). Details of the proposed language model are explained in this chapter with the theoretical demonstration. In experiments, the proposed T-TA is much faster than BERT while maintaining the BERT's accuracy when applying language models to the N -best list re-ranking tasks on ASR and NMT.

Chapter 5 investigates the potential of BLA for extracting contextualized language representations. For various NLP tasks, BLA-trained models show better performances than MLM-trained models when applying language models to feature extraction. The empirical study includes semantic textual similarity, word sense disambiguation [26], and text classification tasks in the GLUE benchmark [27].

Finally, Chapter 6 concludes this dissertation with relative merits of BLA and MLM. In addition, several directions for future research using BLA are provided at the end of this dissertation.

Chapter 2

Background: Language Representation Models

This section reviews several standard language representation techniques from word embedding to BERT.

In general, “word” is considered as a basic unit in natural language. However, the vocabulary size could be problematic when we use the word-level text encoding. Most general solution is using the top- k frequentest words in the total corpus. But it could be still problematic in the specific NLP tasks like named entity recognition (NER) since many unknown words appears in the named entity. Character embedding could be another solution to overcome this out-of-vocabulary issue. However, it may be too fine-grained to keep some important information and it increases the length of the text.

To reduce the vocabulary size but handle unknown word or rare word, there has been proposed several subword-level text encoding methods such as Byte Pair Encoding (BPE) [28], WordPiece [29], and SentencePiece [30]. For example, “subword” can be split into “sub” and “word”, and we use two vectors to represent the “subword”. As you can see in the above example, subword is in between word and character. It seems to use more resource to compute a word, but the reality is that we can use less footprint with the fewer subword vocabulary than word. We note that this dissertation uses a term “token” as a basic input unit, which comprehends word, subword, or even character.

2.1 Non-contextualized Word Representations: Word Embeddings

Word embeddings is a technique for mapping words into a continuous vector space, and it is used as the underlying input representation in the modern NLP tasks. The primary purpose of the use of word embeddings is to reduce the number of dimensions for representing words in a vector form. Compressing a high dimensional and very sparse vector space (*i.e.*, one-hot embedding) into a low dimensional and dense vector space is useful not only to resolve the curse of dimensionality problem but also to express words since it is possible to give semantics that was impossible in random indexing. Indeed, when used as the input representations, word embedding has been shown to boost the performance in NLP tasks such as sentiment analysis.

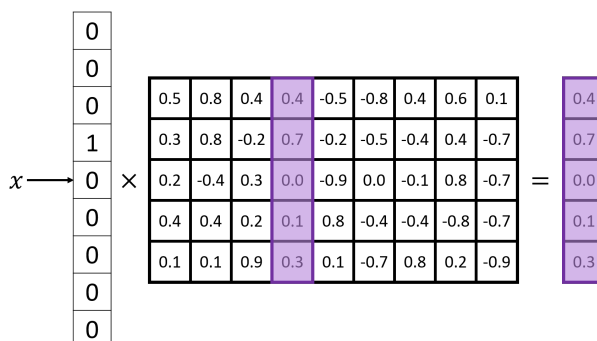


Figure 2.1: Representing a word from one-hot encoding to a dense vector.

Word vector for each word or token x of input sequence can be represented as following:

$$e(x) = I(x)W_V, \quad (2.1)$$

where $W_V \in \mathbb{R}^{V \times d}$ is an embedding matrix for the total vocabulary, $I(x) \in \mathbb{R}^{1 \times V}$ is a function for the one-hot encoding, where only a single digit in the whole vector is one and the rest are zero. Practically, we use an embedding lookup table for the efficient computation.

Word embeddings can be trained both in task-agnostic and task-specific fashions. Among them, training task-agnostic word embeddings is more broadly used since it further boosts by capturing generic word meanings learned from a large text corpora without any label. Once trained, it can be used as a backbone embedding for any NLP task. The pre-training of word embeddings is especially useful when the task-specific labeled data is insufficient.

We review a representative algorithm for pre-training word embeddings known as word2vec introduced by [1, 4]. The word2vec algorithm uses a simple neural network model to learn word associations from a large text corpus. Word2vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or skip-gram.

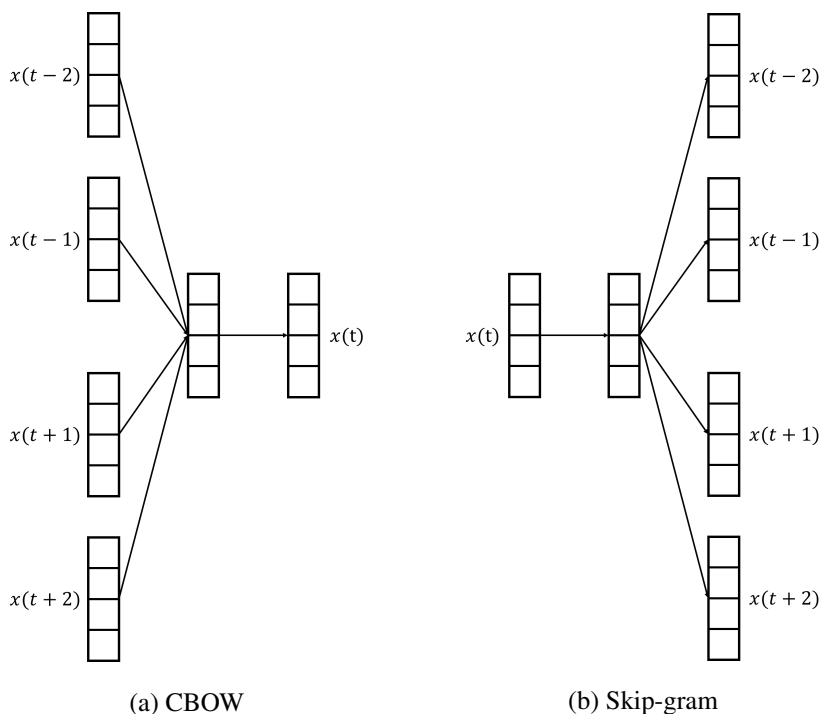


Figure 2.2: Architectures of (a) CBOW and (b) skip-gram.

In the CBOW architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction

as the bag-of-words assumption. More formally, given a training words x_1, \dots, x_T , the objective of the CBOW model is to maximize the average of log probability:

$$\frac{1}{T} \sum_{t=1}^T \log p(x_t | x_{t-c}, \dots, x_{t-1}, x_{t+1}, \dots, x_{t+c}), \quad (2.2)$$

where c is the size of the training context, $p(x_t | x_{t-c}, \dots, x_{t-1}, x_{t+1}, \dots, x_{t+c})$ is defined by the softmax function, and context vectors are projected into the same position by averaging them.

In the skip-gram architecture, the model uses the current word to predict the surrounding window of context words. More formally, the objective of the skip-gram model is to maximize the average of log probability:

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(x_{t+j} | x_t), \quad (2.3)$$

where $p(x_{t+j} | x_t)$ is also defined by the softmax function. To reflect the assumption that the more distant words are usually less related to the current word than those close to it, skip-gram assigns weights to nearby context words more heavily than more distant context words. According to the authors' note, CBOW is faster while skip-gram is slower but does a better job for infrequent words.

This dissertation omits details of other popular techniques for pre-training word embeddings such as GloVe [2] and FastText [31]. Instead, we note that those techniques including word2vec do not capture contextual meaning themselves. By applying neural networks that can treat sequential input over the trained word embedding can capture contextual meaning, and it actually improves task-specific accuracy in many supervised learning tasks.

However, training additional neural networks needs a lot of task-specific labels, which is laborious to collect. As data sparsity is a major problem in building deep neural networks for most NLP tasks, learning contextual language representations from the supervised learning tasks might be insufficient. To handle this problem, following

sections deal with more advanced self-supervised learning techniques for pre-training contextual language representations.

2.2 ALM-based Language Representation Models

To pre-train contextual language representations, most methods are based on a language modeling objective, which is simple but powerful tools for learning contextualization in a task-agnostic fashion.

As mentioned earlier, autoregressive language modeling (ALM) is one of the most successful pre-training objectives. More formally, given a text sequence $\mathbf{x} = [x_1, \dots, x_T]$, ALM is the task of assigning the likelihood by forward product $p(\mathbf{x}) = \prod_{t=1}^T p(x_t | \mathbf{x}_{<t})$. Neural network language models (LMs) performs pre-training by maximizing the likelihood under the forward ALM:

$$\max_{\theta} \log p_{\theta}(\mathbf{x}) = \sum_{t=1}^T \log p_{\theta}(x_t | \mathbf{x}_{<t}) = \sum_{t=1}^T \log \frac{\exp(h_{\theta}(\mathbf{x}_{1:t-1})^{\top} e(x_t))}{\sum_{x'} \exp(h_{\theta}(\mathbf{x}_{1:t-1})^{\top} e(x'))}, \quad (2.4)$$

where $h_{\theta}(\mathbf{x}_{1:t-1})$ is a context representation produced by neural models, such as RNNs or SANs, and $e(x)$ denotes the embedding of x . Figure 2.3a shows an example of the autoregressive language model based on RNN.

Theoretically, RNN can process an infinite number of states. However, vanilla RNN suffers from gradient vanishing and gradient explosion problems, and thus it hardly capture contexts in a long sequence. To resolve these problems and effectively use longer contexts, RNN states have been replaced by gated states such as Long Short-Term Memory (LSTM) [32, 6] and Gated Recurrent Unit (GRU) [7]. Actually, these RNN variants became very popular after the invent of these controlled states since they significantly outperformed in many NLP tasks, including language modeling tasks. However, this dissertation omits specific formulations on these sophisticated RNNs because they are out-of-scope of this dissertation.

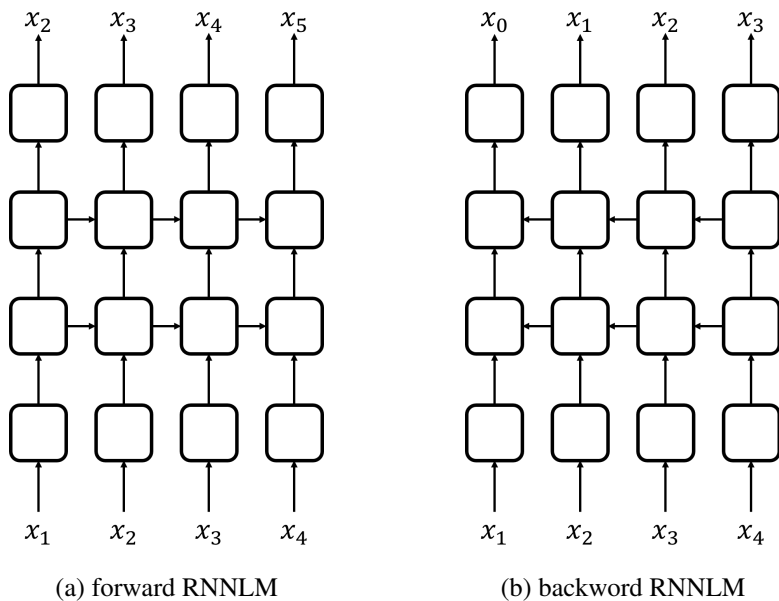


Figure 2.3: Schematic diagrams of a (a) forward and (b) backward RNNLMs.

2.2.1 ELMo

Since an autoregressive language model is only trained to encode a unidirectional context, it is not effective at modeling deep bidirectional contexts. On the contrary, downstream language understanding tasks such as named entity recognition [33] and question answering [34] often require bidirectional context information.

To fill this gap, ELMo (abbreviation of Embeddings from Language Models) was proposed to use bidirectional context by training two RNN language models (RNNLMs) with an additional backward product $p(\mathbf{x}) = \prod_{t=T}^1 p(x_t | \mathbf{x}_{>t})$, shown in Figure 2.3b, as well as the forward product. Specifically, their bidirectional RNNLM (biRNNLM) is trained to jointly maximize the log likelihood of the forward and backward directions:

$$\sum_{t=1}^T (\log p(x_t | x_1, \dots, x_{t-1}; \Theta_x, \Theta_f, \Theta_s) + \log p(x_t | x_{t+1}, \dots, x_T; \Theta_x, \Theta_b, \Theta_s)), \quad (2.5)$$

where Θ_x and Θ_s are tied parameters for word embedding and softmax layers respectively, and Θ_f and Θ_b are separate parameters for the RNNs in each direction.

Once trained, ELMo computes contextual word representations by concatenating each hidden layer of bidirectional RNNs and then computing weighted sum of all layers. More formally, the t -th token representation in ELMo is follows:

$$\mathbf{ELMo}_t = \gamma \sum_{l=0}^L s_l \mathbf{h}_{t,l}, \quad (2.6)$$

where $\mathbf{h}_{t,l} = [\mathbf{h}_{t,l}^f; \mathbf{h}_{t,l}^b]$, $\mathbf{h}_{t,l}^f$ and $\mathbf{h}_{t,l}^b$ are forward and backward vectors in the l -th layer, $\mathbf{h}_{t,0}$ means the token embedding, are $\mathbf{s} = [s_0, \dots, s_l]$ are softmax-normalized weights. Figure 2.4 shows the process to get the entire ELMo vectors. ELMo vectors are often scaled by a scalar task-specific parameter γ to aid the optimization process.

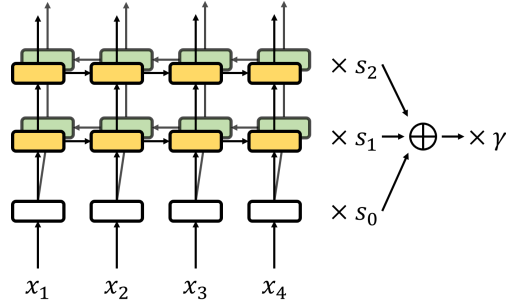


Figure 2.4: Schematic diagram of ELMo.

Note that the pre-trained weights for ELMo vectors are fixed when applied to downstream tasks. By constructing task-specific networks on top of ELMo vectors, just task-specific network parameters including \mathbf{s} and γ are trained to the target task. This is called *feature-based* approach in transfer learning. Since ELMo learns ALM, its contextualized word representations can be used directly without fine-tuning pre-trained weights. In other words, ELMo works as a linguistic feature extractor, which is a desirable property for applying on unsupervised learning tasks.

2.2.2 GPT

GPT [10], the abbreviation for Generative Pre-Training, is another model that learns contextualized word representations with ALM. Its training scheme is not so special.

However, GPT uses SAN based language models (SANLMs) rather than RNN, and it stacks multiple SAN layers, more than or equal 12 layers, for capturing linguistic features in a large text corpus. The configuration of SANLMs is slightly different from RNNLMs as shown in Figure 2.5. Note that GPT uses the masking operation with an

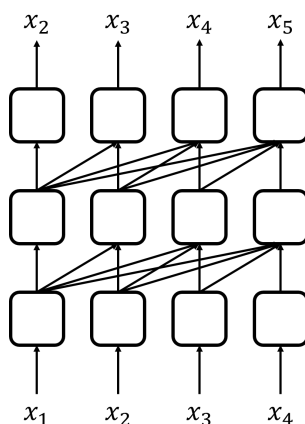


Figure 2.5: Schematic diagram of SANLM.

upper triangular matrix in the self-attention mechanism for prevent to see the future input, just like in the masked self-attention in the Transformer decoder.

To use the pre-trained network, fine-tuning includes minimal modification for the target task. Unlike feature-based approach, the whole network parameters are also updated during the fine-tuning stage of GPT, and this is called the *fine-tuning* approach. GPT-based models often outperforms many state-of-the-art models including ELMo on many downstream tasks. With its deeper architecture whilst unidirectional language modeling, GPT can understand more context than ELMo, which consists of only 2 layers. Rather than building complex network blocks with random initialization on top of shallow pre-trained models, building simple task-specific networks on top of deep pre-trained networks is empirically proven to be more powerful. Besides, using BPE, the subword-level vocabulary, instead of the word-level vocabulary is another difference between GPT and ELMo.

There have been more developments of GPT such as GPT-2 [35] and GPT-3 [36].

Rather than supervised learning tasks, they focus more on unsupervised learning and few-shot learning settings, showing that bigger models, larger data, and longer training are important for unsupervised or few-shot learning. This is because the main training task of GPT series is still ALM, and thus they only use unidirectional context. While advanced GPT focuses no longer on supervised learning tasks and the fine-tuning approach, the first GPT promotes the fine-tuning approach and gives a motivation to BERT.

2.3 MLM-based Language Representation Models

2.3.1 BERT

BERT [37] is the abbreviation of Bidirectional Encoder Representations from Transformers, and is based on the Transformer encoder (Figure 2.9), as in the name. BERT marked a milestone in the recent NLP thanks to its simple and effective pre-training method for improving the fine-tuning approach. The originality of BERT is its pre-training objective, called masked language modeling (MLM) which is inspired by the Cloze task [38]. The MLM task randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked token based only on its context. Specifically, they replace 15% of the tokens with [MASK], which is a newly introduced special symbol only used for pre-training stage. To fill the gap between pre-training and fine-tuning of the emergence of [MASK], 13% of the input tokens are changed to [MASK], 1.5% of the input tokens are changed to a random token, and 1.5% of the masked tokens remains unchanged but training occurs on all 15% of the masked positions.

In other words, BERT is similar to the denoising auto-encoding. More formally, for a text sequence \mathbf{x} , BERT first constructs a corrupted version $\hat{\mathbf{x}}$ with the masked

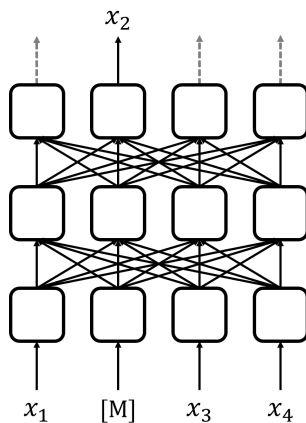


Figure 2.6: Schematic diagram of MLM. Dashed arrows denote not used during MLM training.

tokens $\bar{\mathbf{x}}$. The training objective is to reconstruct $\bar{\mathbf{x}}$ from $\hat{\mathbf{x}}$

$$\max_{\theta} \log p_{\theta}(\bar{\mathbf{x}}|\hat{\mathbf{x}}) \approx \sum_{t=1}^T m_t \log p_{\theta}(x_t|\hat{\mathbf{x}}) = \sum_{t=1}^T m_t \log \frac{\exp(H_{\theta}(\hat{\mathbf{x}})_t^{\top} e(x_t))}{\sum_{x'} \exp(h_{\theta}(\hat{\mathbf{x}})_t^{\top} e(x'))}, \quad (2.7)$$

where $m_t = 1$ indicates x_t is masked, and H_{θ} is a Transformer that maps a length- T text sequence \mathbf{x} into a sequence of hidden vectors $H_{\theta}(\mathbf{x}) = [H_{\theta}(\mathbf{x})_1, \dots, H_{\theta}(\mathbf{x})_T]$.

Unlike left-to-right (autoregressive) language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer. While autoregressive language models like GPT is also successful, but unidirectional restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions. Although ELMo uses the left and right context during the fine-tuning stage, separate pre-training of the left-to-right and the right-to-left language models is limited to the shallow fusion.

To make the BERT model to understand sentence relationship in a self-supervised learning, BERT has another pre-training objective called next sentence prediction (NSP). This is because many important downstream tasks such as question answering and

natural language inference are based on understanding the relationship between two sentences, which is not directly captured by language modeling. Specifically, NSP is to predict whether the subsequent sentence is the real next sentence or a randomly sampled one. Although NSP is controversial to the robustness of BERT [14], distinguishing two input sentences in the bottom input representations is still universal. To learn sentence relationship in the self-supervised learning, follow-up researches have proposed other tasks such as sentence order prediction [39].

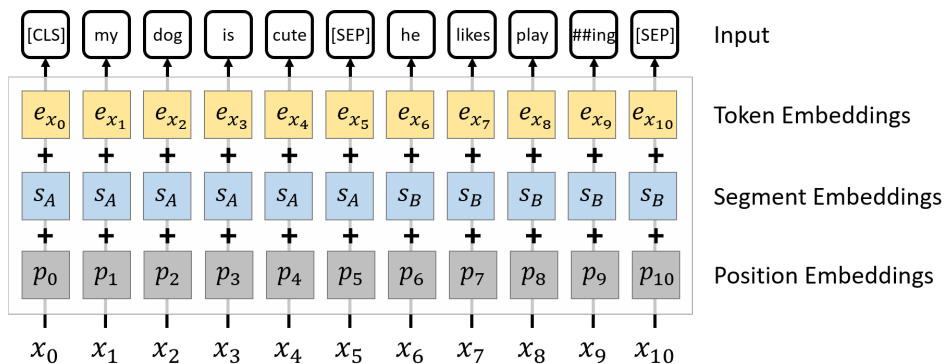


Figure 2.7: Illustration of input representations of BERT.

Figure 2.7 shows that input representations of BERT are the sum of token embeddings, segment embeddings and position embeddings, where all embeddings are trainable parameters. Segment embeddings are invented for distinguishing two input sentences and learning sentence relationship. [CLS] and [SEP] symbols are special tokens to represent the start of the document and the end of the sentence, respectively. Similar to GPT, BERT uses subword vocabulary, WordPiece, rather than word-level vocabulary. For example in Figure 2.7, the word “playing” is divided into “play” and “ing”, and a special symbol “##” in front of “ing” is for showing the middle of the word and distinguish with the space character.

BERT showed that pre-trained representations reduce the need for many heavily-engineered task-specific architectures or hand-crafted linguistic features. Since BERT achieved tremendous performance improvements on many downstream tasks, the fine-

tuning approach has been major trend in NLP rather than the feature-based approach.

2.3.2 Other Language Representation Models

The MLM objective allows the model to be pre-trained to better capture bidirectional context than ALM. However, the input to BERT contains artificial symbols like [MASK] that never occur in downstream tasks, which creates a pretrain-finetune discrepancy, where ALM has no such discrepancy. In addition, MLM factorizes the joint conditional probability $p(\bar{\mathbf{x}}|\hat{\mathbf{x}})$ based on an independence assumption that all masked tokens $\bar{\mathbf{x}}$ are separately reconstructed. In comparison, the ALM objective factorizes $p_\theta(x)$ using the product rule that holds universally without such an independence assumption. As discussed above, ALM and MLM possess their unique advantages over the other.

To integrate the advantages of ALM and MLM, a new language model, XLNet, introduces another novel language modeling task called permutation language modeling (PLM) [16]. Specifically, for a sequence x of length T , there are $T!$ different orders to perform a valid autoregressive factorization. Intuitively, if model parameters are shared across all factorization orders, in expectation, the model will learn to gather information from all positions on both sides. More formally, the PLM objective can be expressed as follows:

$$\max_{\theta} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_T} \left[\sum_{t=1}^T \log p_\theta(x_{z_t} | \mathbf{x}_{\mathbf{z}_{<t}}) \right], \quad (2.8)$$

where \mathbf{z} is a sampled permutation in all possible permutation \mathcal{Z}_T , z_t and $\mathbf{z}_{<t}$ are the t -th token and the first $t - 1$ tokens of the permutation.

Note that the PLM objective only permutes the factorization order, not the sequence order, as shown in the Figure 2.8. Namely, XLNet keeps the original sequence order, uses the positional encodings corresponding to the original sequence, and relies on a proper attention mask in Transformers to achieve permutation of the factorization order. This choice is necessary, since the model will only encounter text sequences with the natural order during fine-tuning. While XLNet also introduces architectural mod-

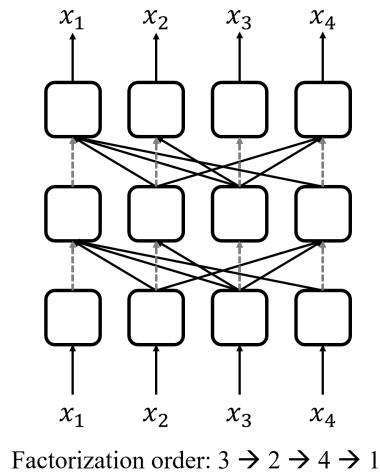


Figure 2.8: Schematic diagram of PLM. Dashed arrows denote that the only position information without token information is forwarded to the upper layer. Note that the sequence order is not permuted, and input and output of PLM is the same instead of delayed. To achieve permutation of the factorization order $3 \rightarrow 2 \rightarrow 4 \rightarrow 1$, the model should rely on a proper attention mask in Transformer.

ifications based on self-attention networks such as two-stream self-attention to learn PLM, we do not include those contents because they are too many details unrelated to this dissertation.

In addition to GPT, BERT, and XLNet, pre-training deep and bidirectional language representations in the self-supervised learning has been widely adopted in the NLP domain. For example, there have been advanced pre-training techniques such as RoBERTa [14] and ELECTRA [18], but their training objectives are still based on the BERT’s MLM. Similarly, there have been extended approaches such as XLM [15], UniLM [17], MASS [19], BART [20], and T-5 [21], but their targets are pre-training sequence-to-sequence models and their training objectives are the variants of MLM. Although building deeper networks, using larger text corpus, and training with larger batch size are empirically proven to be important in the literature, those problems are out of the scope of this dissertation.

However, this dissertation argues that MLM is only for the fine-tuning approach. Due to the training-inference discrepancy, caused by the artificial [MASK] token during training, MLM is limited to the fine-tuning approach, which requires a lot of labeled samples for supervised learning tasks.

2.4 Base Language Model Architecture: SAN

As this dissertation mainly deals with SAN based language models throughout the dissertation, this section reviews the architecture of the Transformer model [9], which is a sequence-to-sequence model originally developed for the task of neural machine translation (NMT).

As most competitive neural machine translation models, Transformer has an encoder-decoder architecture. Both the encoder and decoder have stacked self-attention and point-wise fully connected layers, shown in Figure 2.9. A residual connection [40] around each of sub-layers followed by layer normalization [41, 42] is employed to im-

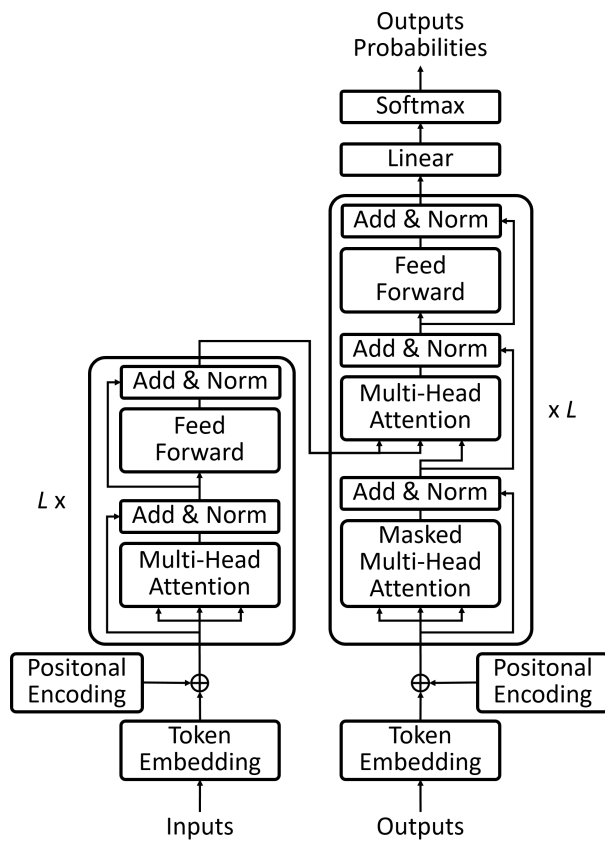


Figure 2.9: Architecture of the Transformer model.

prove the training. Namely, the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$. To enable these residual connections, all sub-layers as well as the embedding layers produce outputs of the same dimension throughout the model.

In addition to the two sub-layers in each encoder layer, the decoder has a third sub-layer, called an encoder-decoder attention, that performs attention over the output of the encoder. To prevent positions from attending to subsequent positions and keep the autoregressive property, the self-attention sub-layer in the decoder is modified. Specifically, masking the upper triangular matrix is employed to ensure that the predictions for position t can depend only on the previous (known) outputs at positions less than t .

Generally speaking, each layer of SAN means the encoder layer of Transformer, the left part of Figure 2.9. As its name shows, the key function of SAN is in its attention function. An attention function can be described as mapping a query (Q) and a set of key-value (K - V) pairs into an output, where the query, keys, values, and the output are all vector sequence. It computes a weighted sum of the values as the output, where the weight of each value is obtained by a similarity function of the query with the corresponding key.

The SAN uses the scaled dot-product attention as its attention mechanism (Figure 2.10). Computation of the output matrix of this attention is:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \quad (2.9)$$

where the dot-product attention is scaled by $\frac{1}{\sqrt{d}}$ to normalize the gradients since this scaling factor prevents performance drop with the large values of the hidden dimension d .

Furthermore, instead of performing a single attention function, multi-head attention is employed. Multi-head attention is beneficial to allow the model to jointly attend to information from different representation sub-spaces at different positions. The final

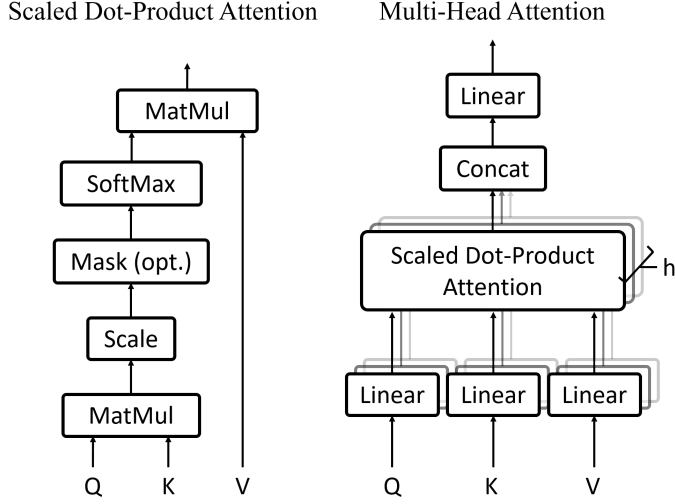


Figure 2.10: (left) Scaled dot-product attention and (right) multi-head attention.

attention is followed as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O, \quad (2.10)$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, parameter matrices $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d_h}$, and $W^O \in \mathbb{R}^{hd_h \times d}$. To prevent the increase of computational cost, $d_h = d/h$ is used.

Each of the SAN layers contains the position-wise feed-forward neural networks, which is applied to each position separately and identically. Two linear transformations with a ReLU activation is used as following:

$$\text{FNN}(x) = f(xW_1 + b_1)W_2 + b_2, \quad (2.11)$$

with $W_1 \in \mathbb{R}^{d \times d_f}$, $W_2 \in \mathbb{R}^{d_f \times d}$, and biases b_1 and b_2 , where $d_f = 4d$ in general. In the right-hand-side of the above equation, f is a non-linear activation function such as sigmoid, tanh, Rectified Linear Unit (ReLU) [43], Exponential Linear Unit (ELU) [44], and Gaussian Error Linear Unit (GELU) [45].

To make use of the order of the text sequence, SAN based model needs to be injected some information about the relative or absolute position of the tokens in the

sequence. In order to that, positional encodings, having the same dimension d as the embeddings, are added to the input embeddings at the bottoms of the model, shown in Figure 2.9. While the original Transformer model uses fixed (sinusoidal) positional encodings for NMT, most SAN-based language models used trainable positional encodings.

When comparing SAN with RNN and CNN in several aspects, SAN has some benefits on computational complexity per layer and parallelization of computation. In terms of computational complexity, SAN layer is faster than RNN layers as well as CNN layers when the sequence length T is smaller than the representation dimension d , which is most often the case with NLP tasks. For parallelization property, SAN and CNN can be fully parallelized due to the constant number of sequential operations, but a RNN requires $O(T)$ sequential operations.

Although conventional language models are based on RNN, self-attention network (SAN) LMs have recently shown competitive performance on sequence modeling with a slight trade-off between speed and accuracy [46, 47]. More recently, SANLMs have drawn a big interest in many NLP communities since BERT [37] was proposed and achieved state-of-the-art performances on many NLP tasks. As GPT-2 is also based on the Transformer decoder, and shows remarkable performance on text generation.

Chapter 3

Masked Language Modeling for Sentence Scoring

This chapter introduces methods for applying BERT to sentence scoring on automatic speech recognition proposed in the research paper [23]. With the proposed method, this chapter empirically proves that masked language modeling (MLM) performs better than autoregressive language modeling (ALM) on the N -best list re-ranking task, which is the representative unsupervised learning task.

3.1 Accurate Bidirectional LMs

3.1.1 Overview

Language modeling is the task of assigning a probability to word sequence. A language model (LM) is an essential component in recent automatic speech recognition (ASR) systems. Since the LM captures the possibility of any word sequence, it helps to distinguish between words with similar sounds. Conventionally, LMs have been used to predict the probability of the next word given its preceding words. Many state-of-the-art speech recognition systems have achieved performance improvements with these *unidirectional* LMs, including n -gram LMs [48] and recurrent neural network (RNN) LMs [5].

Recently, there have been several studies that use *bidirectional* LMs for ASR in

order to capture the full context rather than just the previous words [49, 50]. They applied their bidirectional LMs for the N -best list re-ranking task, which is the task of selecting the most likely sentence from the hypothesis list that is recognized from acoustic models. Even though bidirectional networks are superior to unidirectional ones in many applications from phoneme classification [51] to acoustic modeling [52], previous bidirectional LMs for ASR did not show their excellence compared to the unidirectional LMs when applying the LMs to the re-ranking. This is because there is no interaction between the past and the future words in the bidirectional LMs, although the words on both sides are used to predict the current word. Namely, the left and the right representations are not fused in the bidirectional LMs since they use a shallow concatenation of independently encoded representations, and it may limit the bidirectional LM’s potential.

The same issue has been addressed by the recently suggested model, BERT (Bidirectional Encoder Representations from Transformers) [37]. In the BERT, the model is mainly trained to predict a masked word from its context in order to enable the model to fuse the left and the right representations, unlike the previous bidirectional LMs. Their work proves the importance of the interaction between the past and the future words in language understanding by achieving significant success on many downstream tasks such as text classification [53] and question answering [34]. Therefore, the BERT is a promising bidirectional LM for the task of the N -best list re-ranking [11].

3.1.2 Contributions

First, we develop an approach of adjusting the BERT to the re-ranking task, and verify the empirical effectiveness of the BERT for ASR. The core idea of our approach is to bridge the gap between training and testing environments. For training, we simplify the training objective and the input pipeline of the original BERT. Specifically, we focus only on the “masked word prediction” task and its relevant pipeline from the original BERT, and discard the “next sentence prediction” task because only one sentence is

taken at inference. For testing, we mask each word one at a time in a given sentence, and then make the bidirectional LM predict the probability of the original word at the masked position from its context. We consider that hiding the target word is essential to obtain the proper probability by preventing the bidirectional LM from getting a meaninglessly high probability of the exposed words. The score of the sentence is obtained by aggregating all the probabilities, and this score is used to re-score the N -best list of the speech recognition outputs. Although it may not be a meaningful sentence probability like perplexity, this sentence score can be interpreted as a measure of naturalness of a given sentence conditioned on the bidirectional LM.

We conduct experiments on the 1000-hour LibriSpeech ASR corpus [24]. We first obtain the N -best hypothesis lists from an acoustic model that we implement, and we use our bidirectional LM for re-ranking them to reduce the final word error rates (WERs). Our bidirectional LM achieves 1.61% and 3.00% absolute reductions in WER on the test-clean and test-other sets, which are the subsets of LibriSpeech corpus, while the WER of the acoustic model is 7.26% and 20.37%. Moreover, we empirically prove that our sentence scoring method that uses bidirectional LM significantly outperforms not only the unidirectional LM but the combination of the forward and backward LMs regardless of the experimental conditions. In addition, an analysis of where WERs occur in a sentence shows that the bidirectional LM is more robust than the unidirectional LM especially when a recognized sentence is short or a misrecognized word is at the earlier part of the sentence. Through these results, we demonstrate that the left and right representations in the bidirectional LM should be fused for scoring a sentence.

It is the first study for empirically demonstrating not only that the bidirectional LM is notably better than the unidirectional LM for the N -best list re-ranking on a practical scenario, but also that the BERT is successfully applied to the unsupervised learning task of measuring the naturalness of a given sentence.

3.2 Related Works

3.2.1 Bidirectional LMs in NLP

In natural language processing (NLP), many bidirectional language models (LMs) have been studied [54, 55, 37]. [54] investigated the training of bidirectional recurrent neural network language models (RNNLMs) using noise contrastive estimation. ELMo (Embeddings from Language Models) [55] used bidirectional LMs in order to obtain the contextualized word representations, which were trained with large plain text. Also, [55] proposed the method of transferring the forward and backward RNNLMs in order to improve the performances on many downstream tasks from text classification [53] to question answering [34].

3.2.2 Bidirectional LMs for ASR

There have been several studies on bidirectional RNNLMs in automatic speech recognition (ASR) [56, 49, 50]. [56] interpolated the scores obtained from the forward and backward RNNLMs, which were trained independently. [49, 50] investigated the training of bidirectional RNNLMs with jointly conditioned on backward and forward representations. However, the bidirectional LMs achieved small or no improvements over their unidirectional LM counterparts for the N -best list re-ranking. To the best of our knowledge, however, no study is conducted on the N -best list re-ranking using BERT. While [11] mentioned the re-ranking using BERT, but they did not conduct experiments in practice.

3.3 Methodology

One of the main interests in this part is to demonstrate the superiority of MLM for sentence scoring over ALM. For a fair comparison of MLM and ALM, the two LMs must have the same architecture. As our bidirectional LM is a variant of the BERT [37], which consists of the encoder of the Transformer [9], we also construct the unidirectional LM based on this self-attention network (SAN). Although recurrent neural networks (RNNs) appear to be a natural choice for language modeling, SANLMs have recently shown competitive performance on sequence modeling with a slight trade-off between speed and accuracy [46, 47]. From these reasons, this paper only considers the bidirectional SANLM (biSANLM) and the unidirectional SANLM (uniSANLM) for the re-ranking tasks.

3.3.1 Architecture of SAN-based LMs

We deal with the language models (LMs) that are based on the self-attention network (SAN) as shown in Figure 3.1. Self-attention is an attention mechanism that computes the representation of a single sequence by relating all positions by themselves. As shown in Figure 3.1b, this computation is done by using the scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (3.1)$$

where Q, K, V are query, key, value matrices respectively, which are generated from the input sequence $X^l \in \mathbb{R}^{T \times d}$ with the number of words T and the input dimension d . To leverage the capacity of the SAN, multi-head self-attention is applied:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O, \\ \text{where } \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \end{aligned} \quad (3.2)$$

$W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d_k}$ and $W^O \in \mathbb{R}^{d_k h \times d}$ are the parameter matrices for projections with the number of heads h , and $d_k = d/h$ is used for reducing the computational

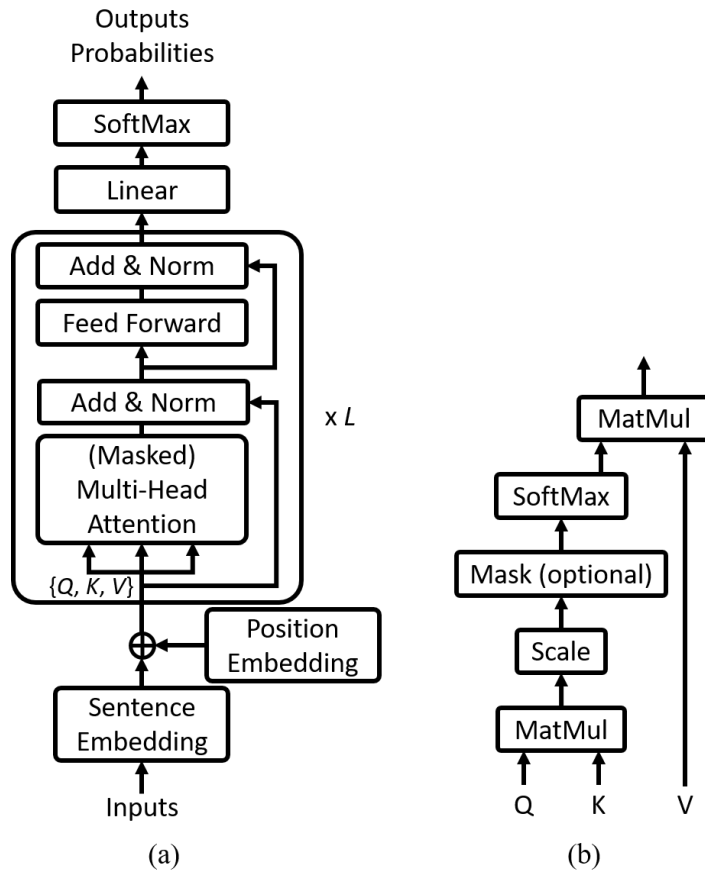


Figure 3.1: Architectures of (a) the self-attention network language model and (b) the scaled dot-product attention.

cost. The position-wise feed-forward network, the layer normalization with the residual connection, and dropout are also used in the SAN module for effective training of the model as in the original Transformer [9].

To construct the conventional unidirectional LM, the optional operation of masking the key-query attention in the scaled dot-product attention should be used. This masking operation prevents words from attending to the future words by making the upper triangle of the key-query attention to be 0 as in the decoder of the Transformer [9]. To make the LM aware of the order of the words in the sequence, the position embedding $X_P \in \mathbb{R}^{T \times d}$ is added to the sentence embedding $X_S \in \mathbb{R}^{T \times d}$ at the bottom of the encoder, and thus we have $X^l = X_S + X_P$ as the input of the LM. On top of this input, we can build an encoder by stacking the SAN layers as many as we want. The output sequence of the highest layer, $Y^L \in \mathbb{R}^{T \times d}$ with the number of layers L , is used to predict the probabilities of the words through the softmax layer with the linear projection.

The unidirectional LM is trained with the “next word prediction” task. Figure 3.2a shows and schematic diagram of the uniSANLM that predicts next word using only its preceding words. We consider the sum of all log-likelihoods of each word in an input sentence as the sentence score of the uniSANLM.

3.3.2 Sentence Scoring with SANLMs

In this section, we present the sentence scoring method which uses bidirectional language model (LM) for re-ranking the N -best list. First, we outline the construction of our bidirectional SANLM (biSANLM) which is a variant of the BERT [37]. we then introduce the procedure of the sentence scoring using biSANLM.

BERT as BiSANLM

BERT is a recently proposed language representation model that consists of a multi-layer bidirectional Transformer encoder [37]. Unlike the traditional LM that predicts

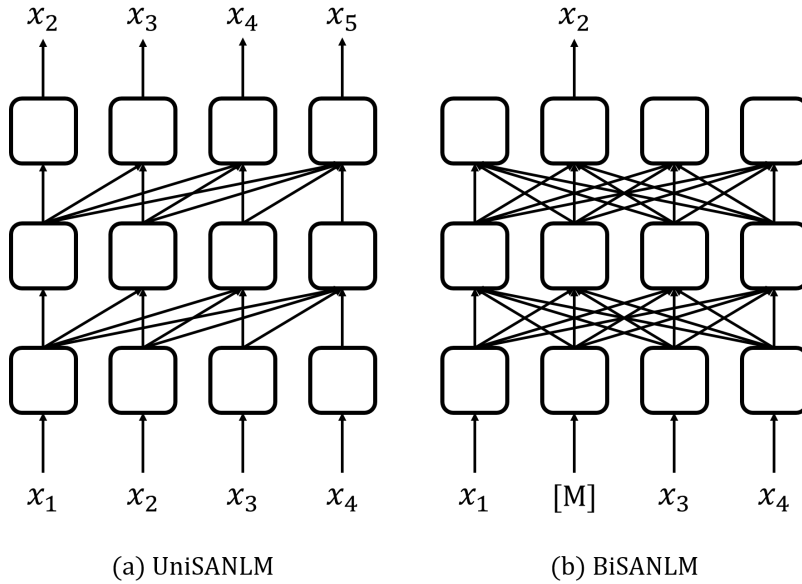


Figure 3.2: Schematic diagram of the SANLMs.

a word from its left context, BERT predicts a word from its left and right context as depicted in Figure 3.2b. In training the BERT, we mask some words and let the model predict the original words before they are masked, and this task is called “masked language modeling”. The BERT benefits a lot from the fusing of the left and right representations, and it can achieve state-of-the-art performances by transferring the pre-trained BERT to many downstream tasks [37].

The original BERT has one more training objective called “next sentence prediction”, which is designed to learn the relationship between two sentences. For this objective, the BERT has an additional construction of adding segment embedding as well as the corresponding input processing. The next sentence prediction task is proven to be beneficial to some downstream tasks such as question answering and natural language inference, which need understanding sentence relationship.

We now explain the construction of our biSANLM, which is used for scoring a given sentence in the next section. The core idea of developing our biSANLM is to bridge the gap between the training and testing environments. Note that the archi-

texture of the biSANLM is the same with that of uniSANLM for a fair comparison, including the summation of sentence embedding and position embeddings, the SAN layer, and the softmax layer (Figure 3.1a). Unlike the uniSANLM, however, the biSANLM do not use the masking operation so as to catch the left and right context during the sentence scoring.

Our training approach has many differences from that of the BERT because our purpose of training bidirectional LM is for scoring a sentence rather than for fine-tuning the model to the other task [37]. To train our biSANLM, we only consider the masked word prediction task from the BERT [37], and make several adjustments: First, our training instance has a single sentence (maximum of 128 words) instead of multiple sentences. Second, we randomly sample some words from the sentence like in the BERT, but replace them by [MASK] tokens *all the time* unlike in the BERT [37]. Lastly, the maximum number of masked tokens in a training instance is limited by the small number of 4, because our instance has only one sentence and too much loss in information is unhelpful to train the model. Note that we make the training instances have multiple masked tokens [MASK] for efficient training, while we make the inference instance have only one [MASK] for the sentence scoring.

In addition, we neglect the next sentence prediction task of the BERT for training our biSANLM because this objective is not designed to learn to evaluate the probability of a sentence. We also exclude the segment embedding from the input representation of the original BERT, and ignore the [CLS] and [SEP] tokens from input processing and our vocabulary. Even in our preliminary study, we observe that this task is not helpful to language modeling, but rather hampers the predicting the masked word. Discarding the next sentence prediction task with its corresponding input processing is another difference between our biSANLM and the original BERT.

Sentence Scoring Method Using BiSANLM

This section introduces our sentence scoring method that adopts the masked word prediction of the BERT [37]. The basic principle of our sentence scoring is to mask one word in a given sentence and then compute the probability of the original word on the masked position using the trained biSANLM. Because the whole sentence with the masked word is taken to the model as an input, both past and future representations can be fused during prediction.

Our sentence scoring method takes the following procedure: First, we create a set of instances from a given sentence by replacing each word with the predefined token [MASK] one at a time. For example, if the sentence has seven words, we create seven instances as below:

- **A given sentence:**

```
move the vat over the hot fire
```

- **A set of instances we create:**

1. **Input** = [MASK] the vat over the hot fire

Label = move

2. **Input** = move [MASK] vat over the hot fire

Label = the

...

7. **Input** = move the vat over the hot [MASK]

Label = fire

After the creation, our bidirectional LM takes each instance and computes the likelihood of the original word in the masked position as shown in Figure 3.2b. Finally, the score of the given sentence is obtained by summing all log-likelihoods of the masked words from each input instance. Although it may not be a sentence probability as of

traditional LM, the score can still be used for the N -best list re-ranking conditioned on the biSANLM. We note that the past and future words are connected through the designated token [MASK] in the input instance, and thus we can make our biSANLM have interactions between both sides without making the prediction task trivial.

3.3.3 Re-ranking the N -best List with SANLMs

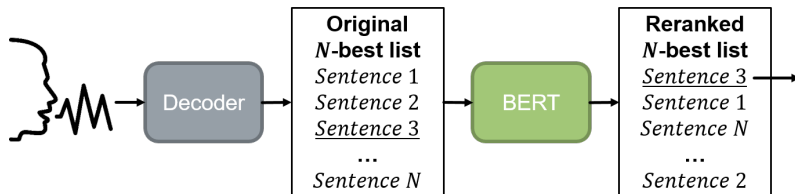


Figure 3.3: Example of procedure for re-ranking using BERT on ASR.

Figure 3.3 shows an example of procedure for re-ranking using BERT on the ASR system. We note that BERT can be replaced with any language model including the proposed biSANLM and conventional uniSANLM.

We consider the sum of all log-likelihoods of each word as the sentence score in the uniSANLM. Likewise, We consider the sum of all log-likelihoods of each masked word in each input sentence as the sentence score of the biSANLM. Following the previous works on bidirectional LMs for speech recognition [49, 50], we use our sentence score for re-ranking the N -best hypotheses. We linearly interpolate the scores obtained by the acoustic model (AM) and the language model (LM):

$$\text{score} = (1 - \lambda) \cdot \text{score}_{\text{AM}} + \lambda \cdot \text{score}_{\text{LM}}, \quad (3.3)$$

where λ is the interpolation weight, which is determined empirically on development data. For a fair comparison in terms of information, we average the scores of the forward and the backward SANLMs like $\text{score}_{\text{LM}} = (\text{score}_{\text{uniSANLM}_{fw}} + \text{score}_{\text{uniSANLM}_{bw}}) / 2$, which is an ELMo-like bidirectional LM.

3.4 Experiments

We evaluate the proposed approach on the LibriSpeech ASR task [24]. The 960-hour of training data is used to train an acoustic model, which is our base speech recognition system. We obtain the 100-best hypothesis list for each audio in development and test data using the acoustic model, and then use language models (LMs) to rescore these 100-best lists. For comparison, we use our biSANLM, the forward uniSANLM, and the backward uniSANLM. The details of the acoustic model and language model settings are explained in the following sections.

3.4.1 Acoustic Model Setups

In this study, we use the attention-based seq2seq model *Listen, Attend and Spell* (LAS) [52] as our acoustic model with some differences. First, there are additional bottleneck fully connected (FC) layers between every bidirectional long-short term memory (BLSTM) layer. Second, the number of time steps is reduced in half by just subsampling hidden states for even number time steps before the FC layer, instead of concatenating every two hidden states. Third, LAS is trained with additional CTC objective function because the left-to-right constraint of CTC helps LAS learn alignments between speech-text pairs [57]. Based on 5K case-insensitive sub-word units created via unigram byte-pair encoding [58],

The details of our acoustic model follow the default settings provided in ESPNet toolkit v.0.2.0 [59]. For the input features, we use 80-band mel-scale spectrogram derived from the speech signal. The encoder consists of 5-layer pyramidal-BLSTM with subsampling after second and third layers. The decoder is comprised of 2-layer LSTM with location-aware attention mechanism [60]. The target sequence is processed in 5K case-insensitive sub-word units created via unigram byte-pair encoding [58]. All the LSTM and FC layers have 1024 hidden units each. Our model is trained for 10 epochs using Adadelta optimizer [61] with learning rate of 1e-8. Including all these hyperpa-

rameters, for all hyperparameters, are default values in ESPNet and we also use the other default values.

Using this acoustic model, we obtain 100-best decoded sentences for each input through hybrid CTC-attention based scoring [57] method, and these 100-best lists will be used for re-ranking. Table 3.1 shows the word error rates (WERs) obtained from the acoustic model and the oracle WERs, which is the best possible errors of the 100-best lists on the LibriSpeech tasks.

Table 3.1: Oracle WERs of the 100-best lists on LibriSpeech

Method	dev		test	
	clean	other	clean	other
1-best	7.17	19.79	7.26	20.37
100-best (oracle)	2.85	12.21	2.81	12.85

3.4.2 Language Model Setups

The model parameters of our language model (LM) are as follows: $L = 3$ for the number of layers, $d = 512$ for the dimensions of the model and the embeddings, $h = 8$ for the number of head. 2048 hidden units are used in the position-wise feed-forward layers. We use trainable positional embeddings with supported sequence lengths up to 128 tokens. We use a *gelu* activation [62] rather than the standard *relu*, following [10, 37]. Weight matrix of the softmax layer is shared with the word embedding table. The word vocabulary is used in three sizes: 10k, 20k and 40k most frequent words. For a fair comparison in terms of the number of parameters, our biSANLM and uniSANLMs have the same architecture and parameters.

We train the LMs with the 1.5G normalized text-only data of the official LibriSpeech corpus. We use Adam optimizer [63] with learning rate of $1e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. We use a dropout probability of 0.1 on all layers. Batch size is set to 128

for biSANLMs and 64 for uniSANLMs, and all the LMs are trained for 1M iterations. We confirmed that all our LMs are converged before the 1M training steps.

3.4.3 Results: Re-ranking the N-best List

In this section, we compare the LMs for the N -best re-ranking on all test sets of the LibriSpeech ASR corpus, in which the test sets are classified as “clean” or “other” set based on their difficulties. We first prepare 100-best hypotheses using our acoustic model (AM), which is a base speech recognition system in our experiments. For re-ranking the 100-best list, the AM is linearly interpolated with one or two of LMs as in Equation 3.3.

Table 3.2 shows re-ranking results of the biSANLMs and the other LMs with different test sets and different vocabulary sizes $|V|$. The WER results show that the biSANLM with our approach is consistently and significantly better than the uniSANLM regardless of the test set and the vocabulary size. While WER reductions are also observed from the backward SANLMs, amounts of WER reduction are smaller than the forward SANLMs. Moreover, combining the forward and the backward SANLMs is not helpful to re-ranking the N -best list. The results demonstrate that the fusion of the left and right representations is important to predict a score of a given sentence.

The interpolation weight is set to a value that achieves the best performance in the development sets. We find that $\lambda = 0.2$ and 0.3 are the best weights for dev-clean and dev-other sets respectively. Considering that the dev-other set is more difficult for the acoustic model to recognize, it is reasonable to have larger interpolation weight in dev-other ($\lambda = 0.3$) than in dev-clean ($\lambda = 0.2$).

Towards Further Improvements

To see greater performance improvements, we conduct linear interpolation of the biSANLM and the uniSANLM for further improvements:

$$\text{score}_{\text{LM}} = (1 - \alpha) \cdot \text{score}_{\text{uniSANLM}} + \alpha \cdot \text{score}_{\text{biSANLM}},$$

Table 3.2: WERs for unidirectional and bidirectional SANLMs interpolated with the baseline model on LibriSpeech

Model	V	dev		test	
		clean	other	clean	other
AM only		7.17	19.79	7.26	20.37
+ biSANLM	10k	5.65	16.85	5.69	17.59
	20k	5.57	16.71	5.68	17.37
	40k	5.52	16.61	5.65	17.44
+ uniSANLM _{fw}	10k	6.09	17.50	6.08	18.33
	20k	6.05	17.48	6.11	18.25
	40k	6.08	17.32	6.11	18.13
+ uniSANLM _{bw}	10k	6.15	17.78	6.24	18.51
	20k	6.17	17.60	6.22	18.49
	40k	6.17	17.57	6.24	18.29
+ uniSANLM _{fw} + uniSANLM _{bw}	10k	6.11	17.71	6.15	18.41
	20k	6.12	17.52	6.16	18.32
	40k	6.16	17.42	6.18	18.22

where α is another interpolation weight and score_{LM} is used in Equation 3.3. We find $\alpha = 1$ shows the best performances on all dev sets, which means only the biSANLM is used for interpolation (log-linear interpolation of the two LMs shows the same phenomenon). Contrary to our first expectation, the biSANLM and the uniSANLM do not complement each other in our experiments.

3.4.4 Analysis: Misrecognized Position

To understand how bidirectional LMs is better than unidirectional LMs in re-ranking tasks, we analyze the position of the misrecognized words to see where the “word error” occurs. We assume that bidirectional LMs will be more effective than unidirectional LMs particularly when a recognized sentence is short or a misrecognized word is at the beginning of the sentence. This is reasonable assumption since unidirectional LMs also can utilize enough context when the sentence is long or misrecognized words are at the end of the sentence.

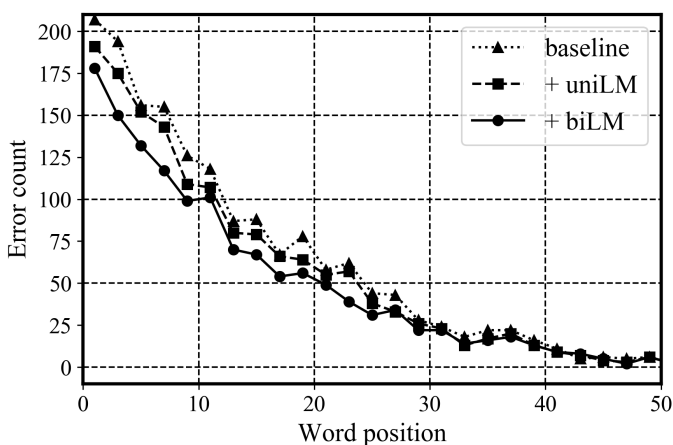


Figure 3.4: Error count by word position.

Figure 3.4 shows the total number of the misrecognized words for each model according to the position of the final hypotheses. Analysis shows that the bidirectional LM is more robust than the unidirectional LM at the earlier position (< 30) of the sentence. At the latter position (> 30) of a long sentence, however, the gap between the unidirectional LM and bidirectional LM is reduced. From this analysis, we can confirm our assumption that unidirectional LMs also perform well in general, but bidirectional LM are more powerful especially when erroneous words appear in the beginning of the recognized sentence.

Consequently, all experimental results demonstrate that our sentence scoring method

using the biSANLM is almost strictly better than the traditional method using uniSANLM for the N -best list re-ranking. As far as we know, this is the first study that the bidirectional language model significantly and consistently outperforms the unidirectional language model for speech recognition.

3.5 Summary of MLM for Sentence Scoring

In this chapter, we proposed a sentence scoring method using a biSANLM and verified its effectiveness for the N -best list re-ranking in automatic speech recognition. By adapting BERT to the sentence scoring, the left and right representations are fused in our biSANLM unlike previous bidirectional LMs for ASR. Experimental results on the LibriSpeech ASR tasks show that the proposed sentence scoring method with our biSANLM significantly and consistently outperforms the conventional uniSANLM for re-ranking the N -best list. In addition, we confirm that bidirectional LMs are more robust than unidirectional LMs especially when a recognized sentence is short or the earlier part of the sentence is misrecognized.

We can conclude that a MLM-trained model (like biSANLM) understands a given sentence better than a ALM-trained model (like uniSANLM) on unsupervised learning tasks as well as on supervised learning tasks. Indeed, [64] supports this conclusion with more experimental results in a large scale. However, MLM-trained models has a critical limitation that their computational complexities increase by a factor of $O(T)$ to compute likelihoods of word sequence of length T . This mask-and-predict repetition will be tackled in detail in the next chapter.

Chapter 4

Bidirectional Language Autoencoding for Sentence Scoring

This chapter introduces bidirectional language autoencoding (BLA), which is a new language modeling objective proposed in the research paper [25]. Even though masked language modeling (MLM) has achieved performance improvements in many NLP tasks over autoregressive language modeling (ALM) including the N -best list re-ranking, MLM is still limited by repetitive inferences for computing the pseudo-likelihood of a given sentence. To resolve this limitation, we propose a novel deep bidirectional language model called a Transformer-based text autoencoder (T-TA) that learns BLA. The proposed BLA-trained model performs much faster than the MLM-trained model on re-ranking tasks while maintaining the accuracy of MLM.

4.1 Fast and Accurate Bidirectional LMs

4.1.1 Overview

A language model is an essential component of many natural language processing (NLP) applications ranging from automatic speech recognition (ASR) [52, 24] to neural machine translation (NMT) [65, 66, 9]. Recently, the Bidirectional Encoder Rep-

representations from Transformers (BERT) [37] and its variations have led to significant improvements in learning natural language representation and have achieved state-of-the-art performances on various downstream tasks such as the General Language Understanding Evaluation (GLUE) benchmark [27] and question answering [34]. BERT continues to succeed in various unsupervised tasks, such as the N -best list reranking for ASR and NMT [23, 64], confirming that deep bidirectional language models are useful in unsupervised applications as well.

However, concerning its applications to unsupervised learning tasks, BERT is significantly inefficient at computing language representations at the inference stage [23, 64]. During training, BERT adopts the *masked language modeling* (MLM) objective, which is to predict the original word of the explicitly masked word from the input sequence. Following the MLM objective, the probability of each word should be computed by a two-step process: masking a word in the input and feeding the result to BERT. During the inference stage, this process is repeated T times to obtain the probabilities of all words within a text sequence [11, 23, 64], resulting in a computational complexity of $O(T^3)$ ¹ in terms of the number of words T . Hence, it is necessary to reduce the computational complexity when applying the model to situations where the inference time is critical, *e.g.*, mobile environments and real-time systems [67, 39]. Considering this limitation of BERT, we raise a new research question: “Can we construct a deep bidirectional language model with a minimal inference time while maintaining the accuracy of BERT?”

4.1.2 Contributions

In this chapter, in response to the question above, we propose a novel bidirectional language model named the Transformer-based text autoencoder (T-TA), which has a reduced computational complexity of $O(T^2)$ when applying the model to unsupervised applications. The proposed model is trained with a new learning objective named *bidi-*

¹A complexity of $O(T^2)$ is derived from the per-layer complexity of the Transformer [9].

rectional language autoencoding (BLA). The BLA objective, which allows the target labels to be the same as the text input, is to predict every token in the input sequence simultaneously without merely copying the input to the output. To learn the proposed objective, we devise both a **diagonal masking** operation and an **input isolation** mechanism inside the T-TA based on the Transformer encoder [9]. These components enable the proposed T-TA to compute contextualized language representations at once while maintaining the benefits of the deep bidirectional architecture of BERT.

We conduct experiments on N -best list re-ranking tasks in automatic speech recognition (ASR) and neural machine translation (NMT). First, by conducting runtime experiments in a CPU environment, we show that the proposed T-TA is 6.35 times faster than the BERT-like model in the re-ranking task. Second, with its faster inference time, T-TA achieves competitive performances relative to BERT on re-ranking tasks. Further analysis shows that

4.2 Related Works

4.2.1 Bidirectional LMs for Unsupervised Tasks

For unsupervised tasks, researchers have adopted recently developed language-representation models and investigated their effectiveness; a typical example is the N -best list re-ranking for ASR and NMT tasks. In particular, studies have integrated left-to-right and right-to-left language models [49, 50, 68] to outperform conventional unidirectional language models [5, 69] in these tasks. Furthermore, BERT-based approaches have been explored and have achieved significant performance improvements on these tasks because bidirectional language models yield the pseudo-log-likelihood of a given sentence, and this score is useful in ranking the N -best hypotheses [11, 64].

4.2.2 Consideration of Inference Speed

Another line of research involves reducing the computation time and memory consumption of BERT. [39] proposed parameter-reduction techniques, factorized embedding parameterization and cross-layer parameter sharing and reported 18 times fewer parameters and a 1.7-fold increase in the training time. Similarly, [67] presented a method to pretrain a smaller model that can be fine-tuned for downstream tasks and achieved 1.4 times fewer parameters with a 1.6-fold increase in the inference time. However, none of these studies developed methods that directly revise the BERT architecture to reduce the computational complexity during the inference stage.

4.3 Methodology

One of the main interests in this chapter is to develop a new deep bidirectional language model named T-TA and to demonstrate the superiority of T-TA for sentence scoring over two baseline language models: (1) a unidirectional language model (uniLM) that learns autoregressive language modeling (ALM) and (2) a bidirectional language model (biLM) that learns masked language modeling (MLM). As our T-TA is a variant of the BERT [37], which consists of the encoder of the Transformer [9], we also construct uniLM and biLM based on this self-attention network (SAN) as in the Chapter 3. We first review ALM and MLM briefly in Section 4.3.1, and introduce our bidirectional language autoencoding (BLA) in Section 4.3.2 and its model architecture in Section 4.3.3. We note that all language models have the same architecture for a fair comparison.

4.3.1 Baselines: ALM for UniLM and MLM for BiLM

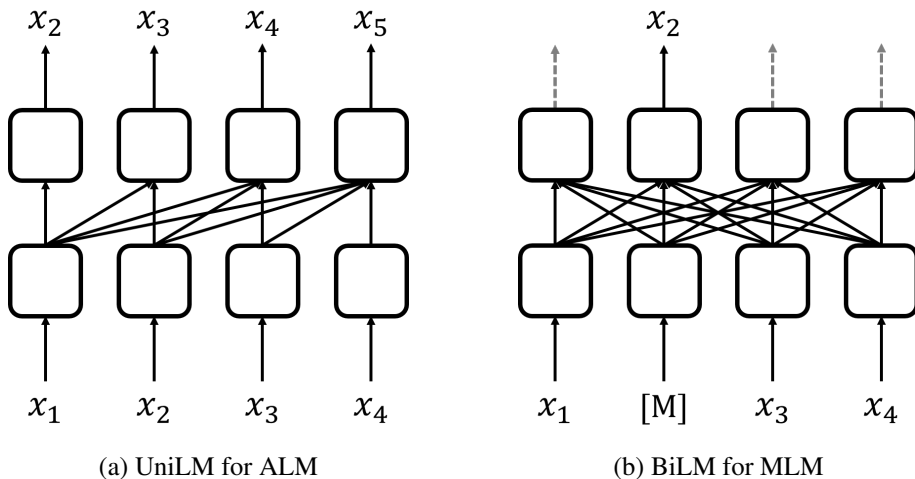


Figure 4.1: Schematic diagram of SAN-based (a) uniLM and (b) biLM.

In a conventional language modeling task, the i -th token x_i is predicted using its preceding context $\mathbf{x}_{<i} = [x_1, \dots, x_{i-1}]$; throughout this dissertation, this objective is

known as autoregressive language modeling (ALM). As shown in Figure 4.1a, we can obtain (left-to-right) contextualized language representations $\mathbf{H}^C=[H_1^C, \dots, H_n^C]$ after feeding the input sequence to the ALM-trained language model only once, where $H_i^C=h^C(\mathbf{x}_{<i})$ is the hidden representation of the i -th token. This dissertation takes this unidirectional language model (uniLM) as our speed baseline. However, contextualized language representations obtained from the uniLM are insufficient to accurately encode a given text because future contexts cannot be leveraged to understand the current tokens during the inference stage.

Recently, BERT [37] was designed to enable the full contextualization of language representations by using the MLM objective, in which some tokens from the input sequence are randomly masked; the objective is to predict the original tokens at the masked positions using only their context. As in Figure 4.1b, we can obtain a contextualized representation of the i -th token $H_i^M=h^M(M_i(\mathbf{x}))$ by masking the token in the input sequence and feeding it to the MLM-trained model, where $M_i(\mathbf{x})=[x_1, \dots, x_{i-1}, [\text{MASK}], x_{i+1}, \dots, x_n]$ signifies an external masking operation. This dissertation takes this bidirectional language model (biLM) as our performance baseline. However, this *mask-and-predict* approach should be repeated n times to obtain all the language representations $\mathbf{H}^M=[H_1^M, \dots, H_n^M]$ because learning occurs only at the masked position during the MLM training stage. Although the resulting language representations are robust and accurate, as a consequence of this repetition, the model is significantly inefficient when applied to unsupervised tasks such as N -best list re-ranking [11, 64].

4.3.2 BLA: New Language Modeling Objective

In this dissertation, we propose a new learning objective named *bidirectional language autoencoding* (BLA) for obtaining fully contextualized language representations without repetition. The BLA objective, with which the output is the same as the input, is to predict every token in a text sequence simultaneously without merely copying the input to the output. For the proposed task, a language model should reproduce the

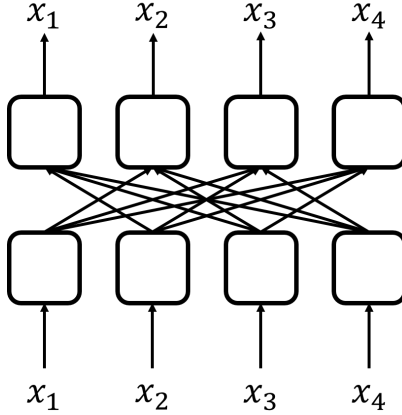


Figure 4.2: Schematic diagram of a one-layer language model for BLA.

whole input at once while avoiding over-fitting; otherwise, the model outputs only the representation copied from the input representation without learning any statistics of the language. To this end, the flow of information from the i -th input to the i -th output should be blocked inside the model shown in Figure 4.2. From the BLA objective, we can obtain fully contextualized language representations $\mathbf{H}^L = [H_1^L, \dots, H_n^L]$ all at once, where $H_i^L = h^L(\mathbf{x}_{\setminus i})$ and $\mathbf{x}_{\setminus i} = [x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n]$. The method for blocking the flow of information is described in the next section.

The most challenging part for the BLA task is that despite of blocking the accessibility of each self-token in one layer, stacking multiple layers can make the model to lose “self-unknown” property. For an example in Figure 4.3, each token can “see-itself” indirectly in the two-layer architecture. Since stacking multiple layers is critical to the performance of a language model, we should solve this challenge for developing deep and bidirectional language model that learns our BLA.

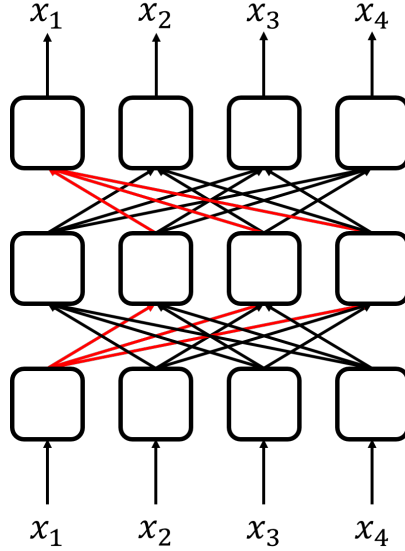


Figure 4.3: Example of losing self-unknown property with a multi-layer language model for BLA.

4.3.3 T-TA: New Deep Bidirectional Language Model

Architecture of T-TA

In this section, we introduce the novel architecture of the proposed **T-TA** shown in Figure 4.4. As indicated by its name, the T-TA architecture is based on the Transformer encoder [9]. To learn the proposed BLA objective, we develop both a **diagonal masking** operation and an **input isolation** mechanism inside the T-TA. Both developments are designed to enable the language model to predict all tokens simultaneously while maintaining the deep bidirectional property (see the descriptions in the following subsections). We refer to Chapter 2 for the Transformer encoder [9] for other details regarding the standard functions, such as the multi-head attention, scaled dot-product attention mechanisms, layer normalization, and the position-wise fully connected feed-forward network.

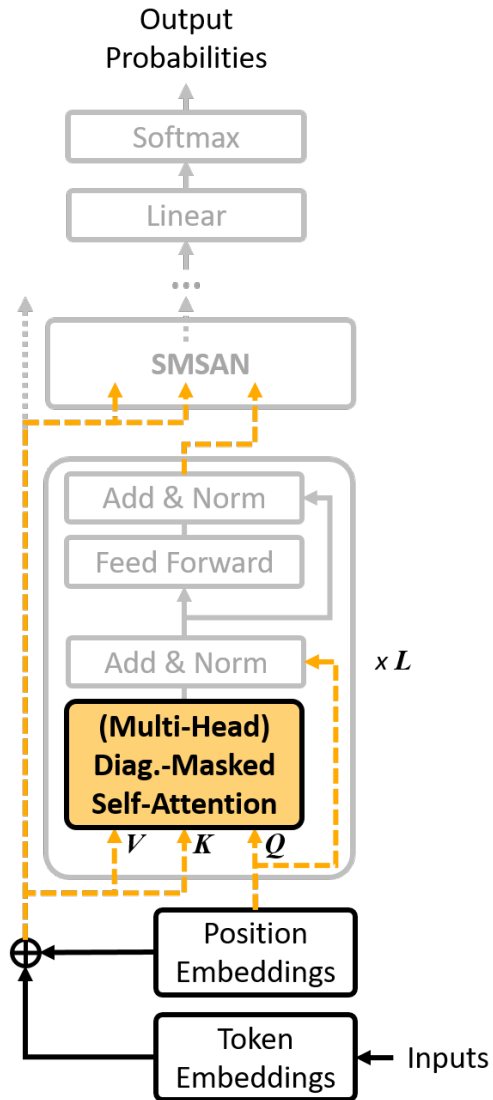


Figure 4.4: Architecture of our T-TA. The highlighted box and dashed arrows are the innovations presented in this dissertation.

Diagonal Masking

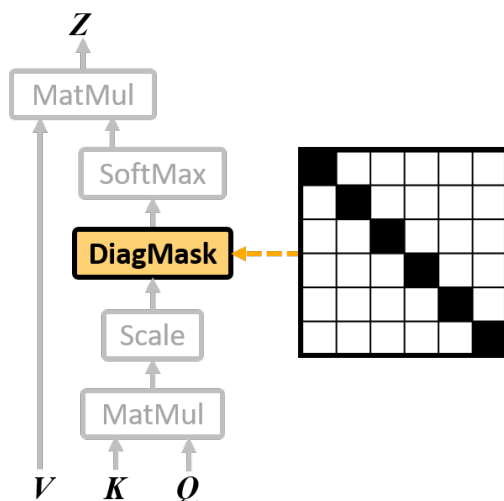


Figure 4.5: Diagonal masking of the scaled dot-product attention mechanism. The highlighted box and dashed arrow represent the innovations reported in this dissertation.

As shown in Figure 4.5, a diagonal masking operation is implemented inside the scaled dot-product attention mechanism to be “self-unknown” during the inference stage. This operation prevents information from flowing to the same position in the next layer by masking out the diagonal values in the input of the softmax function. Specifically, the output vector at each position is the weighted sum of the value V at other positions, where the attention weights come from the query Q and the key K .

The diagonal mask becomes meaningless when we use it together with a residual connection or utilize it within the multilayer architecture. To retain the self-unknown functional, we can remove the residual connection and adopt a single-layer architecture. However, it is essential to utilize a deep architecture to understand the intricate patterns of natural language. To this end, we further develop the architecture described in the next section.

Input Isolation

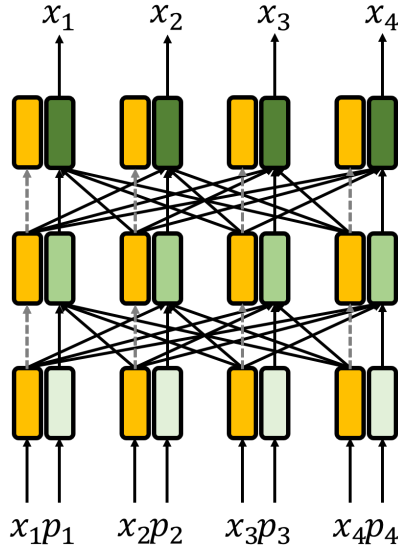


Figure 4.6: Schematic diagram of T-TA for BLA.

We now propose an input isolation mechanism to ensure that the residual connection and the multi-layer architecture are compatible with the aforementioned diagonal masking operation. In the input isolation mechanism, the key and value inputs (\mathbf{K} and \mathbf{V} , respectively) of all encoding layers are isolated from the network flow and are fixed to the sum of the token embeddings and the position embeddings. Hence, only the query inputs (\mathbf{Q}) are updated across the layers during the inference stage by referring to the fixed output of the embedding layer.

Additionally, we input the position embeddings to the \mathbf{Q} of the very first encoding layer, thereby making the self-attention mechanism effective. Otherwise, the attention weights will be the same at all positions, and thus, the first self-attention mechanism will function as a simple average of all the input representations (except the “self” position). Finally, we apply the residual connection only to the query to completely maintain unawareness. The dashed arrows in Figure 4.4 show the proposed input isolation mechanism inside the T-TA.

By using diagonal masking and input isolation in conjunction, the T-TA can have multiple encoder layers, enabling the T-TA to obtain high-quality contextual language representations after feeding a sequence into the model only once. Schematic diagram of the architecture of T-TA is shown in the Figure 4.6. Arrows denote information flow and dotted (gray-colored) arrows denote the copy without updates.

4.3.4 Verification of the T-TA Architecture

Here, we discuss how diagonal masking with input isolation preserves the “self-unknown” property in detail.

As shown in Figure 4.4, we have two input embeddings, namely, token embeddings $\mathbf{X} = [X_1, \dots, X_n]^T \in \mathbb{R}^{n \times d}$ and position embeddings $\mathbf{P} = [P_1, \dots, P_n]^T \in \mathbb{R}^{n \times d}$, where d is an embedding dimension. From the input isolation mechanism, the key and value $\mathbf{K} = \mathbf{V} = \mathbf{X} + \mathbf{P}$ have the information of the input tokens and are *fixed* in all layers, but the query \mathbf{Q}^l is *updated* across the layers during the inference stage starting from the position embeddings $\mathbf{Q}^1 = \mathbf{P}$ in the first layer.

Let us consider the l -th encoding layer’s query input \mathbf{Q}^l and its output $\mathbf{H}^l = \mathbf{Q}^{l+1}$:

$$\begin{aligned} \mathbf{H}^l &= \text{SMSAN}(\mathbf{Q}^l, \mathbf{K}, \mathbf{V}) \\ &= g(\text{Norm}(\text{Add}(\mathbf{Q}^l, f(\mathbf{Q}^l, \mathbf{K}, \mathbf{V})))), \end{aligned} \quad (4.1)$$

where $\text{SMSAN}(\cdot)$ is the self-masked self-attention network, namely, the encoding layer of the T-TA, $g(x) = \text{Norm}(\text{Add}(x, \text{FeedForward}(x)))$ signifies two upper sub-boxes of the encoding layer in Figure 4.4, and $f(\cdot)$ is the (multihead) diagonal-masked self-attention (DMSA) mechanism. As illustrated in Figure 4.5, the DMSA module computes \mathbf{Z}^l as follows:

$$\begin{aligned} \mathbf{Z}^l &= f(\mathbf{Q}^l, \mathbf{K}, \mathbf{V}) = \text{DMSA}(\mathbf{Q}^l, \mathbf{K}, \mathbf{V}) \\ &= \text{SoftMax}(\text{DiagMask}(\mathbf{Q}^l \mathbf{K}^T / \sqrt{d})) \mathbf{V}. \end{aligned} \quad (4.2)$$

In the DMSA module, the i -th element of $\mathbf{Z}^l = [Z_1^l, \dots, Z_n^l]^T$ is always computed by a weighted average of the fixed \mathbf{V} while discarding the information of the i -th token

X_i in V_i . Specifically, Z_i^l is the weighted average of \mathbf{V} with the attention weight vector \mathbf{s}_i^l , i.e., $Z_i^l = \mathbf{s}_i^l \mathbf{V}$, where $\mathbf{s}_i^l = [s_1^l, \dots, s_{i-1}^l, 0, s_{i+1}^l, \dots, s_n^l] \in \mathbb{R}^{1 \times n}$. Here, we note that the DMSA mechanism is related only to the “self-unknown” property since no token representations are referred to each other in subsequent transformations from \mathbf{Z}^l to \mathbf{H}^l . Therefore, we can guarantee that the i -th element of the query representation in any layer, Q_i^l , never encounters the corresponding token representation starting from $Q_i^1 = P_i$. Consequently, the T-TA preserves the “self-unknown” property during the inference stage while maintaining the residual connection and multilayer architecture.

4.3.5 Comparison T-TA with BERT

There are several differences between the strong baseline BERT [37] and the proposed T-TA, while both models learn deep bidirectional language representations.

- While BERT uses an external masking operation in the input, the T-TA has an internal masking operation in the model, as we intend. Additionally, while BERT is based on a denoising autoencoder, the T-TA is based on an autoencoder. With this novel approach, the T-TA does not need *mask-and-predict* repetition during the computing of contextual language representations. Consequently, we reduce the computational complexity from $O(T^3)$ with the BERT to $O(T^2)$ with the T-TA in applications to unsupervised learning tasks.
- As in the T-TA, feeding an intact input (without masking) into BERT is also possible. However, we argue that this process will significantly diminish the model performance in unsupervised applications since the MLM objective does not consider intact tokens much. In the next section, we include experiments that reveal the model performance with intact inputs (described in Tables 4.5 and 4.6). For further reference, we also suggest a previous study that reported the same opinion [64].

4.4 Experiments

To evaluate the proposed method, we conduct experiments on N -best list re-ranking tasks in ASR and NMT. The following sections will demonstrate that the proposed model is much faster than BERT during the inference stage (Section 4.4.2) while showing competitive accuracy than that of BERT on re-ranking tasks (Section 4.4.3).

4.4.1 Language Model Setups

The main purpose of this dissertation is to compare the proposed T-TA with a biLM trained with the MLM objective. For a fair comparison, each model has the same number of parameters based on the Transformer as follows: $|L| = 3$ self-attention layers with $d = 512$ input and output dimensions, $h = 8$ attention heads, and $d_f = 2048$ hidden units for the position-wise feed-forward layers. We use a Gaussian error linear unit (*gelu*) activation function [62] rather than the standard rectified linear unit (*relu*) following OpenAI GPT [10] and BERT [37]. In our experiments, we set the position embeddings to be trainable following BERT [37] rather than a fixed sinusoid [9] with supported sequence lengths up to 128 tokens. We use WordPiece embeddings [29] with a vocabulary of approximately $|V| \simeq 30,000$ tokens. The weights of the embedding layer and the last softmax layer of the Transformer are shared. For the speed baseline, we also implement a uniLM that has the same number of parameters as the T-TA and biLM.

For training, we create a training instance consisting of a single sentence with [BOS] and [EOS] tokens at the beginning and end of each sentence, respectively. We use 64 sentences as the training batch and train the language models over $1M$ steps for ASR and $2M$ steps for NMT. We train the language models with Adam [63] with an initial learning rate of $1e - 4$ and coefficients of $\beta_1 = 0.9$ of $\beta_2 = 0.999$; the learning rate is set to warm up over the first 50k steps, and the learning rate exhibits linear decay. We use a dropout probability of 0.1 on all layers. Our implementation is

based on Google’s official code for BERT².

To train the language models that we implement, we use an English Wikipedia dump (approximately 13 GB in size) containing approximately 120M sentences. The trained models are used for the re-ranking in NMT. For the ASR re-ranking task, we use additional in-domain training data, namely, 4.0 GB of normalized text data from the official LibriSpeech corpus containing approximately 40M sentences.

Table 4.1: Accuracy of each language model for each training task. Plain texts of the test-clean set of LibriSpeech ASR Corpus are used in this experiment.

	Method	Accuracy
uniLM	next token prediction	0.259
biLM	masked token prediction	0.455
biLM _{\M}	self token prediction	0.732
T-TA	language auto-encoding	0.498

Training accuracy of our implementations of uniLM, biLM and T-TA is shown in Table 4.1. First, The next token prediction accuracy is approximately a half of masked token prediction (and language auto-encoding), and this is reasonable result because uniLM can use only unidirectional context. Since biLM has 15% independent masks in a training instance, the accuracy for masked token prediction is lower than language auto-encoding of T-TA. If we take mask-predict approach on biLM to each token, the accuracy for masked token prediction increases to 0.514. The biLM_{\M} obtains high accuracy while biLM is only learn to MLM. It reveals that unmasked tokens in biLM tend to forward themselves strongly.

²<https://github.com/google-research/bert>

4.4.2 Analysis: Runtime Comparison

We first measure the runtime of each language model to compute the contextual language representation $\mathbf{H}^L \in \mathbb{R}^{n \times d}$ of a given text sequence. In the case of the re-ranking task, further computation is required: we compute $\text{Softmax}(\mathbf{H}^L \mathbf{E}^\top)$ to obtain the likelihood of each token, where $\mathbf{E} \in \mathbb{R}^{|V| \times d}$ is the weight parameter of the softmax layer. To measure the runtime, we use an Intel(R) Core(TM) i7-6850K CPU (3.60 GHz) and the TensorFlow 1.12.0 library with Python 3.6.8 on Ubuntu 16.04.06 LTS. In each experiment, we measure the runtime 50 times and average the results.

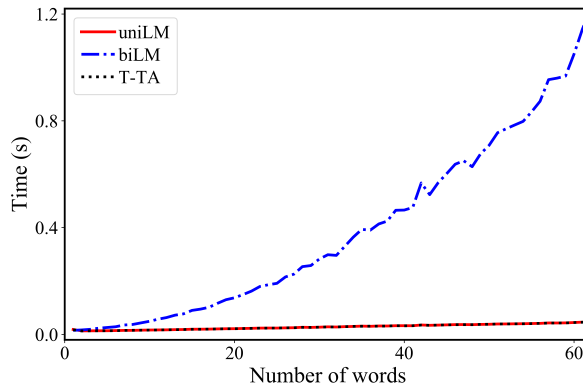


Figure 4.7: Average runtimes of each model according to the number of words on the re-ranking task.

Figure 4.7 shows that the T-TA exhibits faster runtimes than the biLM, and the gap between the T-TA and biLM increases as the sentence becomes longer. To facilitate a numerical comparison, we set the standard number of words to 20, which is approximately the average number of words in a contemporary English sentence [70]. In this setup, the T-TA takes approximately 21.5 ms, while the biLM takes approximately 137 ms; hence, the T-TA is 6.37 times faster than the biLM.

We also measure the runtimes of the uniLM we implement. Figure 4.8 shows the average runtimes of the uniLM and the T-TA for the number of words in a sentence. Since we use subword tokens, the number of words T_w and the number of tokens T

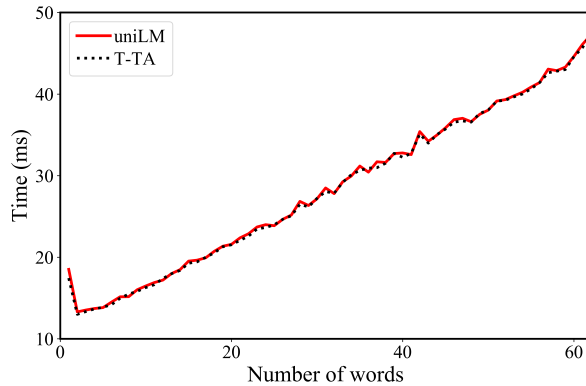


Figure 4.8: Runtimes according to the number of words for the uniLM and T-TA.

can be different ($T_w \leq T$).

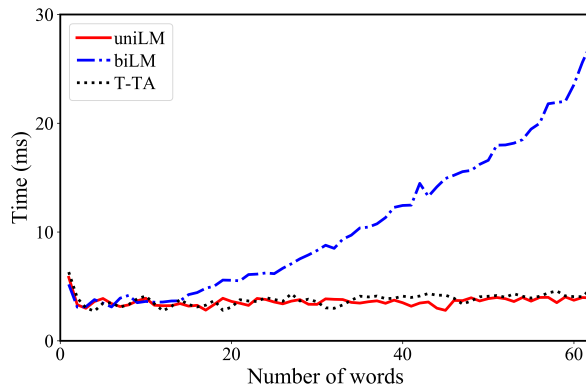


Figure 4.9: Runtimes according to the number of words in the GPU-augmented environment.

Additionally, we similarly measure the runtimes in a GPU-augmented environment (using GeForce GTX 1080 Ti). Figure 4.9 shows the average runtimes of the biLM and the T-TA for the number of words in a sentence. In our 20-word standard, the T-TA takes approximately 3.09 ms, whereas biLM takes approximately 5.56 ms, showing that the T-TA is 1.80 times faster than the biLM. Compared to the CPU-only environment, the speed difference is significantly reduced due to the support offered

by the GPU. Considering Figure 4.7, however, the CPU-only environment and GPU-augmented environment show a similar tendency: the longer the sentence is, the more significant the difference in the runtime between the T-TA and the biLM. With such a fast inference time, we next demonstrate that the T-TA is as accurate as BERT.

4.4.3 Settings: Re-ranking the N-best List

Re-ranking Method

To evaluate the language models, we conduct experiments on the unsupervised task of re-ranking the N -best list. In these experiments, we apply each language model to re-score the 50 best candidate sentences, which are obtained in advance using each sequence-to-sequence model on ASR and NMT.

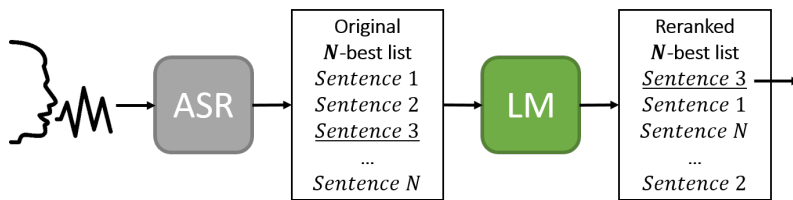


Figure 4.10: Example of procedure for re-ranking using BERT on ASR.

Figure 4.10 shows an example of procedure for re-ranking using BERT on the ASR system. We re-score the sentences by linearly interpolating two scores from a sequence-to-sequence model and each language model as follows:

$$\text{score} = (1 - \lambda) \cdot \text{score}_{s2s} + \lambda \cdot \text{score}_{lm},$$

where score_{s2s} is the score from the sequence-to-sequence model, score_{lm} is the score from the language model calculated by the sum (or mean) of the log-likelihood of each token, and the interpolation weight λ is set to a value that leads to the best performance in the development set.

One of the strong baseline language models, the pretrained BERT-base-uncased model [37], is used for re-ranking tasks. We also include the re-ranking results from

the traditional count-based 5-gram language models trained on each dataset using the KenLM library [71].

We note that the T-TA and biLM (including BERT) assign the pseudo-log-likelihood to the score of a given sentence, whereas the uniLM assigns the log-likelihood. Because the re-ranking task is based on the relative scores of the n -best hypotheses, the fact that the bidirectional models yields the pseudo-log-likelihood of a given sentence does not impact this task [11, 64].

Implementations of *Seq2Seq*_{ASR}

We use the attention-based seq2seq model *Listen, Attend and Spell* (LAS) [52] as our acoustic model with some differences. First, there are additional bottleneck fully connected (FC) layers between every bidirectional long-short term memory (BLSTM) layer. Second, the number of time steps is reduced in half by just sub-sampling hidden states for even number time steps before the FC layer, instead of concatenating every two hidden states. Third, the model is trained with an additional connectionist temporal classification (CTC) objective function because the left-to-right constraint of CTC helps learn alignments between speech-text pairs [57].

The details of our acoustic model follow the default settings provided in the efficient spatial pyramid network (ESPNet) toolkit v.0.2.0 [59]. For the input features, we use 80-band mel-scale spectrogram derived from the speech signal. The encoder consists of 5-layer pyramidal-BLSTM with sub-sampling after second and third layers. The decoder is a 2-layer bidirectional LSTM network with a location-aware attention mechanism [60]. All the layers have 1024 hidden units. The target sequence is processed in 5K case-insensitive sub-word units created via unigram byte-pair encoding [58]. Our model is trained for 20 epochs on 960h of LibriSpeech training data using Adadelta optimizer [61] with learning rate of $1e-8$. Using this acoustic model, we obtain 50-best decoded sentences for each input through hybrid CTC-attention based scoring [57] method, and these 50-best lists will be used for re-scoring. Table

4.2 shows the word error rates (WERs) obtained from the acoustic model and the oracle WERs, which is the best possible errors of the 50-best lists on the LibriSpeech ASR tasks.

Table 4.2: Oracle WERs of the 50 best lists on LibriSpeech from *Seq2Seq_{ASR-}*.

Method	dev		test	
	clean	other	clean	other
<i>Seq2Seq_{ASR-}</i>	7.17	19.79	7.26	20.37
oracle	3.18	12.98	3.19	13.61

Implementations of *Seq2Seq_{ASR+}*

To see the dependency on the power of ASR systems, we implement another ASR system, *Seq2Seq_{ASR+}*. Since most configurations are similar to the *Seq2Seq_{ASR-}*, this section details only differences. We replace a bottleneck FC layer with a VGG module before the encoder, and it reduces the number of encoding time steps by one-quarter through two max-pooling layers. For *Seq2Seq_{ASR+}*, we additionally use a pretrained recurrent neural network language model (RNNLM) to combine the log-probability p^{lm} of the RNNLM during decoding as follows:

$$\begin{aligned} & \log p(y_n | y_{1:n-1}) \\ &= \log p^{\text{am}}(y_n | y_{1:n-1}) + \beta \log p^{\text{lm}}(y_n | y_{1:n-1}), \end{aligned}$$

where β is set to 0.7. Table 4.3 shows the oracle word error rates (WERs) of the 50 best lists measured assuming that the best sentence is always picked from the candidates.

Implementations of *Seq2Seq_{NMT}*

For NMT, we implement the standard Transformer model [9] using the Tensor2Tensor library [72]. Both the encoder and the decoder of the Transformer consist of 6 lay-

Table 4.3: Oracle WERs of the 50 best lists on LibriSpeech from *Seq2Seq_{ASR+}*

Method	dev		test	
	clean	other	clean	other
<i>Seq2Seq_{ASR+}</i>	4.11	12.31	4.31	13.14
oracle	1.80	7.90	1.96	8.39

ers with 512 hidden units, and the number of self-attention heads is 8. The maximum number of input tokens is set to 256, and we use a shared vocabulary of size 32k. For effective training, we let the token embedding layer and the last softmax layer share their weights. The other hyperparameters of our translation system follow the standard `transformer_base_single_gpu` setting in Google’s official Tensor2Tensor repository³.

We train the baseline model on the standard WMT13 dataset. Fr→En and De→En datasets with 250k steps using the Adam optimizer [63]. We use linear-warmup-square-root-decay learning rate scheduling with the default learning rate (2.5e-4) and number of warmup steps (16k). Using this baseline translation model, we obtain the 50 best decoded sentences for each source through the beam search. The oracle BLEU scores for the NMT system are shown in Table 4.4.

Table 4.4: Oracle BLEU scores of the 50 best lists on WMT13

Method	WMT13	
	De→En	Fr→En
<i>Seq2Seq_{NMT}</i>	27.83	29.63
oracle	38.18	39.58

³<https://github.com/tensorflow/tensor2tensor>

4.4.4 Results: Re-ranking the N-best List

Re-ranking on ASR

For re-ranking in ASR, we use prepared N -best lists obtained from dev and test sets using *Seq2Seq_{ASR-}* and *Seq2Seq_{ASR+}*, which we train on the LibriSpeech ASR corpus. Table 4.5 shows the word error rates (WERs) for each method after re-ranking. The ‘other’ sets are recorded in noisier environments than the ‘clean’ sets. Bold font denotes the best performance on each subtask, and LM with subscript $_w$ signifies a word-level language model. The interpolation weights λ are set to be 0.2 or 0.3 in N -best lists for *Seq2Seq_{ASR-}*. The interpolation weights λ are set to be 0.3 or 0.4 in N -best lists for *Seq2Seq_{ASR+}*.

First, we confirm that the bidirectional models trained with the BLA (T-TA) and MLM (biLM) objectives consistently outperform the uniLM trained with the ALM objective. The performance gains from re-ranking are much lower in the better base system *Seq2Seq_{ASR+}*, and it is evidently challenging to re-rank the N -best list using a language model if the speech recognition model performs well enough. Interestingly, the T-TA is competitive with (or even better than) the biLM; this may result from the gap between the training and testing of the biLM: the biLM predicts multiple masks at a time when training but predicts only one mask at a time when testing. Moreover, the 3-layer T-TA is better than the 12-layer BERT-base, showing that in-domain data are critical to language model applications.

Finally, we note that feeding an intact input to BERT (the corresponding model is denoted as “w/ BERT_M” in Table 4.5) causes the model to underperform relative to the other models, demonstrating that the *mask-and-predict* approach is necessary for effective re-ranking.

Table 4.5: WERs after re-ranking with each language model on LibriSpeech

Method	dev		test	
	clean	other	clean	other
<i>Seq2Seq_{ASR-}</i>	7.17	19.79	7.25	20.37
w/ n-gram	5.62	16.85	5.75	17.72
w/ BERT	5.24	16.56	5.38	17.46
w/ BERT _{\M}	7.08	19.61	7.14	20.18
w/ uniLM	5.07	16.20	5.14	17.00
w/ biLM	4.94	16.09	5.14	16.81
w/ T-TA	4.98	16.09	5.11	16.91
<i>Seq2Seq_{ASR+}</i>	4.11	12.31	4.31	13.14
w/ n-gram	3.94	11.93	4.15	12.89
w/ BERT	3.72	11.59	3.97	12.46
w/ BERT _{\M}	4.09	12.26	4.28	13.15
w/ uniLM	3.82	11.73	4.05	12.63
w/ biLM	3.73	11.53	3.97	12.41
w/ T-TA	3.67	11.56	3.97	12.38

Re-ranking on NMT

To compare the re-ranking performances in another domain, NMT, we again prepare N -best lists using *Seq2Seq*_{NMT}⁴ from the WMT13 German-to-English (De→En) and French-to-English (Fr→En) test sets. Table 4.6 shows the bilingual evaluation under-study (BLEU) scores for each method after re-ranking. Bold font denotes the best performance on each sub-task, and the underlined values signify the best performances in our implementations. Each interpolation weight becomes a value that shows the best performance on each test set with each method in NMT. The interpolation weights λ are 0.4 or 0.5 in the N -best lists for NMT.

Table 4.6: BLEU scores after re-ranking with each language model on WMT13

Method	De→En	Fr→En
<i>Seq2Seq</i> _{NMT}	27.83	29.63
w/ n-gram	28.41	30.04
w/ BERT	29.31	30.52
w/ uniLM	28.80	30.21
w/ biLM	28.76	<u>30.32</u>
w/ T-TA	<u>28.83</u>	30.20

We confirm again that the bidirectional models trained with the BLA and MLM objectives perform better than the uniLM trained with the ALM objective. Additionally, the Fr→En translation has less effect on the re-ranking than the De→En translation because the base NMT system for Fr→En is better than that for De→En. The 12-layer BERT model appears much better than the other models at re-ranking on NMT; hence, the N -best hypotheses of the NMT model seem to be more indistinguishable than those of the ASR model from a language modeling perspective.

All the re-ranking results on the ASR and NMT tasks demonstrate that the pro-

⁴The *Seq2Seq* models for De→En and Fr→En are trained independently using the t2t library [72].

posed T-TA performs both efficiently (similar to the uniLM) and effectively (similar to the biLM).

4.4.5 Analysis: Re-ranking and Language models

Interpolation Weight

In our experiment settings on Librispeech ASR corpus, the interpolation weight λ for each language model is set to a value that achieves the best performance in the development sets. Therefore, we can interpret interpolation weight as how language models take part in the re-ranking task. Considering that the interpolation weights for the better ASR system *Seq2Seq_{ASR+}* ($\lambda = 0.3$ for the dev-clean set, $\lambda = 0.4$ for the dev-other set) are higher than those for *Seq2Seq_{ASR-}* ($\lambda = 0.2$ for the dev-clean set, $\lambda = 0.3$ for the dev-other set), we could conclude that better N -best lists need more intervention of language models. Considering that the interpolation weights for the dev-other set ($\lambda = 0.4$) on ASR system *Seq2Seq_{ASR+}* are higher than those for the dev-clean set ($\lambda = 0.3$), we could conclude that language models are more important in a noisy environment that is hard to recognize speech clearly, and the same observation can be found on different ASR system *Seq2Seq_{ASR-}*.

Correlation with Language Modelings

In general, perplexity (PPL) is a measure of how well the language model is trained. To investigate the alignment of the PPL and reranking, we compute the PPL of reference sentences from the LibriSpeech dev-clean and test-clean sets using each language model. We can obtain the pseudoperplexity (pPPL) from the biLM and T-TA since they do not follow the product rule, unlike the uniLM. Note that we compute the subword-level (p)PPL (not word-level); these values are valid only in our vocabulary.

We find that the WERs are better aligned with the median of $pPPL_m$ than with the average $pPPL_a$. Interestingly, the $pPPL_a$ of the T-TA is similar to the PPL_a of the uniLM, but the $pPPL_m$ of the T-TA is similar to that of the biLM. We additionally

	Method [WER]	(p)PPL _a	(p)PPL _m
dev clean	uniLM [3.82]	341.5	70.80
	biLM [3.73]	(76.49)	(11.93)
	T-TA [3.67]	(293.4)	(11.69)
test clean	uniLM [4.05]	495.5	73.18
	biLM [3.97]	(75.43)	(12.72)
	T-TA [3.97]	(590.0)	(12.43)

Table 4.7: (pseudo)Perplexities and corresponding WERs of the language models on LibriSpeech.

discover that if the length of a sentence is short, the T-TA shows a very high PPL, even higher than that of the uniLM.

Larger Language Models

In language modelings, having more layers, bigger batch size, and larger training samples is critical for the performance of language models. As there are many pre-trained models available, we selectively report large language models’ performances on ASR (*Seq2Seq_{ASR}*) for better understanding.

In Table 4.8, we note that GPT-2 is trained on ALM, BERT and RoBERTa are trained on MLM, where RoBERTa is an advanced version of BERT. Also, GPT-2 (117M), BERT (base), RoBERTa (base) consist of 12 SAN layers, and GPT-2 (345M), BERT (large), RoBERTa (large) consist of 24 SAN layers.

In Table 4.8, bidirectional language modeling objectives such as MLM and BLA is better than unidirectional language modeling like ALM, since WERs of GPT-2 always fall behind BERT and RoBERTa. In addition, pre-training on larger data is also important to the language model’s performance (BERT (base, cased) vs RoBERTa (base, cased)), and pre-training on bigger models is not negligible (BERT (base, cased)

Table 4.8: WERs after re-ranking with each large-size language model on LibriSpeech

Method	dev		test	
	clean	other	clean	other
baseline	7.17	19.79	7.25	20.37
uniLM (3-layer, + Libri)	5.07	16.20	5.14	17.00
biLM (3-layer, + Libri)	4.94	16.09	5.14	16.81
T-TA (3-layer, + Libri)	4.98	16.09	5.11	16.91
GPT-2 (117M, cased) [64]	5.39	16.81	5.64	17.60
BERT (base, cased) [64]	5.17	16.44	5.41	17.41
RoBERTa (base, cased) [64]	5.03	16.16	5.25	17.18
GPT-2 (345M, cased) [64]	5.15	16.48	5.30	17.26
BERT (large, cased) [64]	4.96	16.26	5.25	16.97
RoBERTa (large, cased) [64]	4.75	15.81	5.05	16.79
BERT (base, uncased) [64]	5.02	16.07	5.14	16.97
BERT (base, Only Libri.) [64]	4.63	15.56	4.79	16.50
BERT (base, + Libri.) [64]	4.37	15.17	4.58	15.96

vs BERT (large, cased)). Comparing BERT (base, uncased) and BERT (base, Only Libri.), we can conclude in-domain data is important for appropriate language evaluation or sentence scoring. Comparing BERT (base, Only Libri.) and BERT (base, + Libri.), pre-training LMs on a generic and large text corpus is also important.

We note that the performances of BLA-trained models will be the same as MLM-trained models such as BERT and RoBERT. Since BERT and RoBERTa that learns MLM require $O(T)$ times repetitions for obtaining such results in Table 4.8, we can conclude that BLA-trained models are the most promising language models for sentence scoring.

4.5 Summary of BLA for Sentence Scoring

In this chapter, we pointed out that BERT-like bidirectional language models have a trade-off between accuracy and speed for sentence scoring. To eliminate the computational overload of applying MLM to unsupervised applications, we proposed a new language modeling objective named BLA. Also, we introduced a novel language bidirectional language model named the T-TA that learns BLA. Experimental results demonstrated that the proposed learning objective and model architecture effectively eliminate the speed-accuracy trade-off of MLM-trained model. Namely, T-TA maintained the accuracy of MLM without the loss of speed when applying it to the N -best list re-ranking task.

We can conclude that a BLA-trained model (T-TA) understands a given sentence similar to a MLM-trained model (biLM) and better than a ALM-trained model (uniLM) in terms of sentence scoring or sentence likelihood. To extend this claim to general unsupervised learning tasks, we will compare contextualized language representations on the various unsupervised learning tasks as well as supervised learning tasks in the next chapter.

Chapter 5

Bidirectional Language Autoencoding for Feature Extraction

In the previous chapter, we have confirmed that our bidirectional language autoencoding (BLA) is more efficient than masked language modeling (MLM) and more effective than autoregressive language modeling (ALM) for evaluating the naturalness of a given text on sentence scoring tasks. This chapter further investigates BLA for feature extractor on various NLP applications ranging from sentence-level to token-level tasks. Furthermore, contextualized language representations from BLA-trained models are tested both on unsupervised and supervised learning tasks to see potentials of the BLA task.

5.1 Extracting Contextualized Language Representations

5.1.1 Overview

In natural language processing (NLP), extracting useful features from raw text data is one of the most underlying techniques and one of the most challenging tasks, at the same time. At the earlier stage, word embeddings has received a great attention in that word features can be trained with unsupervised learning by using co-occurrence

information of words appearing in a large text corpus [1, 2, 44]. Recently, extracting contextualized word representations from language modeling [55] or machine translation [73] has been shown to be more effective than using non-contextual (or static) word embeddings.

More recently, Bidirectional Encoder Representations from Transformers (BERT) [37] has achieved significant improvements in learning natural language representation by showing state-of-the-art performances on various supervised learning tasks such as text classification [27] and question answering [34]. However, we argue that contextualized language representations extracted from BERT is not ready to use without fine-tuning the whole network. Or, BERT should be used repeatedly to compute contextualized language representations following its learning objective known as masked language modeling (MLM). Otherwise, its performance will be degraded drastically due to the training-inference discrepancy.

5.1.2 Contributions

Throughout this dissertation, we argue that bidirectional language autoencoding (BLA) we propose is the most appropriate language modeling objective for learning deep bidirectional language representations and extracting generic linguistic features. As shown in the previous chapters, conventional autoregressive language modeling (ALM) lacks in contextualization due to the use of unidirectional contexts during training. Furthermore, MLM has the speed-accuracy trade-off when computing contextualized language representations for unsupervised learning tasks.

In this chapter, we provide various experiments about how good contextualized language representations are in different pre-trained models. First, this chapter empirically examines the performance of contextualized language representations from different language models for many unsupervised learning tasks. More precisely, target tasks are ranging from sentence-level to token-level tasks. For the sentence-level, the BLA-trained model outperforms the MLM-trained model by up to 12 points in

Pearson’s r on unsupervised semantic textual similarity tasks. Furthermore, our BLA-trained model (3-layer) outperforms the pre-trained BERT-base model (12-layer) with a big margin in this sentence-level similarity tasks. For the token-level, the BLA-trained model also outperforms the MLM-trained model on unsupervised words in contexts task, and shows competitive performance with the BERT-base model. In addition to unsupervised learning tasks, we further explore BLA-trained models on supervised learning tasks. Specifically, we conduct experiments on text classification tasks in General Language Understanding Evaluation (GLUE) benchmark [27]. We demonstrate that BLA-trained models could be a good candidate for the feature-based approach.

5.2 Related Works

5.2.1 Contextualization in Language Representations

Many studies have been performed on neural network-based language models for word-level representations. Distributed word representations were proposed and attracted considerable interest, as they were considered to be fundamental building blocks for NLP tasks [74, 3, 1]. Subsequently, researchers explored contextualized representations of text where each word has a different representation depending on the context [55]. Most recently, a Transformer-based deep bidirectional model was proposed and applied to various supervised-learning tasks with remarkable success [10, 37].

5.2.2 Word-level VS Sentence-level Representations

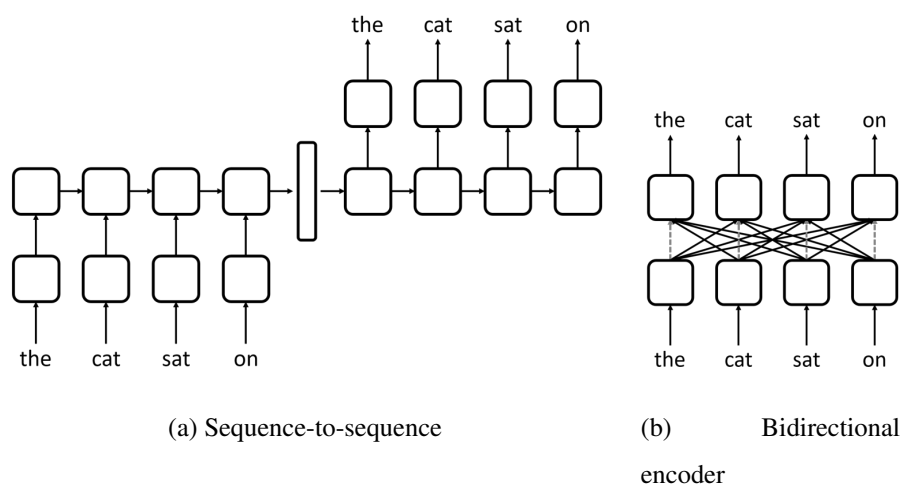


Figure 5.1: Language autoencoding with (a) sequence-to-sequence model and (b) bidirectional encoder.

When referring to an autoencoder for language modeling, sequence-to-sequence learning approaches have been commonly used (Figure 5.1a). These approaches encode a given sentence into a compressed vector representation, followed by a decoder

that reconstructs the original sentence from the *sentence-level* representation based on the autoregressive language modeling objective [65, 7, 75]. Sentence-level language representation learning is useful for sentence-level tasks such as natural language inference, sentiment analysis, and query categorization. However, their compressed representations may be too simplified to solve a bunch of NLP tasks such as question answering, machine translation, and named entity recognition. To the best of our knowledge, however, none of these approaches consider an autoencoder that encodes *word-level* representations (such as BERT) without an autoregressive decoding process (Figure 5.1b).

5.3 Experiments on Unsupervised Learning Tasks

To evaluate contextualized language representations of different language modeling objectives, we conduct experiments on unsupervised learning tasks. First, we apply language representations of each language model to unsupervised semantic textual similarity (STS) tasks. We then examine language representations in the words in context (WiC) task.

5.3.1 Language Model Setups

For these unsupervised learning tasks, we use the same language models as in the previous chapter. Summary of configurations for each language model is in the Table 5.1.

Table 5.1: Pearson’s $r \times 100$ results on the STS-B dataset

Configurations	BERT	T-TA (biLM, uniLM)
Number of layers	12	3
Number of heads	12	8
Hidden dimension	768	512
Intermediate dimension	3072	2048
Maximum sequence length	512	128
Train batch size	256	64
Train iteration	1M	2M
Total words for training	130G	16G

5.3.2 Settings: Unsupervised STS

We apply language models to an STS task, that is, measuring the similarity between the meaning of sentence pairs. We use the STS Benchmark (STS-B) [76] and Sentences

Involving Compositional Knowledge (SICK) [77] datasets, both of which have a set of sentence pairs with corresponding similarity scores. The evaluation metric of STS is Pearson’s r between the predicted similarity scores and the reference scores of the given sentence pairs. Example instance in the STS-B is:

Sentence1: A machine is sharpening a pencil.

Sentence2: The machine shaved the end of the pencil.

Similarity Score: 3.8/5.0

In this section, we address the *unsupervised* STS task to examine the inherent ability of each language model to obtain contextual language representations, and we mainly compare the language models that are trained on the English Wikipedia dump. To compute the similarity score of a given sentence pair, we use the cosine similarity of two sentence representations, where each representation is obtained by averaging each language model’s contextual representations. Specifically, the contextual representations of a given sentence are the outputs of the final encoding layer of each model, denoted as *context* in Tables 5.2 and 5.3. For comparison, we use non-contextual representations, which are obtained from the outputs of the embedding layer, denoted as *embed* in Tables 5.2 and 5.3. Figure 5.2 shows an example of procedure for unsupervised STS.

As a strong baseline for unsupervised STS tasks, we also include the 12-layer BERT model [37], and we employ BERT in the *mask-and-predict* approach for computing the contextual representations of each sentence. Note that we use the most straightforward approach for the unsupervised STS task to focus on comparing token-level language representations.

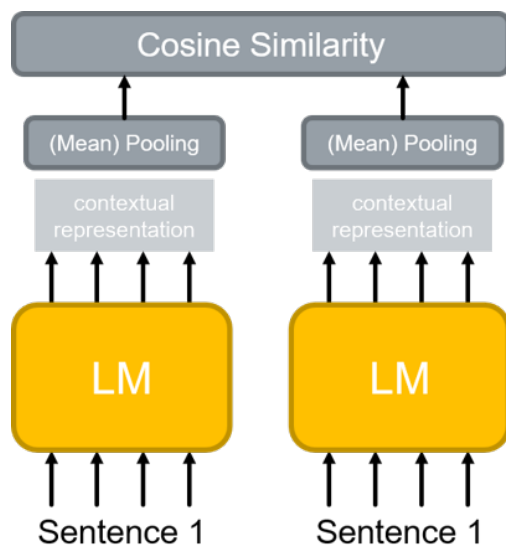


Figure 5.2: Example of procedure for unsupervised STS.

5.3.3 Results: Unsupervised STS

Results on STS-B

The STS-B dataset has (5749, 1500, 1379) sentence pairs with (train, dev, test) splits and corresponding scores ranging from 0 to 5. We test the language models on the STS-B-dev and STS-B-test sets using the simplest approach on the unsupervised STS task. As additional baselines, we include the results of GloVe [2] and Word2Vec [4] from the official sites of STS Benchmark¹.

Table 5.2 shows our T-TA trained with the BLA objective best captures the semantics of a sentence over the Transformer-based language models. “-” denotes an infeasible value, and bold font denotes the top 2-performing models on each sub-task. Remarkably, our 3-layer T-TA trained on a relatively small dataset outperforms the 12-layer BERT trained on a larger dataset (Wikipedia + BookCorpus). Furthermore, the embedding representations are trained better by the ALM objective than by the other language modeling objectives; we suppose that the uniLM depends strongly on

¹<http://ixa2.si.ehu.es/stswiki/index.php/STSBenchmark>

Table 5.2: Pearson’s $r \times 100$ results on the STS-B dataset

Method	STS-B-dev		STS-B-test	
	<i>context</i>	<i>embed</i>	<i>context</i>	<i>embed</i>
BERT	64.78	-	54.22	-
BERT _{\M}	59.17	60.07	47.91	48.19
BERT _[CLS]	29.16		17.18	
uniLM	56.25	63.87	39.57	55.00
uniLM _[EOS]	40.75		38.30	
biLM	59.99	-	50.76	-
biLM _{\M}	53.20	58.80	36.51	49.08
T-TA	71.88	54.75	62.27	44.74
GloVe	-	52.4	-	40.6
Word2Vec	-	70.0	-	56.5

the embedding layer due to its unidirectional context constraint.

Since the uniLM encodes all contexts in the last token, [EOS], we also use the last representation as the sentence representation; however, this approach does not outperform the average sentence representation. Similarly, BERT has a special token, [CLS], which is trained for the “next sentence prediction” objective; thus, we also use the [CLS] token to see how this model learns the sentence representation, but it significantly underperforms the other models.

Results on SICK

We further evaluate the language models on the SICK dataset, which consists of (4934, 4906) sentence pairs with (training, testing) splits and scores ranging from 1 to 5. The results are in Table 5.3, from which we obtain the same observations as those reported for STS-B. “-” denotes an infeasible value, and bold font denotes the best performance on each sub-task.

Table 5.3: Pearson’s $r \times 100$ results on the SICK dataset

Method	SICK-test	
	<i>context</i>	<i>embed</i>
BERT	64.31	-
BERT _{\M}	61.18	64.63
uniLM	54.20	65.69
biLM	58.98	-
biLM _{\M}	53.79	62.67
T-TA	69.49	60.77

All results on unsupervised STS tasks demonstrate that the T-TA learns textual semantics best using the token-level BLA objective.

5.3.4 Settings: Unsupervised WiC

Depending on its context, an ambiguous word can refer to multiple, potentially unrelated, meanings. As mentioned earlier, mainstream static word embeddings, such as Word2vec and GloVe, are unable to reflect this dynamic semantic nature. Contextualised word embeddings are an attempt at addressing this limitation by computing dynamic representations for words which can adapt based on context.

We apply language models to the words in context (WiC) dataset, where the task is to identify the intended meaning of words [26]. WiC is framed as a binary classification task. Example instance in the WiC dataset is:

Target word: part
Sentence1: The government must do its part.
Sentence2: Religions in all parts of the world.
Label: False

This dataset can also be viewed as an application of Word Sense Disambiguation in practise.

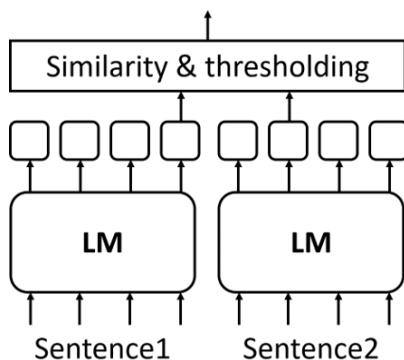


Figure 5.3: Example of procedure for unsupervised WiC.

In this section, we address the *unsupervised* WiC task to examine the inherent ability of each language model to obtain contextual language representations. Following [26], we use the most straightforward approach for the unsupervised WiC task

to focus on comparing token-level language representations. We first obtain contextualized representation of each word in the last layer, and then compute the cosine similarity of the two word representations in different sentences. After the pre-defined threshold, prediction can be obtained.

In this problem, we note that MLM-trained models like BERT and biLM can be directly applied by masking and predicting the target word for computing the contextual word representations in a given sentence. Therefore, it is hard to say the BLA-trained model (T-TA) is more efficient than MLM-trained models in the WiC task.

5.3.5 Results: Unsupervised WiC

Figure 5.3 shows an example of procedure for unsupervised WiC.

Table 5.4: Accuracy on the WiC dataset

Model	Accuracy (%)
BERT	63.8
BERT _{\M}	63.5
biLM	61.6
biLM _{\M}	58.5
T-TA	63.2

As mentioned above, there is no difference in computational complexity between the MLM-trained model and the BLA-trained model in the WiC task. Therefore, we should compare accuracy of contextualized representations of the target word obtained by each language modeling objective.

When using each language model, the train set is used for finding best threshold calibrated by a factor of 0.1 and test on the validation set. The results are in Table 5.4. In word-level contextualization task, T-TA outperforms biLM significantly. Furthermore, despite of its smaller model configuration, T-TA achieves competitive performance

with the bigger model, BERT.

In respect of unsupervised learning tasks, empirical results in this section support that contextualized language representations obtained from BLA-trained models are superior to those from MLM-trained models or ALM-trained models.

5.4 Experiments on Supervised Learning Tasks

Following the recent trend of pre-training and fine-tuning approach, we apply our BLA-trained models to the supervised tasks in addition to the unsupervised tasks. First, we adopt each language model to feature extraction for supervised learning tasks in order to see the inherent property of each pre-trained model. Then, we compare each language model in the fine-tuning approach for the same downstream tasks, and provide a guidance to use BLA-trained models according to the application purpose.

5.4.1 Language Model Setups

The main purpose of this section is to compare the proposed BLA-trained model with the MLM-trained model in supervised learning tasks. In this experiments, the BLA-trained model consists of $|L| = 4$ self-attention layers with $d = 768$ input and output dimensions, $h = 12$ attention heads, and $d_f = 3072$ hidden units for the position-wise feed-forward layers. For a fair comparison, we also build the MLM-trained model that has the same number of parameters with the BLA-trained model, but the only training objective is different.

For training, we create each training instance following the BERT. We add [CLS] and [SEP] tokens at the beginning and end of each sentence, respectively. Instead of a single sentence for each training instance as in the previous experiments, a pair of segments of sentences are concatenated with an additional [SEP] token between them. This segment-pair is swapped at random for the task of sentence order prediction (SOP), and it is used as the additional training objective as well as language modeling objectives, in order to learn sentence relationship. This is because many NLP tasks like Natural Language Inference (NLI) need to understand the relation of the two sentences.

Following BERT, the sentence representation of the MLM-trained model is obtained from the [CLS] token. However, the sentence representation of the BLA-trained model may be more suit to average pooling the sequence output of the final

layer. To that end, we additionally develop a more sophisticated technique that predicts sentence relation from not [CLS] but mean-pooled representations, and we denote this pre-training technique as SRP (abbreviation for sentence relation prediction with mean-pooling) in the following results.

Training texts are from English Wikipedia (2,500M words) and the BookCorpus (800M words) We use 32 sentences as the training batch and train all language models over $1M$ steps. As we set maximum sequence length to 128, $32 \times 128 \times 1M \approx 4G$ tokens are used to pre-training, which is approximately 1.25 epoch over the 3.3 billion word corpus. We train the language models with Adam [63] with an initial learning rate of $1e - 4$ and coefficients of $\beta_1 = 0.9$ of $\beta_2 = 0.999$; the learning rate is set to warm up over the first 10k steps, and the learning rate exhibits linear decay.

Table 5.5: Accuracy on each pre-training task.

Method	language modeling	sentence relationship
MLM-trained	0.585	0.816
BLA-trained	0.607	0.813

Compared to 3-layer models (Table 4.1), 4-layer models (Table 5.5) achieve better accuracy on each language modeling task.

5.4.2 Settings: Text Classification Tasks

To evaluate each language model, we apply the BLA-trained model and MLM-trained model on 6 text classification tasks: MNLI, QQP, QNLI, SST-2, MRPC, and RTE where all tasks are in the GLUE benchmark. While SST-2 is a single-segment classification task, others are sentence relationship classification tasks.

MNLI: Multi-Genre Natural Language Inference is a large-scale, crowdsourced entailment classification task [78]. Given a pair of sentences, the goal is to predict whether the second sentence is an entailment, contradiction, or neutral with respect to the first

one. The amount of training data of MNLI is about 392k.

QQP: Quora Question Pairs is a binary classification task where the goal is to determine if two questions asked on Quora are semantically equivalent [79]. The amount of training data of QQP is about 363k.

QNLI: Question Natural Language Inference is a version of the Stanford Question Answering Dataset [34] which has been converted to a binary classification task [80]. The positive examples are (question, sentence) pairs which do contain the correct answer, and the negative examples are (question, sentence) from the same paragraph which do not contain the answer. The amount of training data of QNLI is about 108k.

SST-2: The Stanford Sentiment Treebank is a binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment [53]. The amount of training data of SST-2 is about 67k.

MRPC: Microsoft Research Paraphrase Corpus consists of sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent [81]. The amount of training data of MRPC is about 3.5k.

RTE: Recognizing Textual Entailment is a binary entailment task similar to MNLI, but with much less training data [82]. The amount of training data of RTE is about 2.5k.

In text classification experiments, we train each model with Adam [63] with an initial learning rate of $2e - 5$ and coefficients of $\beta_1 = 0.9$ of $\beta_2 = 0.999$ during 3 epochs for each dataset with the pre-trained model weights.

5.4.3 Results: Feature Extraction

In feature extraction, pre-trained weights of each language model are frozen during supervised learning on downstream tasks as shown in Figure 5.4. For the task-specific network, we use a linear regression layer on top of the pooled output of each language model to see the potential of BLA-trained models on this feature-based ap-

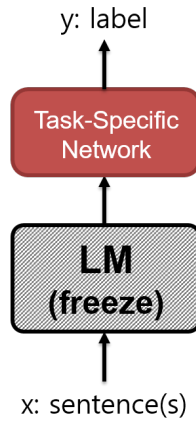


Figure 5.4: Feature extraction with language models for text classification tasks.

proach. Selected pooling strategies are the [CLS] pooling and mean pooling. We note that MLM-trained models are also fitted without using [MASK] during the supervised learning.

Table 5.6: Accuracies on text classification tasks.

Method	Pooling	MNLI	QQP	QNLI	SST-2	MRPC	RTE
BLA-trained	[CLS]	46.2	70.2	64.8	66.5	68.4	59.2
BLA-trained	mean	45.3	71.1	61.9	73.9	68.4	49.8
BLA-trained	SRP	48.8	73.9	71.3	73.5	68.4	58.5
MLM-trained	[CLS]	46.1	67.2	67.8	61.7	68.4	57.0
MLM-trained	mean	45.8	71.5	66.3	70.6	68.4	53.8
MLM-trained	pen.	48.5	71.0	72.7	70.6	68.4	47.7

When we see the tendency of BLA-trained models and MLM-trained models in Table 5.6, BLA-trained models are better or equal to MLM-trained models on 5 out of 6 classification tasks. Therefore, we can conclude that BLA-trained models are better than MLM-trained models for the usage of feature extraction on supervised learning

tasks. Also, the mean pooling is more suitable than the [CLS] pooling for the feature extraction approach. We note that “pen.” in Table 5.6 denotes that the output of the penultimate layer rather than the last layer is used for language representations with mean pooling following the literature [37], which reports that using intermediate layers for feature extraction is more effective than using the last layer of BERT.

We empirically demonstrate our argument that MLM-trained models are not guaranteed to output good contextualized language representations without fine-tuning the whole network. Instead, we encourage practitioner to use BLA-trained models for extracting contextualized language representations in word-level.

5.4.4 Results: Fine-tuning Approach

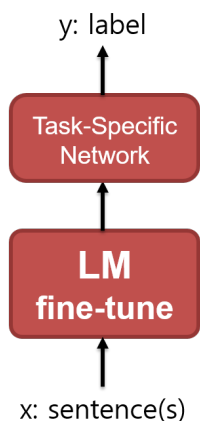


Figure 5.5: Fine-tuning approach for text classification tasks.

In fine-tuning approach, pre-trained weights of each language model are updated during supervised learning on each downstream task as shown in Figure 5.5. Following the literature [37, 16, 18], we use a linear regression layer for classification on top of the final hidden representations.

From results that MLM-trained models outperform BLA-trained models in all cases, we may conclude that BLA is improper to the BERT-like fine-tuning approach.

Table 5.7: Accuracies on text classification tasks.

Method	Pooling	MNLI	QQP	QNLI	SST-2	MRPC	RTE
BLA-trained	[CLS]	69.4	84.3	78.8	83.8	74.0	58.8
BLA-trained	mean	72.0	86.2	81.8	84.4	73.8	54.2
BLA-trained	SRP	72.2	86.2	82.8	85.2	74.7	58.8
MLM-trained	[CLS]	76.2	87.7	84.7	86.1	76.5	62.5
MLM-trained	mean	76.3	88.2	83.5	87.4	75.2	59.2

This is because the layer output of BLA-trained models does not include the token information due to its self-unknown property. Consequently, we recommend that use MLM-trained models for the fine-tuning approach.

5.5 Summary of BLA for Feature Extraction

This chapter empirically demonstrate that contextualized language representations learned from BLA are better than those from MLM as well as ALM. Experimental results on various NLP tasks, including unsupervised learning tasks as well as supervised learning tasks, demonstrate that BLA is the better option for feature extraction without fine-tuning the whole network. In conclusion, the proposed BLA objective is the best candidate for extracting contextualized language representations

Until now, we focused on demonstrating great potential of BLA. Since we trained smaller language models with less training samples, there is still much room for improvements in BLA-trained models. To achieve state-of-the-art performances on supervised learning tasks with the proposed learning objective, we should pre-train bigger language models than 3 or 4 layers using larger training examples than 16G words with bigger batch size than 64. We leave those enhancements for future works.

Chapter 6

Conclusions and Future Works

This dissertation mainly focused on understanding how powerful linguistic features obtained from pre-trained language models are by applying them to various NLP tasks. Throughout the dissertation, we demonstrated that the proposed bidirectional language autoencoding (BLA) is the better learning objective than autoregressive language modeling (ALM) or masked language modeling (MLM) for extracting reliable contextualized language representations with self-supervised learning on a large unlabeled text corpus.

To that end, we first confirmed that the importance of deep and bidirectional learning of language models for understanding a given text by showing MLM-trained models significantly outperform the ALM-trained models on re-ranking tasks in Chapter 3. Subsequently, we pointed out that MLM-trained models suffer from the speed-accuracy trade-off due to the training-inference discrepancy in MLM. To eliminate the trade-off of MLM, we introduced BLA and the corresponding deep bidirectional language model called Transformer-based text autoencoder (T-TA). Chapter 4 demonstrated the advantages of BLA-trained models over MLM-trained models in terms of inference speed and ALM-trained models in terms of accuracy. Finally, we confirmed that contextualized language representations from BLA-trained models are much better than those of MLM-trained models by applying each language model to various

NLP tasks in Chapter 5. In conclusion, we convinced that BLA is the key technique for extracting generic linguistic features.

To sum up, this dissertation provides a better option in language modeling for feature extraction.

Table 6.1: Comparison of bidirectional language modeling objectives.

Property	Speed	Accuracy
Feature Extraction	MLM \leq BLA	MLM \leq BLA
Fine-tuning Approach	MLM = BLA	MLM $>$ BLA

6.1 Future Works

Since the BLA-trained model is a totally different language model, it could be analyzed and be compared with BERT and other language models. For one example, analysis of layer-wise representation modifications from replacing a word in a given text or the same word used in a different context will provide a deep understanding of contextualization in each language model [83]. Probing task is another candidate for evaluating language representations of pre-trained language models [84, 85].

In the meantime, the performance of a language model depends on the model size, the corpus size, and the training time. Namely, having more layers, bigger batch size, and longer supporting sequence length is critical for the performance of language models. Therefore, it is desirable to enlarge the construction of the BLA-trained model to be equal to that of BERT. After these enhancements, we can check performances of BLA-trained models on General Language Understanding Evaluation (GLUE) benchmark and Stanford Question Answering Dataset (SQuAD) and compare them with those of the state-of-the-art pre-training methods such as RoBERTa, XLNet, T-5, *etc.*

As we verified the usefulness of BLA for sentence scoring, we could extend the

utilization of BLA to the *conditional* sentence scoring. Since many text generation tasks such as dialogue system and question answering are conditional on the context, scoring a sentence in company with the context could be a good metric. For example, to tackle the lack of meaningful automatic evaluation metrics for dialog that has impeded open-domain dialog research, a recent study presented USR, an UnSupervised and Reference-free evaluation metric for dialog [86]. In USR, the conditional MLM score is one of the key algorithms to measure how appropriate a generated response is given the previous dialog, as shown in Figure 6.1 We can replace this scoring method with our BLA, eliminating the mask-and-predict repetition while keeping the scoring performance. This scenario is similar to the invention of BLA in Chapter 4 for solving the inefficiency of the MLM sentence scoring method presented in Chapter 3.

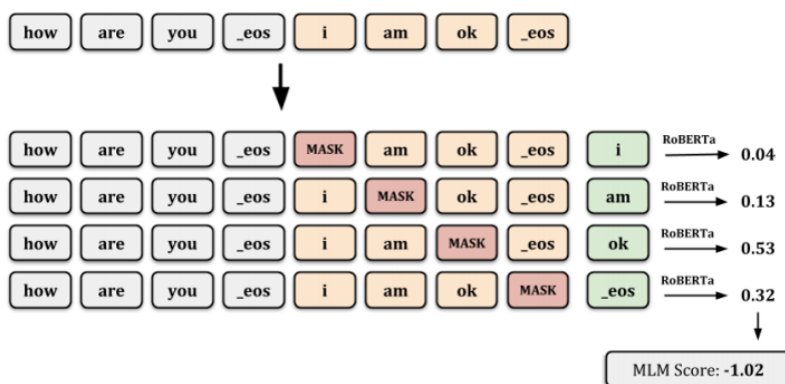


Figure 6.1: Conditional MLM score for text evaluation.

In addition to the text scorer, we can use BLA as a generator to train a discriminator, as shown in Figure 6.2. A study suggested replaced token detection, which is a pre-training task where the model learns to distinguish real input tokens from plausible but synthetically generated replacements [18]. It is called ELECTRA, the abbreviation for Efficiently Learning an Encoder that Classifies Token Replacements Accurately. They focus on the sample-efficient pre-training algorithm, and we believe that BLA is a more efficient generator than MLM. This is because BLA can sample all tokens

simultaneously, whereas MLM can sample only 15% tokens.

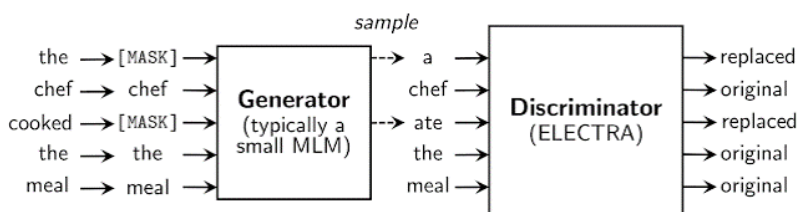


Figure 6.2: Generator with MLM for pre-training discriminator.

Besides, documents that are much longer than the pre-trained length are frequently addressed in real-world situations. Information retrieval and text summarization are representative examples of handling lengthy documents. When dealing with such long documents, fine-tuning MLM-trained networks may require too much computing resources. On the other hand, using BLA-trained networks, it can be efficient and effective to extract contextualized language representations for the proper-sized chunk and learn task-specific networks on top of them as shown in Figure 6.3.

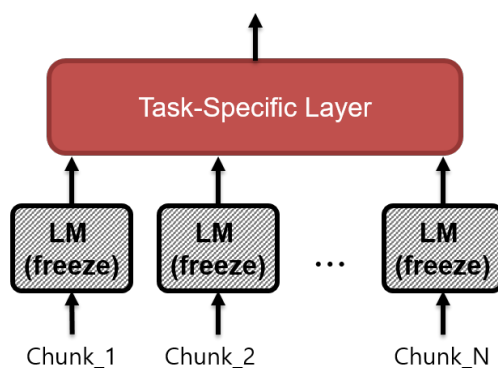


Figure 6.3: Illustration of BLA for long texts.

Like in those examples above, we believe that BLA and BLA-trained models have huge potentials for unsupervised linguistic feature learning and could be applied to many NLP applications.

Bibliography

- [1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, 2013, pp. 3111–3119.
- [2] Jeffrey Pennington, Richard Socher, and Christopher D Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.
- [4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [5] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur, “Recurrent neural network based language model,” in *Eleventh annual conference of the international speech communication association*, 2010.
- [6] Alex Graves, “Long short-term memory,” in *Supervised sequence labelling with recurrent neural networks*, pp. 37–45. Springer, 2012.
- [7] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, “Learning phrase representations using rnn encoder–decoder for statistical machine translation,” in

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724–1734.

- [8] Yoon Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014.
- [9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
- [10] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever, “Improving language understanding by generative pre-training,” 2018.
- [11] Alex Wang and Kyunghyun Cho, “Bert has a mouth, and it must speak: Bert as a markov random field language model,” in *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, 2019, pp. 30–36.
- [12] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau, “A c-lstm neural network for text classification,” *arXiv preprint arXiv:1511.08630*, 2015.
- [13] Joongbo Shin, Yanghoon Kim, Seunghyun Yoon, and Kyomin Jung, “Contextual-cnn: A novel architecture capturing unified meaning for sentence classification,” in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2018, pp. 491–494.
- [14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

- [15] Alexis Conneau and Guillaume Lample, “Cross-lingual language model pretraining,” in *Advances in Neural Information Processing Systems*, 2019, pp. 7059–7069.
- [16] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” in *Advances in neural information processing systems*, 2019, pp. 5753–5763.
- [17] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon, “Unified language model pre-training for natural language understanding and generation,” in *Advances in Neural Information Processing Systems*, 2019, pp. 13063–13075.
- [18] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning, “Electra: Pre-training text encoders as discriminators rather than generators,” in *International Conference on Learning Representations*, 2019.
- [19] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu, “Mass: Masked sequence to sequence pre-training for language generation,” in *International Conference on Machine Learning*, 2019, pp. 5926–5936.
- [20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer, “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020, pp. 7871–7880, Association for Computational Linguistics.
- [21] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu, “Exploring the limits

- of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.
- [22] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee, “Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks,” in *Advances in Neural Information Processing Systems*, 2019, pp. 13–23.
- [23] Joongbo Shin, Yoonhyung Lee, and Kyomin Jung, “Effective sentence scoring method using bert for speech recognition,” in *Asian Conference on Machine Learning*, 2019, pp. 1081–1093.
- [24] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.
- [25] Joongbo Shin, Yoonhyung Lee, Seunghyun Yoon, and Kyomin Jung, “Fast and accurate deep bidirectional language representations for unsupervised learning,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020, pp. 823–835, Association for Computational Linguistics.
- [26] Mohammad Taher Pilehvar and Jose Camacho-Collados, “Wic: the word-in-context dataset for evaluating context-sensitive meaning representations,” 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)*, 2019, pp. 1267–1273.
- [27] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

- [28] Rico Sennrich, Barry Haddow, and Alexandra Birch, “Neural machine translation of rare words with subword units,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1715–1725.
- [29] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al., “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
- [30] Taku Kudo and John Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 66–71.
- [31] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, “Enriching word vectors with subword information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.
- [32] Sepp Hochreiter and Jürgen Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [33] Erik F Sang and Fien De Meulder, “Introduction to the conll-2003 shared task: Language-independent named entity recognition,” *CoNLL*, 2003.
- [34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, “Squad: 100,000+ questions for machine comprehension of text,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.
- [35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, pp. 9, 2019.

- [36] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “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)*, 2019, pp. 4171–4186.
- [38] Wilson L Taylor, ““cloze procedure”: A new tool for measuring readability,” *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.
- [39] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut, “Albert: A lite bert for self-supervised learning of language representations,” in *International Conference on Learning Representations*, 2019.
- [40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [41] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.
- [42] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin, “Understanding and improving layer normalization,” in *Advances in Neural Information Processing Systems*, 2019, pp. 4381–4391.
- [43] Vinod Nair and Geoffrey E Hinton, “Rectified linear units improve restricted boltzmann machines,” in *ICML*, 2010.

- [44] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *ICLR*, 2016.
- [45] Dan Hendrycks and Kevin Gimpel, “Gaussian error linear units (gelus),” *arXiv preprint arXiv:1606.08415*, 2016.
- [46] Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih, “Dissecting contextual word embeddings: Architecture and representation,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 1499–1509.
- [47] Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich, “Why self-attention? a targeted evaluation of neural machine translation architectures,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4263–4272.
- [48] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn, “Scalable modified kneser-ney language model estimation,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2013, vol. 2, pp. 690–696.
- [49] Ebru Arisoy, Abhinav Sethy, Bhuvana Ramabhadran, and Stanley Chen, “Bidirectional recurrent neural network language models for automatic speech recognition,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5421–5425.
- [50] Xie Chen, Anton Ragni, Xunying Liu, and Mark JF Gales, “Investigating bidirectional recurrent neural network language models for speech recognition.,” in *INTERSPEECH*, 2017, pp. 269–273.
- [51] Alex Graves and Jürgen Schmidhuber, “Framewise phoneme classification with bidirectional lstm and other neural network architectures,” *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

- [52] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.
- [53] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Citeseer, 2013, vol. 1631, p. 1642.
- [54] Tianxing He, Yu Zhang, Jasha Droppo, and Kai Yu, “On training bi-directional neural network language model with noise contrastive estimation,” in *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2016, pp. 1–5.
- [55] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, “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)*, 2018, vol. 1, pp. 2227–2237.
- [56] Yangyang Shi, Martha Larson, Pascal Wiggers, and Catholijn M Jonker, “Exploiting the succeeding words in recurrent neural network language models,” in *INTERSPEECH*, 2013.
- [57] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan, “Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm,” *Proc. Interspeech 2017*, pp. 949–953, 2017.
- [58] Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa, “Byte pair encoding: A text

compression scheme that accelerates pattern matching,” Tech. Rep., Technical Report DOI-TR-161, Department of Informatics, Kyushu University, 1999.

- [59] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson-Enrique Yalta Soplín, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al., “Espnet: End-to-end speech processing toolkit,” *Proc. Interspeech 2018*, pp. 2207–2211, 2018.
- [60] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, “Attention-based models for speech recognition,” in *Advances in neural information processing systems*, 2015, pp. 577–585.
- [61] Matthew D Zeiler, “Adadelata: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.
- [62] Dan Hendrycks and Kevin Gimpel, “Bridging nonlinearities and stochastic regularizers with gaussian error linear units,” *arXiv preprint arXiv:1606.08415*, 2016.
- [63] Diederik Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [64] Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff, “Masked language model scoring,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020, pp. 2699–2712, Association for Computational Linguistics.
- [65] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.
- [66] Rico Sennrich, Barry Haddow, and Alexandra Birch, “Improving neural machine translation models with monolingual data,” in *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016, pp. 86–96.

- [67] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” *arXiv preprint arXiv:1910.01108*, 2019.
- [68] Alvaro Peris and Francisco Casacuberta, “A bidirectional recurrent neural language model for machine translation,” *Procesamiento del Lenguaje Natural*, vol. 55, pp. 109–116, 2015.
- [69] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, “Lstm neural networks for language modeling,” in *Thirteenth annual conference of the international speech communication association*, 2012.
- [70] William H DuBay, “The classic readability studies.,” *Impact Information, Costa Mesa, California.*, 2006.
- [71] Kenneth Heafield, “Kenlm: Faster and smaller language model queries,” in *Proceedings of the sixth workshop on statistical machine translation*. Association for Computational Linguistics, 2011, pp. 187–197.
- [72] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al., “Tensor2tensor for neural machine translation,” in *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, 2018, pp. 193–199.
- [73] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher, “Learned in translation: Contextualized word vectors,” in *Advances in neural information processing systems*, 2017, pp. 6294–6305.

- [74] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [75] Andrew M Dai and Quoc V Le, “Semi-supervised sequence learning,” in *Advances in neural information processing systems*, 2015, pp. 3079–3087.
- [76] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia, “Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation,” in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 1–14.
- [77] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli, “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)*, 2014, pp. 1–8.
- [78] Adina Williams, Nikita Nangia, and Samuel Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1112–1122.
- [79] Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao, “Quora question pairs,” 2018.
- [80] Wei Wang, Ming Yan, and Chen Wu, “Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 1705–1714.

- [81] William B Dolan and Chris Brockett, “Automatically constructing a corpus of sentential paraphrases,” in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.
- [82] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo, “The fifth pascal recognizing textual entailment challenge.,” in *TAC*, 2009.
- [83] Kawin Ethayarajh, “How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, Nov. 2019, pp. 55–65, Association for Computational Linguistics.
- [84] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel Bowman, Dipanjan Das, et al., “What do you learn from context? probing for sentence structure in contextualized word representations,” in *7th International Conference on Learning Representations (ICLR)*, 2019.
- [85] John Hewitt and Christopher D. Manning, “A structural probe for finding syntax in word representations,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, June 2019, pp. 4129–4138, Association for Computational Linguistics.
- [86] Shikib Mehri and Maxine Eskenazi, “USR: An unsupervised and reference free evaluation metric for dialog generation,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020, pp. 681–707, Association for Computational Linguistics.

초 록

딥 뉴럴 네트워크는 비정형 텍스트 데이터에서 상황에 맞는 정보를 유연하게 다룰 수 있기 때문에 자연어 처리 분야에서 강력한 도구로 활용되고 있다. 큰 말뭉치에서 감독되지 않는 방법으로 학습한 언어 표현은 딥 뉴럴 네트워크가 문맥 정보를 더 잘 활용할 수 있도록 원천을 제공한다. 자연어 이해는 대표적인 자가지도학습 기술인 언어 모델링을 사용하여 문맥화된 언어 표현을 배움으로써 사전 훈련된 (비 문맥적) 단어 임베딩을 넘어 괄목할 만한 발전을 이루었다. 언어 모델링은 크게 자기 회귀 언어 모델링과 마스킹된 언어 모델링으로 분류될 수 있으며, 최신의 사전 훈련 방법들 또한 이 두 언어 모델링에 기반하고 있다. 본 학위 논문은 두 언어 모델링의 장점을 모두 취하는 새로운 언어 모델링 양방향 언어 오토인코딩을 제시한다. 제시된 양방향 언어 오토인코딩은 마스킹된 언어 모델링처럼 깊은 양방향 언어 이해를 가능하게 하고, 동시에 자기회귀 언어 모델링처럼 미세조정없이 문맥화된 언어 표현을 추출해서 사용할 수 있게 한다. 본 학위 논문에서는 양방향 언어 오토인코딩을 학습을 가능하게 하기 위한 새로운 뉴럴 네트워크 구조를 설계한다. 제시된 양방향 언어 모델은 단순한 복사가 아닌 유용한 언어 표현을 학습할 수 있도록 하며, 각각의 단어가 문맥화된 표현을 갖게 하여 정보에 손실이 없도록 한다. 본 논문의 주요 공헌은 새로운 양방향 언어 모델을 제안하여 자연어 이해 문제에 있어서 문맥화된 언어 표현을 추출해서 사용할 때에 기존보다 좋은 방안이 될 수 있음을 검증한 것에 있다. 실험 결과는 *N*-베스트 목록 재순위, 의미론적 텍스트 유사성 검사, 단어 의미 중의성 해소, 그리고 텍스트 분류에 대해 제시되며, 제안된 기법이 이전 기법들 보다 나은 장점을 보여준다.

주요어: 딥 뉴럴 네트워크, 언어 모델링, 비지도 학습, 문맥화된 언어 표현
학번: 2014-21625

ACKNOWLEDGEMENT

I would like to thank the MILAB members who provided insight and expertise that greatly assisted the research. In addition, I would like to express my gratitude to professor K. Jung for the advisory and discussion. To conclude, I cannot forget to thank my family, Sunwoo and parents, to support me spiritually with unconditional love throughout this very intense academic year.