University of Sheffield

A stitch in BERT saves Machine Learning!

Xuan-Rui Fan

Supervisor: Aline Villavicencio

A report submitted in fulfilment of the requirements
for the degree of BSc in Artificial Intelligence and Computer Science BSc

in the

Department of Computer Science

February 22, 2023

Declaration

Name: Xuan-Rui Fan

Signature:

Xuan-Rui Fan

Date: 11 May 2022

Abstract

Some expressions can be ambiguous between literal and idiomatic interpretations depending on the context. In nearly all Natural Language Processing (NLP) tasks, existing methods of constructing representations of expressions mainly rely on putting together the meaning of each constituent word, assuming the semantics of a phrase or sentence can be literally composed of them, thus failing to capture the nuances of some idiomatic expressions. This shortcoming leads to the loss of linguistic presence which is detrimental to downstream tasks, hence, classification of literal or idiomatic usage of a phrase is of crucial importance for NLP tasks. Training a classifier for detecting idiomaticity always requires tons of labelled data, but there are always ways for us to tackle the problem. This paper introduces some prompt based methods trying to address this problem, and experiments with these prompt based methods for classifying the intended usages of idioms, with only a limited number of training samples.

Contributions and Publications

The code in this dissertation was modified and deployed for the experiments in the paper, “Sample Efficient Approaches for Idiomaticity Detection”, accepted by the 18th Workshop on Multiword Expressions (MWE 2022). In that paper, some sample efficient methods were explored and a few impressive analyses along with clean conclusions were conducted.

Acknowledgements

This work would not have been possibly done without the guidance of Dr. Aline Villavicencio, my direct supervisor, giving me an opportunity to do the paper, providing me enthusiastic encouragement and useful critiques of this research work, and Dr. Harish Tayyar Madabushi, providing importantly valuable feedback closely related to my topic of this paper. I would like to express my very great appreciation to Dr. Harish Tayyar Madabushi for his valuable and constructive suggestions during the early implementation of this research work. His willingness to give his time so generously has been gratefully appreciated.

I am also grateful for the insightful suggestions offered by the PhD students of Dr. Aline Villavicencio and Dr. Harish Tayyar Madabushi, and the department for their help in offering me the computer resources.

Last but not the least, I wish to thank my dear and beloved family and Shu-Ting, for providing heartwarming mental supports during this difficult time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Idiomatic phrase is common and provides unique challenges for language models. For example, a *wet blanket* can be literally referred as a moist piece of woollen or similar material used as a covering, but also can be a person who spoils other people's fun, idiomatically. Such lexical composition can shift the meanings of the constituent words and introduce implicit information (Shwartz et al., 2019) [30], making them "a pain in the neck" for NLP applications, such as Machine Translation, Word Sense Disambiguation and Sentiment Analysis (Sag et al., 2002)[27].

To dig into the issue of the phrases, people tried to investigative the structure of them. For example, Diab and Bhutada (2009) purposed a supervised method using YamCha (a text chunker) (Kudo et al., 2001)[39] and Inside–outside–beginning tagging (IOB), dividing phrases into small chunks[10] and classifying them as idiomatic or literal in context. They used different window sizes for context ranging, and much attention was devoted to examining the window size and the formats of word chunks.

Different from studying the phrase structures, some researchers tried to employ the relations of the words. Sporleder and Li (2009)[31] used two unsupervised classifiers, relying on the strength and distances between words, with Normalized Google Distance[7] to calculate relatedness with a search engine. Specifically, they determined whether component words participate in any of the literally related words and observed how literal relation changes when the expression is inserted or left out.

Likely, Peng and Feldman (2016) experimented with idiom detection with different distance calculation methods[23], such as term frequency-inverse document frequency (TF-IDF) and other statistical analyses, based on the hypothesis of distributions and representations of the contexts of the idiomatic phrases will be different from the contexts of the literal usages. Some other deep networks with different structures were also used, Liu et al. (2017) compared the words' context with the use of TreeLSTM[33][18] (tree-structured long short-term memory network), a network that processes sentences words by words, implemented a supervised idiomaticity detector.

With increasing trend of computing power, some methods have allowed tons of text to be trained and represented in a numerical way, for example, CBOW[20]. Liu and Hwa (2019) purposed a learning method for recognizing the intended usages of idioms, transferring similarity scores into soft labels and calculating their usage with Latent Dirichlet Allocation[5] and Naive Bayes including some other linguistic features[17]. They determined usage with the semantic similarity between the context of the instance and the literal representation (high dimensional vectors spaces of words) via trained CBOW, then calculated the semantic similarity score between the context of the instance and the literal usage representation, in order to classify them.

These methods all failed to capture some nuances of multi word expressions, mostly since they concentrate only the words' or chunks' neighbour words. These methods rarely make use of the intact information of the whole sentence but only direct the efforts on the relations of the words, although the structure of LSTM (long short-term memory network) and its derivatives, LSTM + attention alleviate the problem of long-term dependence to a certain extent, they are still powerless for particularly long-term dependence.

It was then advances for coping with natural languages come into being. Some researchers noticed the defect of such sequential models. Transformer[36] was introduced by Vaswani et al (2017). The sequences of text are inputted into its network at once instead of one token by one token, and only a part of the input sentence is focused on. This addressed the problem of long sequences and because of its structure, it does not take long to train a transformer, compared to sequential models such as recurrent neural networks (RNNs) that take words and learn them words by words.

Recently, with the developments of Transformer, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018)[9] and Generative Pre-trained Transformer 3 (GPT-3) (Brown et al., 2020)[6], using these pre-trained large language models to do NLP downstream tasks has become an art. "Standing on the Shoulders of Giants" is a good metaphor to describe the exciting potential of using them, saying the models can be rapidly fine-tuned on a specific downstream task with relatively few labels after pre-training in an unsupervised way on a massive amount of text data since the general linguistic patterns and the common knowledge have already been learnt during BERT's pre-training phase.

1.1 Aims and Objectives

It is really common for people to use the models' word-embedding features to convert sentences into their vector-space representations while training a text classifier or comparing their similarities. This seems to be a mainstream practice to deal with words, however, some of the information that the language models encoded is unfortunately not used.

To compare the accuracy of prompt based methods with a general BERT classifier when classifying phrases, prompt based methods and a pre-trained model are used. The objective of the tasks is to experiment with the semi-supervised and supervised methods, on the performance of idiom detection, by giving instances. In detail, the methods of Pattern-Exploiting Training (PET), Iterative PET (iPET)[28] and A Densely-supervised Approach to Pattern Exploiting Training (ADAPET)[34] have been deployed.

The reason the prompt based methods is of central concern is the gap between the pre-training stage and the downstream task can be significantly different. For example, in classic binary classification for BERT, a set of additional parameters are introduced (the new network for classifier). Whereas prompt based learning allows downstream tasks to take the same format as the pre-training objectives (Masked language modelling tasks (MLM tasks)) and requires no new parameters. In particular, this allows us to utilise such MLM tasks to perform classification, by providing language models with a prompt with cloze question style format. Additionally, using prompt based methods could also allow unleashing the commonsense[35] of language models have learnt from tons of text to some extent.

As few-shot learning methods, PET and iPET require a few proportions of annotated data to fine-tune language models and label its unlabelled dataset, given them prompts and verbalizers mapped to their labels, then all the data is afterwards used to train a new classifier. Beside, ADAPET simply provides a better way to fine-tune language model, and the fine-tuned language model is directly used to perform text classification. It is claimed to perform better than PET and iPET in some tasks.

By making use of PET and its variants, iPET and ADAPET, the expectation is they could deliver better results in such classification tasks, and overall iPET should give better results. Different prompts are expected to affect how the language models modeling, consequently elevating or demoting the overall performances.

Therefore, different amount of annotated data is tested and justified to be suitable for these models. In addition, the qualities of prompts used in the experiments are determined by the results. Results are obtained by examining different amounts of training samples with different prompts and a basic pre-trained model, BERT-base-uncased.

1.2 Overview of the Report

The current dissertation consists of five chapters and it has been divided as follows:

Chapter 2: Literature Survey. Idioms and Multiword expressions are explained here. In addition, LSTM + attention, Transformer, BERT, PET, iPET, and ADAPET have been introduced in this section.

Chapter 3: Methods and Analysis This section covers the dataset used for experiments, the analysis of prompts used and preprocessing steps on the dataset for each model, and also, the performance metrics used for evaluating the models, with some patterns used in PET and their variants explained in this section.

Chapter 4: Results and Discussions This section covers the results of the experiments on each learning model with the language model, each result has also been discussed.

Chapter 5: Conclusions and Future work This section covers the summary of experiments and gives a further improvement at the end of the work.

Chapter 2

Literature Survey

In this chapter, Multiword expression (MWE), a key problem in the domain of Natural Language expression, has been explained briefly. In addition, following that is focused on the continuum of idiomaticity. Furthermore, the procedures, Pattern-Exploiting Training and its variants have also been explained. Some related works have also been mentioned in this chapter.

2.1 Multiword Expression

MWEs are very roughly defined as "idiosyncratic interpretations that cross word boundaries (or spaces)"[27]. They are word combinations with linguistic properties that cannot be predicted from the properties of the constitutional words or the way they have been combined[21]. The proportion of MWEs in WordNet 1.7 (Fellbaum 1998) is 41%, and it will likely rise as the language system adds vocabulary for new domains and as each new domain adds more MWEs than simplex words[27]. MWEs pose significant problems for every kind of Natural Language Processing (NLP) task, including the overgeneration problem and the idiomaticity problem. In this paper, it is focused on the idiomaticity problem.

2.2 Idiom

MWEs' meanings may not be directly related to the meanings of their individual words, Therefore, the challenge is for language models to capture the nuances between idiomaticity and literality. A hypothesis is distributions of the contexts of the idiomatic phrases will be different from the contexts of the literal usages[23]. Being influenced by the principle of compositionality, the traditional approaches based on it are also influenced by theories of generative grammar[12]. There are also methods that combined the use of taggers[11], trying to catch the divergence between literal and idiom usages. Firstly, simply computing the similarity between words may fall into many problems, for instance, a word may be related

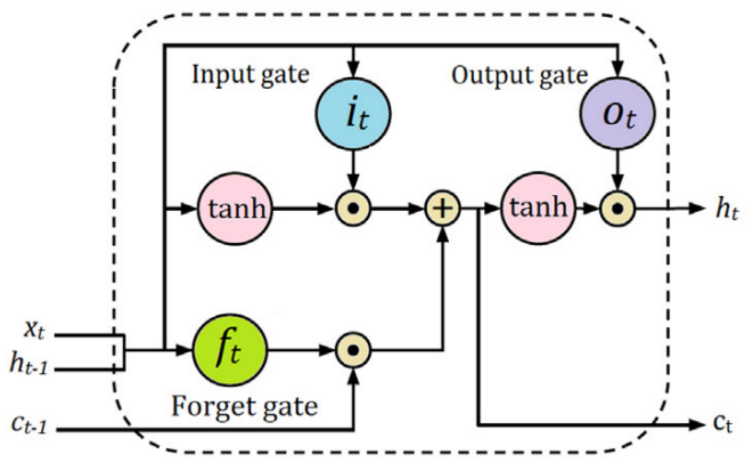

Figure 2.1: LSTM Cell. Source:[4]

to another with real world knowledge but they are far away from each other in vector spaces (e.g., *a football player's name* and *something corresponded to kicking*)[31]. In addition, it is also a challenge to decide which words in a phrase are involved to calculate similarity.

2.3 LSTM + Attention

LSTM was once the solution to NLP problems in past days. It was meant to solve the problem of gradient disappearance and gradient explosion during long sequence training, performing better in longer sequences than ordinary RNNs. As a sequential model, it introduced extra parameters to store the previous states while handling new inputs. With these parameters, the performance of this network is slightly improved compared to RNNs but it still failed to treat particularly long sequences; moreover, these new parameters caused LSTM difficult to train, additionally, the unusual gradient paths and the structure of the network leads to an extremely long training time. Even though there are researchers trying to fatherly improve LSTM with the combination of it and the attention mechanism, the fact of the improper structure of NLP tasks is still an insurmountable obstacle.

2.3.1 LSTM

Different from RNNs, LSTM networks are a type of RNN that additionally uses new parameters: forget gate, input gate and output gate, as illustrated in Figure 2.1, where x refers to the input, h refers to the hidden state, c refers to the cell state at the time t. These gates respectively act roles to delete unnecessary information, selecting useful information, and deciding which pieces of information is the current output.

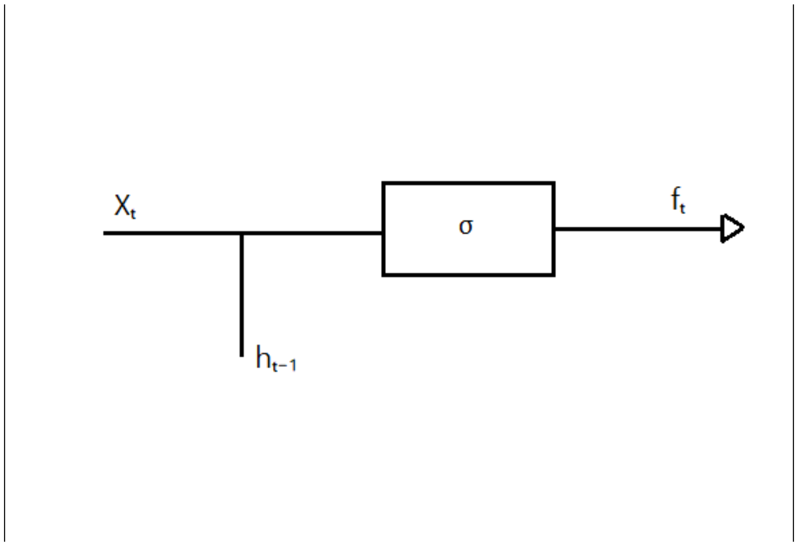

Figure 2.2: forget gate of LSTM

Forget gate

Forget gate removes information that is no longer necessary for the completion of the task, as illustrated in Figure 2.2. The f_t is calculated with Equation 2.1, where b_f indicates the bias of f:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

Input gate

Input gate is responsible for adding information to the cells, as illustrated in Figure 2.3. It includes two parts, i_t and C_t, which are calculated with Equation 2.2 and 2.3, where b_i and b_C indicate the biases of i and C:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$C_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.3}$$

Output gate

Output gate is responsible for selecting and outputting necessary information, as illustrated in Figure 2.4. It also includes two parts, o_t and h_t, which are calculated with Equation 2.4 and 2.5, where b_o indicates the bias of o:

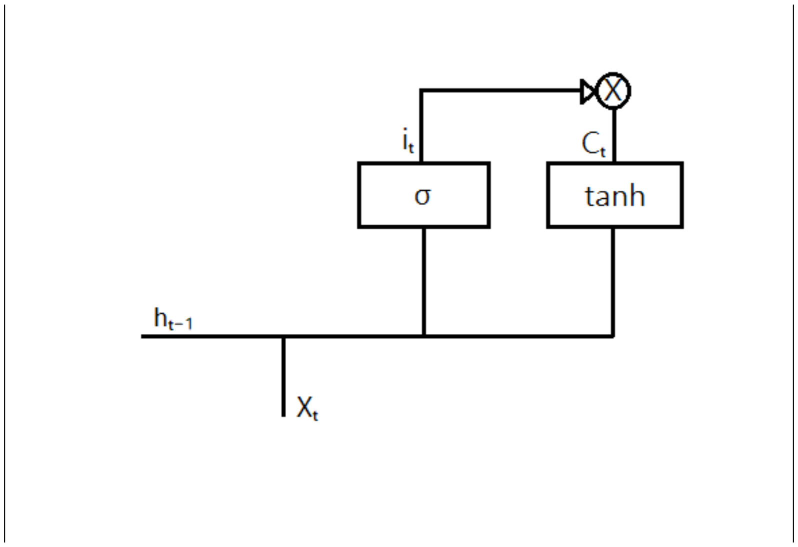

Figure 2.3: input gate of LSTM

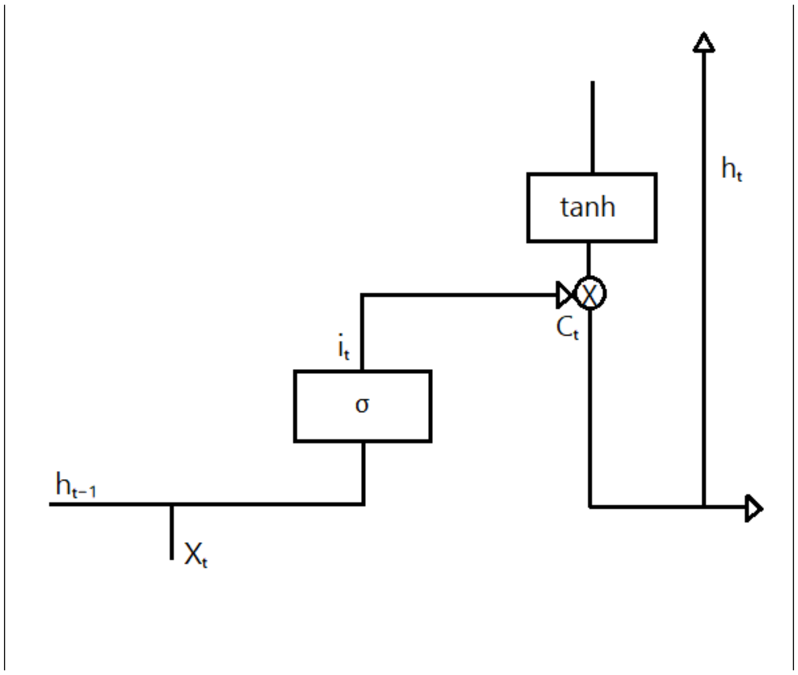

Figure 2.4: output gate of LSTM

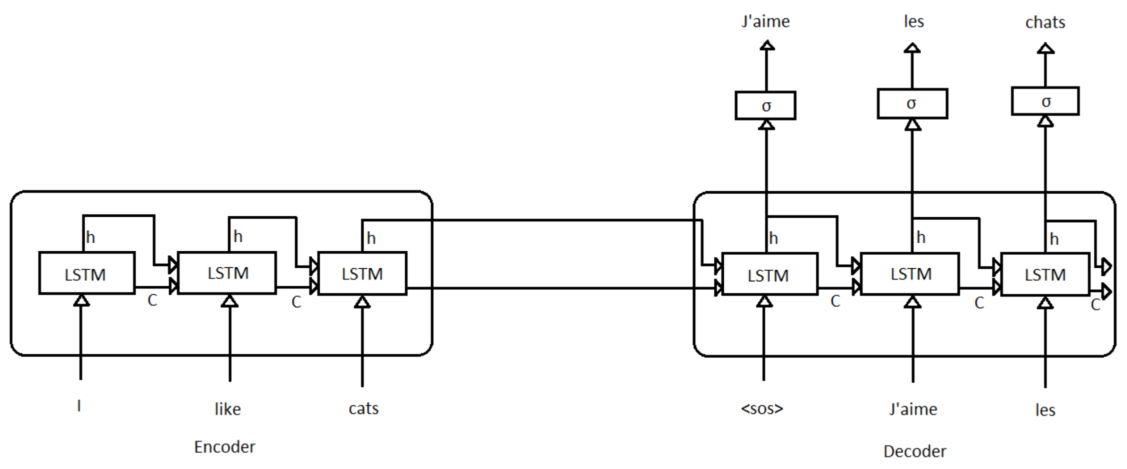

Figure 2.5: structure of Encoder-Decoder LSTM

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.4}$$

$$h_t = o_t * \tanh(C_t) \tag{2.5}$$

2.3.2 Encoder-Decoder LSTM

A sequence to sequence model has two components, an encoder and a decoder. The encoder encodes a source sentence to a context vector that represents context, where the decoder takes in the context vector from the encoder as an input and computes the translation using the encoded representation, as illustrated in Figure 2.5. It is mainly used in transferring input sequence into target sequence (Sequence-to-sequence), such as machine translation, speech conversion, but its character faults are also obvious. Take machine translation for example, the encoder and decoder does not selectively focus on relevant input words while generating each output word, but each output word is expected to be more influenced by some specific parts of the input sequence; in addition to that, context vector is also expected to convey the information of subject, objects, and their relations. The bottleneck of this context vector is knotty especially for long sequences.

2.3.3 Attention

Attention[3] was firstly proposed by Bahdanau et al. (2014) to regulate the problem mentioned above. Instead of using only context vector, attention uses additional matrix, W and U, to calculate V with the Equation 2.6. Afterwards, the values are normalized using a softmax function as e, and these normalised e are then multiplied with h to produce a

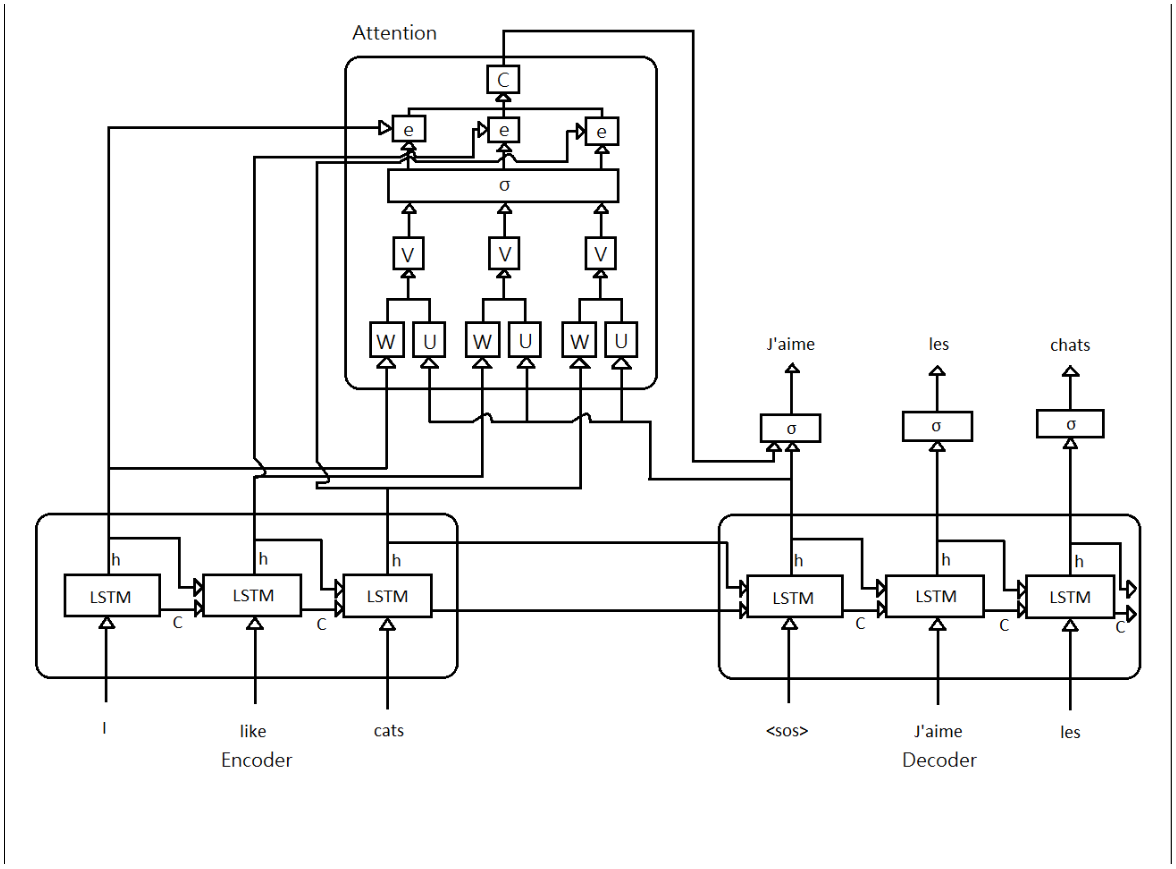

Figure 2.6: structure of Encoder-Decoder LSTM with attention

new attended context vector from which the current output time step can be decoded. The context vector thus obtained is a weighted sum of the annotations and normalized alignment scores. This main idea is illustrated in Figure 2.6.

$$V = align(W, U) \tag{2.6}$$

2.4 Transformer

Transformer based models have primarily replaced LSTM, and it has been proved to be superior in quality for many sequence-to-sequence problems. Transformer was initially introduced to solve machine translation tasks, detaching from sequential models such as RNNs and LSTMs inputting a single word at each step in a sequential manner to generate an output one word at a time, transformer is accomplished by using self-attention. Avoiding the black box problem in deep learning, the way to understand the mechanism behind it is easier and it is closer to how humans read and process natural language and text. In addition, the computational efficiency by parallelising the operations can be achieved by using this

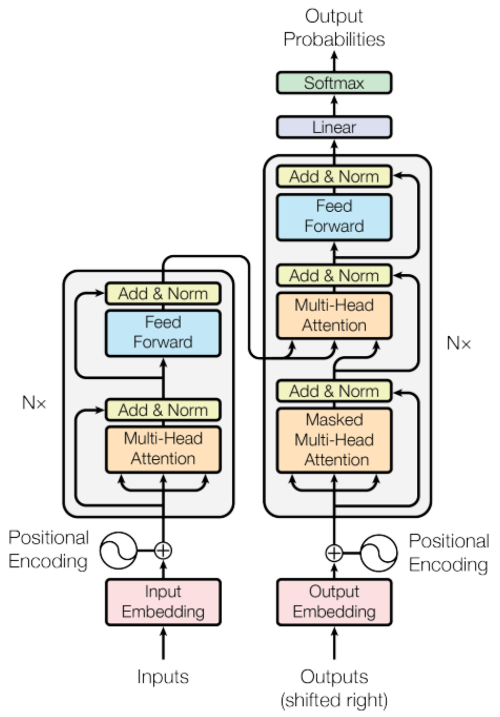

Figure 2.7: Transformer model architecture. Source:[36]

self-attention principle, which is not possible in sequential modelling. The architecture of the model is illustrated in Figure 2.7.

2.4.1 Self-attention

In a given sentence, not all words require equal importance, because some words play a better role in conveying the meaning and carrying more information. Self-attention simply helps to focus on those important words in the sentence, thereby allowing the model to encode the context. Self-attention can be described as mapping a query and a set of key-value pairs to an output, describing the relations of all the words to each other. The weight assigned to each value is computed by a compatibility function of the query with the corresponding key, afterwards, the output is computed as a weighted sum of the values. For example, in the sentence:

Example 1: the animal didn't cross the street because **it** was too tired.

For computers, it is not simple to know what the "it" in the sentence refers to, but attention is meant to help the models to encode that. For each word in the sentence, query can be regarded as the current word, value is known to capture the information present in the input

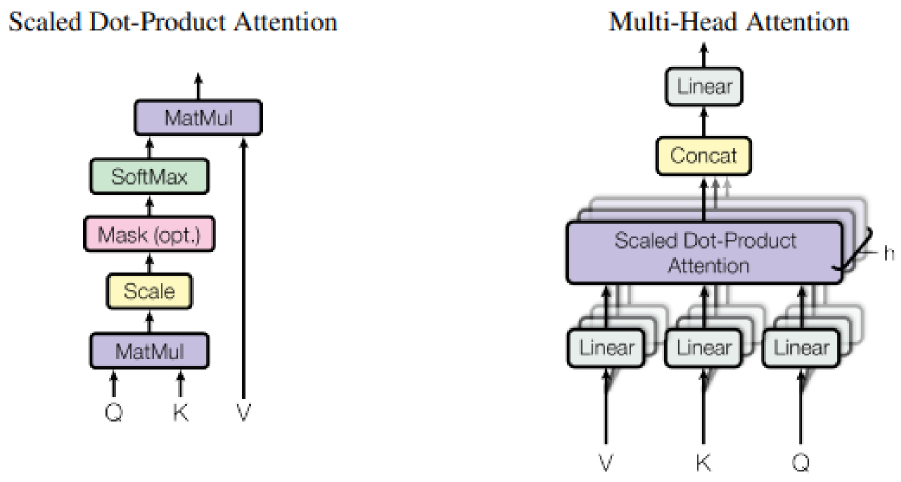

Figure 2.8: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. Source:[36]

word, and key can be seen as an indexing mechanism for the value. By calculating these parameters, the model captures the relations of words and knows that in **Example 1** the word "it" has a close relationship with the word "animal".

Scaled Dot-Product Attention

The input of scaled dot-product attention consists of queries and keys of dimension d_k, and values of dimension d_v. The matrix of attention is calculated with the Equation 2.7, with queries, keys, and values packed together as matrix Q, K and V:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.7}$$

Multi-Head Attention

Multi-head attention is a parallelised version of scaled dot-product attention to improve efficiency and accuracy, without adding the number of total parameters. By linearly projecting the d_{model}-dimensional query, key and value matrix h to $Q \in R^{m \times d_k}$, $K \in R^{m \times d_k}$ and $V \in R^{m \times d_v}$, and reproduce this procedure h times, the once again projected result is found to be beneficial. The concept is illustrated in Figure 2.8.

There are three multi-head attention modules in the encoder-decoder structure of Transformer, as illustrated in Figure 2.7. The first one in the encoder is in the encoder, mainly

Masking Rates			Dev Set Results		
Mask	Same	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	95.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Table 2.1: Ablation over different masking strategies. Source:[9]

focusing on understanding the context of the input, the second one is in the decoder, mainly on understanding the generative procedure of the context, and the third one is between the encoder and decoder, responsible to interpretation between the encoder and decoder.

Since attention is done by calculating them for each word corresponding to all words, no matter how long the distance between them is, the path lengths between them will be normalised, which allows the model to capture long-distance dependencies.

2.5 BERT

Bidirectional Encoder Representations from Transformers is a transformer-based machine learning model, with numbers of encoder layers and attention heads stacked up together. It was pre-trained with Wikipedia database and book collection database on two tasks: masked language modelling (MLM) and next sentence prediction (NSP).

2.5.1 MLM

In MLM, 15% of tokens were masked and BERT was trained to predict them from context. The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, and then the model attempts to predict the original words, based on the context provided by the non-masked sequence. Since the token to represent the masked words, [MASK], does not appear during fine-tuning, the authors of BERT do not always replace masked words with the actual [MASK] token. If the i^{th} token is chosen to be masked, it will be replaced with (1) the [MASK] token 80% of the time (2) a random token 10% of the time (3) the unchanged i^{th} token 10% of the time. Afterwards, it will be used to predict the original token with cross entropy loss. The Ablation over different masking strategies is shown in Table 2.1.

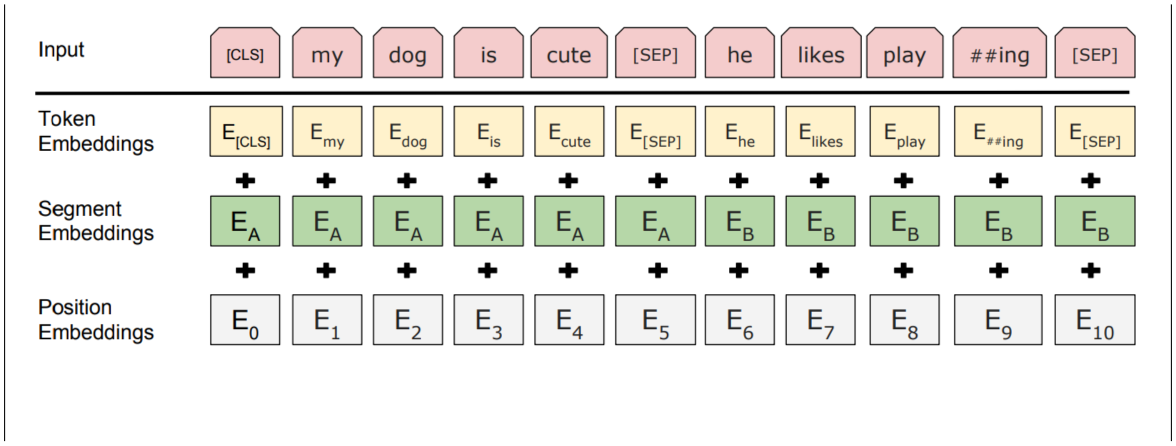
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
E[CLS] E my E dog E is E cute E[SEP] E he E likes E play E ##ing E[SEP]
Segment Embeddings
E A E A E A E A E A E A E B E B E B E B E B
Position Embeddings
E 0 E 1 E 2 E 3 E 4 E 5 E 6 E 7 E 8 E 9 E 10

Figure 2.9: BERT input representation. [9]

2.5.2 NSP

In NSP, BERT was trained to predict if a chosen next sentence was probable or not given the first sentence. When choosing the sentences A and B for each pre-training example, 50% of the inputs are a pair in which B is the subsequent sentence in the original document, and the other 50% of the inputs are random sentences from the corpus. NSP pre-training task is found leading to a higher performance to language models when it is removed, since in BERT only 2 sentences are inputted and the model fails to learn some long dependencies between words. This gap is addressed by improving it or replacing it with other tasks in other BERT's variant models such as RoBERTa[19] (liu et al., 2019) and ALBERT[16].

2.5.3 Bert Classifier

BERT can be used for a wide variety of downstream tasks. The classic case is that BERT can be used to train a classifier, as illustrated in Figure 2.10. A new set of parameters is introduced, usually a fully connected network, and it takes the first token of BERT's output as its input, which concluded all the information of the whole sentence, and other encoded information learnt from its MLM pre-training task is abandoned. For BERT-base, the number of new parameters of the new network is at least 768×2.

2.6 PET, iPET, and ADAPET

2.6.1 PET

Pattern-Exploiting Training[28] is a semi-supervised method proposed by Timo Schick et al., to leverage pre-trained language models to label data for downstream tasks. In BERT, as one of its pre-training tasks, Masked-Language Modeling (MLM) consists of providing BERT with a sentence with some of its words masked off and optimizing the weights inside BERT, in order for BERT to output the same sentence on the other side. However, in the phase of

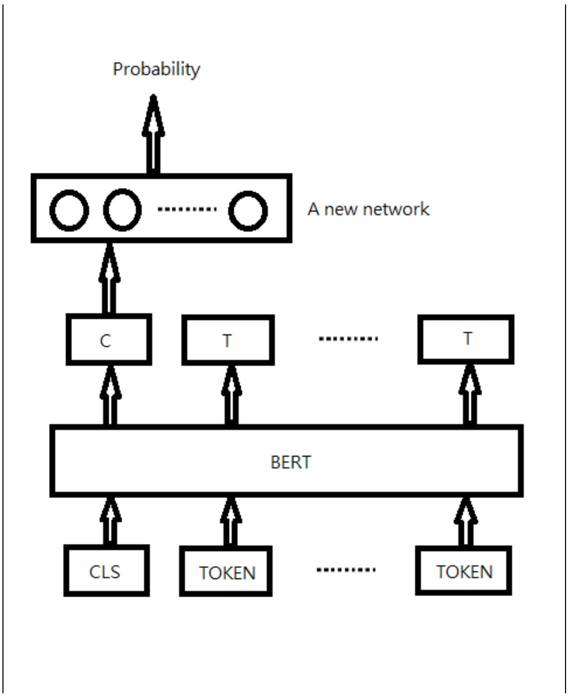

Figure 2.10: BERT Classifier

fine-tuning (Text Classification), the parameters that are learnt in MLM task are abandoned. In order to also make use of these parameters, PET introduces "patterns" that escort the language model to label the outline of sentences, including a "verbalizer" that maps from the language model's vocabulary into task labels of the outline, making sentences cloze-style prompts. A pattern with its verbalizers is called a Pattern Verbalizer Pair (PVP).

Single PVP Inference

Initially, for each pattern, a separated pre-trained language model is fine-tuned with a small amount of labelled data, T.

In the above processes, each pattern and its verbalizers are mapped together, following intra-linguistic relation.

Given some input x, the score for label $l \in \mathcal{L}$ is calculated using Equation 2.8:

$$s_p(l|x) = M(v(l)|P(x)) \tag{2.8}$$

and a probability distribution over labels is obtained using Equation 2.9 with softmax:

$$q_p(l|x) = softmax(s_p(l|x)) \tag{2.9}$$

The cross entropy between $q_p(l|x)$ and the true (one-hot) distribution of training example (x, l) as loss for fine-tuning M for p.

The loss of classification loss L_{CE} and MLM loss L_{MLM} are then added and weighted ($\alpha = 10^{-4}$) to compute total loss L_{total} as shown in Equation 2.10, in case of catastrophic forgetting under such small sample scenarios.

$$L_{total} = (1 - \alpha) \cdot L_{CE} + \alpha \cdot L_{MLM} \tag{2.10}$$

Afterwards, unlabelled dataset D can be used to perform MLM tasks with these fine-tuned models. The verbalizers along with the patterns predicted by these fine-tuned models are referred to as the soft labelled dataset with softmax for each of the unannotated instances.

Subsequently, the new dataset D with soft labels are used to train a final classifier.

Combining PVPs

However, there is no luxury of unlimited annotated data to check which PVP gives better predictions. The final results only indicate which PVPs adapt better to the data provided. In the original paper, this problem has been addressed by knowledge distillation[15]. In particular, let $p = (P, v)$ be a PVP, the unlabelled data is labelled by the models fine-tuned with different PVPs, the averaged results is calculated with the Equation 2.11:

$$s_{\mathcal{M}}(l|x) = \frac{1}{Z} \sum_{p \in P} w(p) \cdot s_p(l|x) \tag{2.11}$$

where $\mathcal{M} = \{M_p | p \in P\}$, examples $x \in D$, $Z = \sum_{p \in P} w(p)$ and the $w(p)$ are weighting terms for the PVPs. The weighting term could be averagely (uniform) or weighting them with their accuracy (weighted) in the training set, in the original paper the authors used 'weighted'. Next, the above scores are transferred into a proba-bility distribution q using softmax with a temperature $T = 2$.

The idea that has been illustrated by the authors in their paper is shown in Figure 2.11.

2.6.2 iPET

When fine-tuning ensembles of PLMs, there is no coupling between different patterns. That is, iPET is meant to make them learn different information from each other.

To achieve this, each PLM is firstly fine-tuned and generates datasets T_{new_i} with a random subset of other models and the data from D as shown in Figure 2.12 (a).

Afterwards, a new set of PET models is trained using the larger, model-specific datasets T_{new_i}, as illustrated in Figure 2.3 (b). The above process is repeated k generations, each time

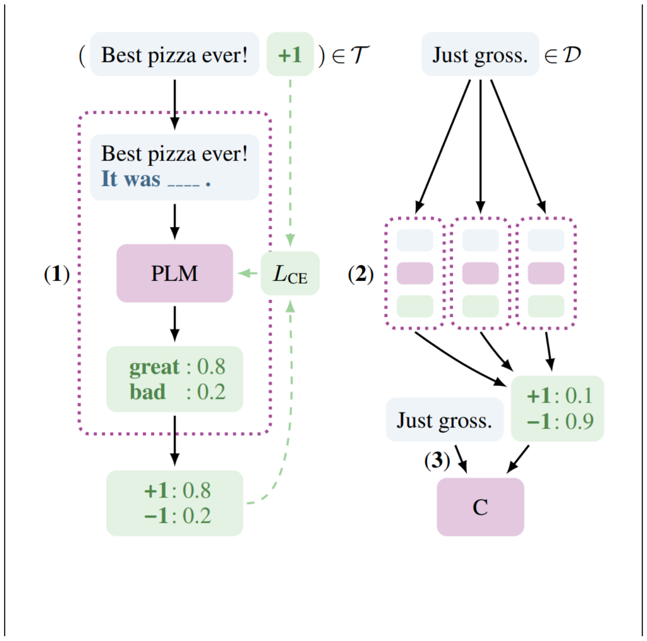
(Best pizza ever! +1) ∈ $\mathcal{T}$
Just gross. ∈ $\mathcal{D}$
Best pizza ever! It was ____ .
(1)
PLM
L_{CE}
(2)
great : 0.8
bad : 0.2
+1 : 0.1
−1 : 0.9
Just gross.
(3)
+1 : 0.8
−1 : 0.2
C

Figure 2.11: PET for sentiment classification. Source:[28]

enlarging the size of the generated training sets T_{new_i} by a factor of d, by selecting λ of all models to label examples for the next generation.

Thereafter, the amount of data for each fine-tuned PLM is increased, additionally, the information between PVP has exchanged. This has been illustrated by the authors in their paper.

2.6.3 ADAPET

PET still needs a lot of unlabeled data in the field for semi-supervised learning, whereas as a supervised method, ADAPET can achieve better results in some tasks with the same amount of labelled data but not any unlabelled data.

ADAPET computes the probability of each token as a softmax normalized across all tokens in vocabulary, instead of only across mapped verbalizers in PET. This process is called Decoupling Label Loss, by maximising the probability of the correct class tokens and minimizing the probability of incorrect class tokens (binary cross entropy), the models incline toward choosing the tokens with correct labels, as shown in Figure 2.13 (a).

In addition, ADAPET predicts the input with the label given and encourages the model to predict the masked out tokens in the input. If the label is the correct verbalizer, the model has to predict the original token, otherwise, the model is forced to not predict the original token, as illustrated in Figure 2.13 (b). By doing this, the models would learn some patterns of the downstream tasks.

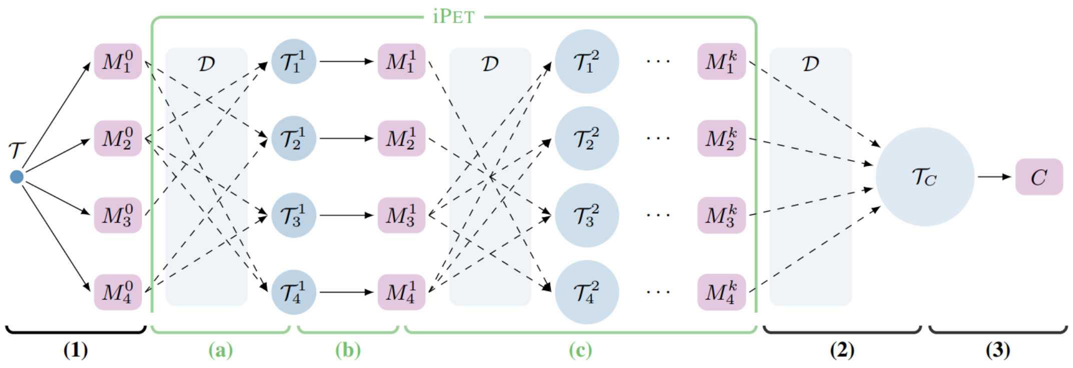

Figure 2.12: Schematic representation of iPET. Source:[28]

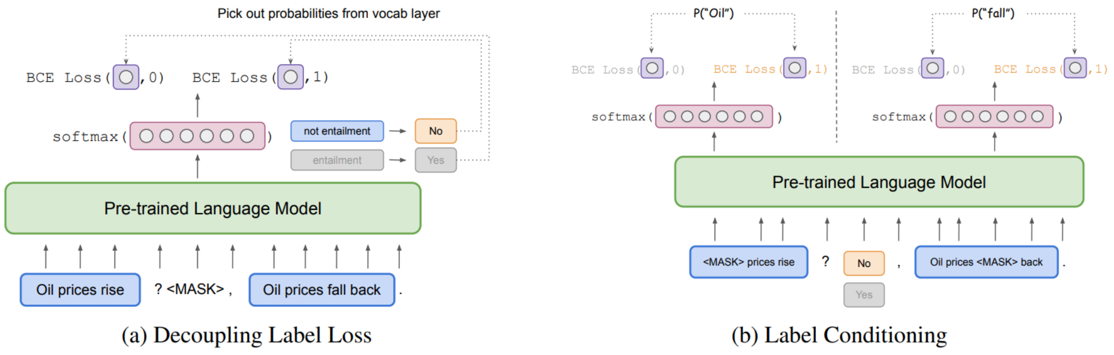

Figure 2.13: Two components of ADAPET. Source:[34]

These two techniques are applied to ADAPET models simultaneously.

2.7 Related work

Pre-training ever-larger language models on massive corpora has led to large improvements in NLP[34], for instance, BERT-large, RoBERTa, GPT-3, etc., however, enormous amounts of computational power is required for training and applying such huge models [29]. In addition, the performance of models can suffer when there is very limited labelled data available for a downstream task [38]. Therefore, language models that can work equally well and fully make use of pre-trained models with limited data are needed.

It started attracting people's attention in GPT-3, with 175 billion parameters, it can generate all kinds of text without fine-tuning, from routine tasks (such as dialogues and articles) to

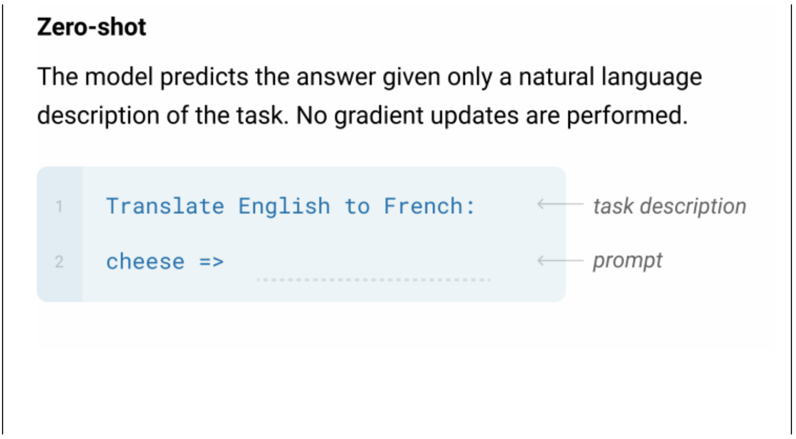
Zero-shot
The model predicts the answer given only a natural language description of the task. No gradient updates are performed.
Translate English to French:
cheese =>
task description
prompt

Figure 2.14: In-context learning of GPT-3: Zero shot learning. Source:[6]

some outlandish scenarios (generating SQL, HTML code) etc. What really catches peoples' eyes is how it works in a zero-sample scenario - based on a certain task description, generate text as specified by this description, as illustrated in Figure 2.14. (note: the prompt in the picture is not the prompt in this paper since GPT is a generative language model) However, its shortcoming is also obvious, it has too many parameters, so it is strenuous to deploy in practical applications. With the combination of this idea and MLM pre-training tasks of BERT, the very beginning concept of PET is born.

Petroni et al. (2019) measured the level of the factual and commonsense knowledge of some language models[25] with cloze-question-like prompts and showed language models indeed encode knowledge from text. They consider a language model capable of learning and storing knowledge if it is able to answer queries related to knowledge present in the training data. Subject to this, they designed the experiments for four language models, fairseq-fconv[8], Transformer-XL large, ELMo[24], ELMo 5.5B, BERT-base and BERT-large. They found some language models do learning and BERT is able to recall the knowledge better than its competitors.

Puri and Catanzaro (2019)[26] enabled zero-shot model adaptation to new tasks, by providing models with natural language descriptions of classification tasks (prompt), reformulating text classification problems as multiple choice question answering. They achieved up to a 45% absolute improvement in classification accuracy over random or majority class baselines despite no access to training data and concluded that natural language could serve as simple and powerful descriptors for task adaptation, pointing the way to new meta learning strategies for text problems.

For the most of the models, the prompt based methods are based on their MLM tasks; interestingly, there are also methods based on the NSP task of BERT. Sun et al. (2019) introduced NSP-BERT[32], based on the sentence-level pre-training task to achieve prompt-learning. They declared their NSP-BERT can be used in various forms of tasks, as

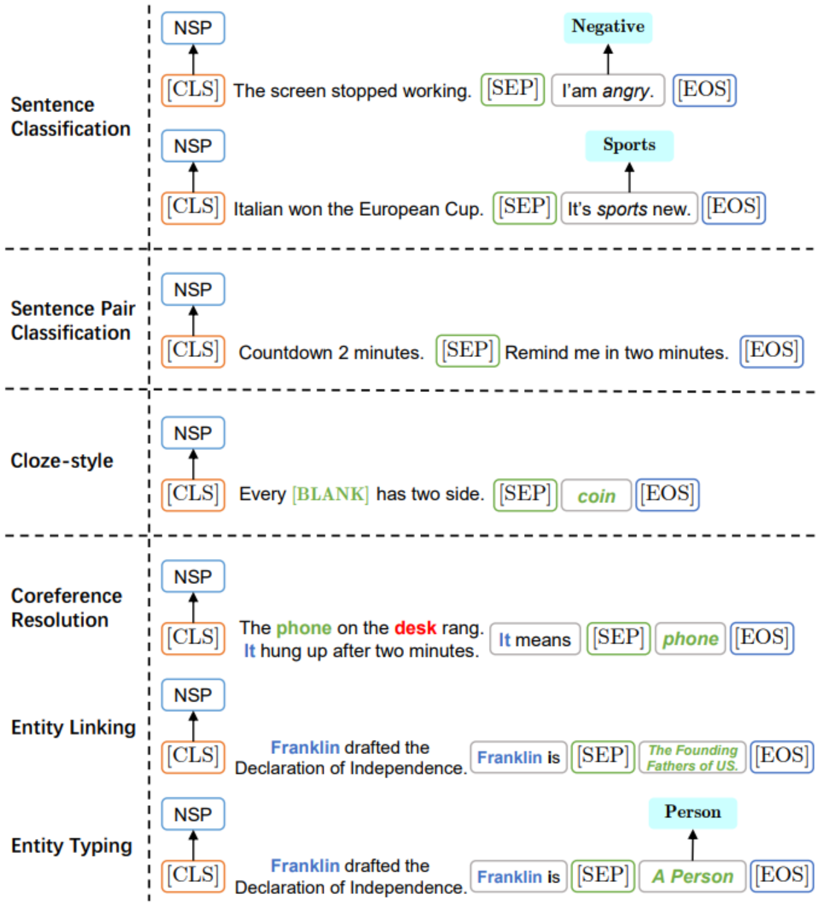
Sentence Classification
NSP
Negative
[CLS] The screen stopped working. [SEP] I'am angry. [EOS]
NSP
Sports
[CLS] Italian won the European Cup. [SEP] It's sports new. [EOS]
Sentence Pair Classification
NSP
[CLS] Countdown 2 minutes. [SEP] Remind me in two minutes. [EOS]
Cloze-style
NSP
[CLS] Every [BLANK] has two side. [SEP] coin [EOS]
Coreference Resolution
NSP
[CLS] The phone on the desk rang. It hung up after two minutes. It means [SEP] phone [EOS]
Entity Linking
NSP
[CLS] Franklin drafted the Declaration of Independence. Franklin is [SEP] The Founding Fathers of US. [EOS]
Entity Typing
NSP
Person
[CLS] Franklin drafted the Declaration of Independence. Franklin is [SEP] A Person [EOS]

Figure 2.15: Prompts for various NLP tasks of NSP-BERT. Source:[32]

illustrated in Figure 2.15.

Chapter 3

Methods and Analysis

In this chapter, four main aspects of this dissertation have been introduced. They are the details of the dataset used for training and evaluating the PVPs of PETs, and their corresponding pre-processing steps with a brief justification for the choice of preprocessing. In addition to that, the performance metrics that are used to evaluate and compare all the models and PVPs.

3.1 Dataset

The dissertation aims to reveal the difference in each method's performance, so the same dataset is used for all tasks. Idiomatic datasets like MAGPIE[14] and PIE[2] are initial candidates to be used, because their instances are mostly multiple-line, providing sufficient contextual information.

3.1.1 PIE

An idioms corpus with classes of idioms beyond the literal and the general idioms classification, containing over 20,100 samples with almost 1,200 cases of idioms from 10 classes. The classes in this dataset are metaphor, simile, euphemism, parallelism, personification, oxymoron, paradox, hyperbole, irony and literal. This corpus is built based on the British National Corpus (BNC) and UK Web Pages (UKWaC), annotated by two independent annotators.

3.1.2 MAGPIE

A high-quality corpus comprising more than 50K instances that is built using a fixed idiom list, automatic pre-extraction, and a strictly controlled crowdsourced annotation procedure, this is a large sense-annotated corpus of potentially idiomatic expressions (PIEs), based on the British National Corpus (BNC). There are 4 classes of instances in this corpus, 'idiom', 'literal', 'other' and 'unclear', the distribution of them is 70.66% 28.55% 0.77% 0.01% respectively.

3.1.3 Dataset Choice

Unfortunately, the distribution of classes in PIE is overlapped, meaning one instance can have multiple labels, which is not in the scope of this dissertation, detection of idiomaticity. Contrary to PIE, MAGPIE only contains two classes, literal and idiomatic.

3.2 Dataset Pre-processing

Firstly, all instances with labels 'other' and 'unclear' are removed. Next, sequence length larger than 499 are removed, in order to keep all information provided in an instance. This is because BERT has a max length limit of 512 tokens, by considering the max length of the PVPs to be added, instances that consist of more than 499 tokens are deleted, determined by a tokenizer of BERT. In spite of the fact that there are literatures and papers trying to address the problem of sequence length, simply removing those instances can ensure the total accuracy is not coupled with the reliability of those methods.

Afterwards, in order to keep the distribution of two classes equal, the class with more instances is partly dropped off.

Later, the dataset is split into 3 sets, training set, development (dev) set and test set, with the ratio of 80%, 10%, and 10%. In addition, the instances with each label are equally distributed in each set, in order to avoid an imbalanced classification problem, which would lead to the results being biased or skewed.

Finally, there are 19280 instances in the training set, 2412 instances in the dev set and 2412 instances in the test set. The distribution of data is illustrated in Figure 3.1 and 3.2.

3.3 Approach and setups

Designing PVPs

As stated earlier in chapter 1, the task is to experiment with PETs performances of idiom detection. Different PVPs are designed for PET, iPET and ADAPET. In order to examine the performance of the prompts with different levels of detail, the patterns and their verbalizers are created and listed in Table 3.1. The first and the third PVPs are meant to provide more details than the others, with the verbalizers put before and after the patterns.

Task 1 - BERT classifier

BERT supervised text classifier with 10, 100, 500, 1000, 2500, 5000, 7500, 10000, 12500, 15000, 17500, and 19280 labelled data, as a baseline of the experiments. The pre-trained

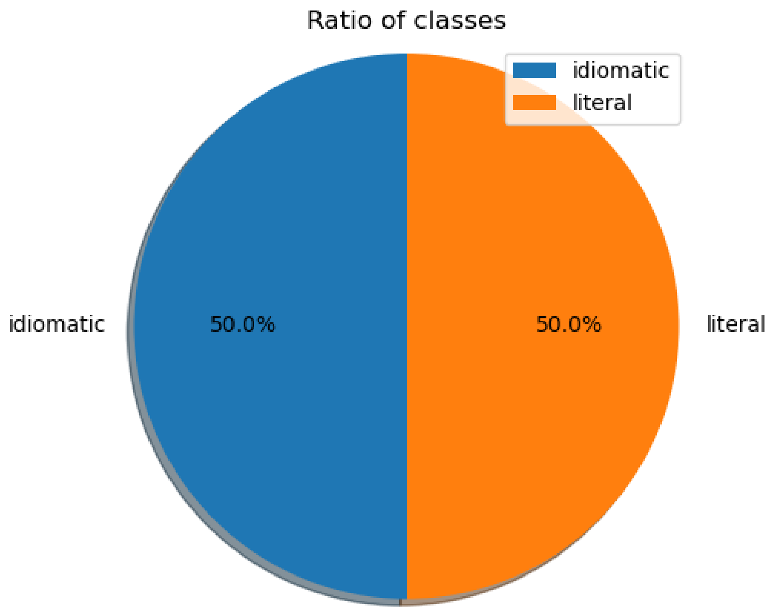

Figure 3.1: Ratio of Classes

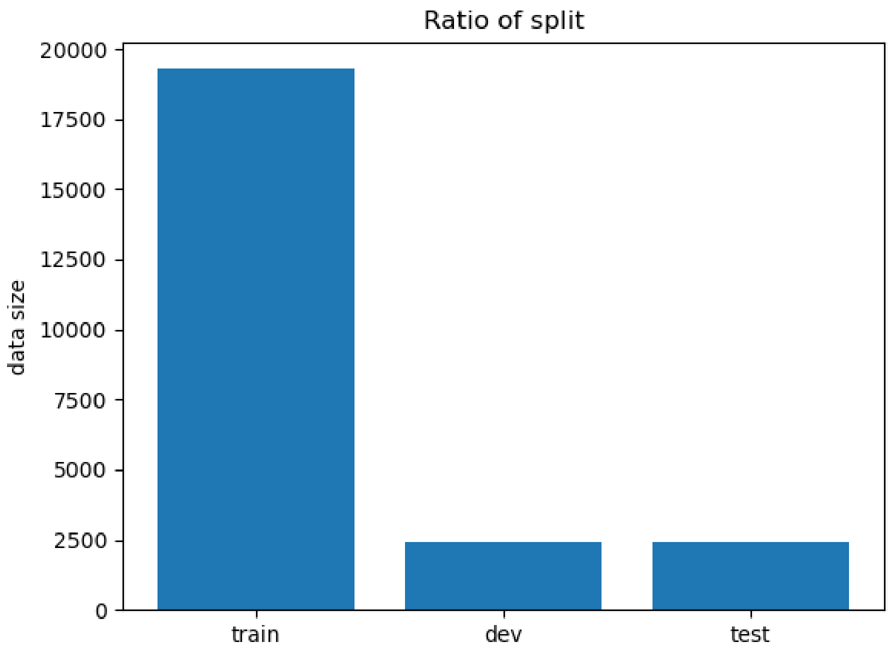

Figure 3.2: Ratio of Split

P(x)	V(0)	V(1)
x The sentence is ____.	literal	phrase
x : ____	literal	phrase
The following sentence is ____. x	literal	phrase
(____) x	literal	phrase

Table 3.1: Pattern-verbalizer pairs, where x is input sentence

language model is BERT-base-uncased.

Task 2 - PET

For each pattern, PET will be implemented with 10, 100, 1000, 2500, 5000 and 7500 labelled instances, with 10000 unlabelled instances. Additionally, PET with all patterns will be implemented with the same amount of data above. The pre-trained language model is BERT-base-uncased.

Task 3 - iPET

Following task 2, the number of generation k is set to 5, λ is set to 0.5 and the d is 2, meaning in each generation 50% of the models will be selected to label dataset D and the number of training examples is quintupled with factor $d = 2$.

Task 4 - ADAPET

ADAPET with the same amount of labelled data as task 1 and PVPs used in task 2 will be implemented for each pattern. The pre-trained weight is BERT-base-uncased.

Further hyperparameters for all experiments are given in Appendix A

3.3.1 Metrics

The objectives of the tasks are to determine which model captures more idiomatic expressions correctly. Each of them can be considered as a classification problem, to evaluate whether the expressions are used in a literal or figurative sense. Thus, the metric of F1 score (macro) in Equation 3.1 has been used to measure the performance of the models and also to compare them against one another.

$$F1 = \frac{True\_Positive}{True\_Positive + \frac{1}{2}(False\_Positive + False\_Negative)} \tag{3.1}$$

Chapter 4

Result

In this chapter, the results of previous works are shown. For each experiment, the outcomes of the models are justified and some discussions are covered at the end of each experiment.

4.1 BERT classifier - as baseline

It can be observed that as the number of the annotated data used to train the classifier, the F1 score of the classifier increases, in Table 4.1. They showed a positive correlation, this increasing trend stopped when the amount of labelled data is between 5000 and 15000, and then after that, the performance started going up again. The results in the table are rounded to the second decimal place.

4.2 PET

In Figure 4.1, for PET+p_0, PET+p_1, PET+p_2, and PET+p_3, the F1 scores of them are also proportional to the annotated data used, and they all outperformed BERT classifier in Table 4.1. This increasing trend generally plateaued when the labelled data reached 2500. The average performances of different patterns are close, only several percent of difference is drawn. Different from the expectation, the patterns with details could not provide higher performance, but this could arise from the poor design of patterns. For PET+all, there are noteworthy improvements across different amounts of data, compared to PETs with one PVP (sPET). This could be the result of some bad inferences of models being weighted less during the process of distillation.

4.3 iPET

The indigent performance across low data amount in Table 4.1 could suggest that the deficient models at the beginning of the generations fail to provide labelled data with good quality to their offspring models when the amount of data is below 500. The results in the table are rounded to the second decimal place. However, beyond that, the iPET with all PVPs shows significant improvements when the amount of labelled data reached 500. This could be concluded that if there are enough labelled data to guide the starting generations, this method could have promising potential for such tasks.

4.4 ADAPET

ADAPET generally showed surprisingly better performance than sPET and PET with all PVPs, and overall slightly higher F1 score than iPET when the amount of labelled data is less than 500, with zero unlabelled data, as illustrated in Table 4.2. The performance of ADAPET models also increases when with a higher number of instances. In addition, it could also be spotted that the models with different PVPs have similar performance, as also illustrated in Figure 4.1.

4.5 Discussions

PET, iPET and ADAPET showed a significant improvement in the performance of detecting the existence of idiomaticity in MAGPIE instances, compared to the BERT classifier, in all tasks in the experiments. This could suggest using MLM tasks in BERT actually boost the performance of text binary classification tasks.

Generally, ADAPET models have averagely better accuracy than sPET models and PET with all PVPs with even no labelled data, this could be due to the better strategy of selecting correct labels and updating the losses of ADAPET.

Additionally, the iPET models presented lower performance than other PETs when the number of annotated data is lower than 500. Such an amount of data seems to be a watershed, leading to a worse or a better F1 score, nevertheless, with such small data, it still outmatched the BERT classifier. Apart from it, iPET showed the best performance among all prompt based methods on account of its mechanism of pattern info exchange in generations.

In terms of the patterns, they did not result in overwhelming different performances when prompts with different levels of information details were provided. This could be due to the poor design of unclear and ambiguous patterns and verbalizers, the language model cannot discriminate them in a clear way.

The results of prompt based methods are as expected, since fine-tuning BERT does not require training a new network, and in the pre-training phase of BERT, the model itself should already have learnt the ability of inference, and some relations between words among phrases. Different from the expectations, the PVPs with different info rarely affect BERT to get distinct performances, it seems that they failed to provide clear-cut details, but they still played the role of guiding the language models.

annotated data	BERT	PET+all	PET+p_0	PET+p_1	PET+p_2	PET+p_3	iPET+all
10	24.05%	53.86%	51.32%	53.12%	53.85%	50.91%	44.83%
100	29.60%	61.50%	60.04%	58.17%	59.87%	58.24%	53.64%
500	33.14%	67.37%	61.79%	67.19%	63.27%	65.27%	69.20%
1000	40.80%	70.43%	68.01%	69.44%	68.72%	68.51%	72.34%
2500	52.43%	72.68%	71.24%	71.92%	71.10%	71.23%	75.74%
5000	62.85%	72.80%	71.54%	71.80%	70.85%	72.38%	82.56%
7500	62.94%	72.13%	70.16%	70.84%	71.50%	70.37%	85.09%
mean	43.69%	67.25%	64.87%	66.07%	65.59%	65.27%	69.06%

Table 4.1: BERT Classifier vs PET & iPET

annotated data	BERT	ADAPET+p_0	ADAPET+p_1	ADAPET+p_2	ADAPET+p_3
10	24.05%	38.84%	35.40%	36.52%	49.10%
100	29.60%	48.72%	35.40%	49.57%	52.08%
500	33.14%	65.18%	64.56%	66.84%	68.38%
1000	40.80%	69.64%	69.61%	69.45%	68.56%
2500	52.43%	74.07%	73.84%	75.06%	75.89%
5000	62.85%	77.98%	79.15%	80.34%	80.68%
7500	62.94%	83.42%	83.75%	84.95%	84.66%
10000	62.52%	84.94%	85.77%	86.24%	86.56%
15000	63.02%	87.02%	87.98%	87.77%	88.39%
19280	69.26%	88.47%	89.18%	88.97%	88.51%
mean	43.69%	71.83%	70.46%	72.57%	74.28%

Table 4.2: BERT Classifier vs ADAPET

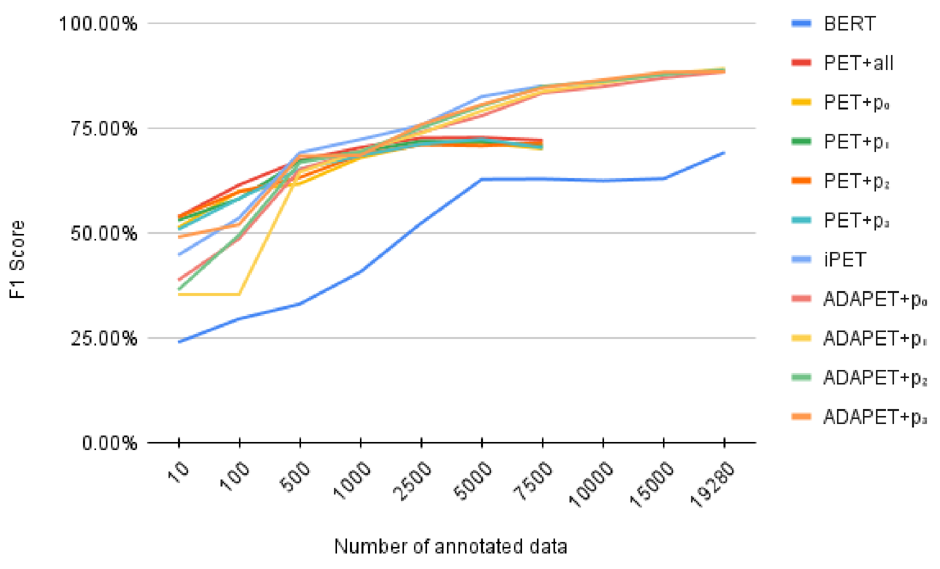

Figure 4.1: BERT Classifier vs PET & iPET

Chapter 5

Conclusions and Future work

In this chapter, all of the conducted inferences of the experiments are summarised in a concise manner. In addition, additional works and improvements that can be done in the future along the line of the current dissertation have been listed at the end of this chapter.

5.1 Conclusion

In this paper, the aim was to examine the performance of the prompt based learning methods, and how different prompts affect their results. The experiments of PET, iPET, and ADAPET were conducted and compared to the baseline, BERT classifier, in binary classification tasks with MAGPIE dataset. It can be seen from the results that the knowledge of the language model, BERT-base-uncased, indeed provides uplift in these tasks, leading remarkable superior performances compared to the BERT classifier. It was also found different prompts rarely affect how language models modelling, the problem of this needs to be investigated on a deeper level.

Intuitively, speaking of binary text classification, people think of the BERT classifier. This paper showed it is absolutely worth trying and doable using prompts based learning methods, which yield a better accuracy in this paper, and choose a better method among them based on the available data, to stand on the shoulders of giants efficiently and accurately, if sufficient time is allowed for training.

However, it is really hard to draw a clear conclusion with such results since there are still improvements that can be done.

5.2 Improvement and Future work

Firstly, the most problematic issue could be the design of the patterns, some of them tried to provide more info but the results showed only a slight difference. An ideal pattern could also include the sequence of MWE, additional particular, or some more direct and understandable words.

Secondly, if the models were trained with data in a smaller scope, that is, only focusing on multi word expressions on a specific pattern, such as noun compounds, it is believed that the conducted results will be more persuasive.

Thirdly, the results of experiments were conducted only one time, the experiments should be repeated multiple times and the conclusions of the experiments should be based on these trials.

Fourthly, some of the experiments were conducted with different amounts of GPUs, this could cause some variations while calculating gradient descents of the models in the training phase. In spite of the difference could be very small, it is still needed to be noted since the results are also very close.

In addition, even though the dataset looked relatively large, the performance of the BERT classifier and some of the other models are likely still increasing. Unfortunately, there is not enough data to test the limit of these models. How much and how fast the models could these methods gain the knowledge from the data would be definitely worth examining out.

Theoretically, these prompt based methods could also be applied to multi classification tasks, iPET could also take no labelled data, and ADAPET could also introduce generations. It would also be interesting to implement them, to see how much better performance could be reached.

Bibliography

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (2016), pp. 265–283.

[2] Adewumi, T. P., Javed, S., Vadoodi, R., Tripathy, A., Nikolaidou, K., Liwicki, F., and Liwicki, M. Potential idiomatic expression (pie)-english: Corpus for classes of idioms. *arXiv preprint arXiv:2105.03280* (2021).

[3] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[4] Basiri, M. E., Nemati, S., Abdar, M., Cambria, E., and Acharya, U. R. Abcdm: An attention-based bidirectional cnn-rnn deep model for sentiment analysis. *Future Generation Computer Systems 115* (2021), 279–294.

[5] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *the Journal of machine Learning research 3* (2003), 993–1022.

[6] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems 33* (2020), 1877–1901.

[7] Cilibrasi, R. L., and Vitanyi, P. M. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering 19*, 3 (2007), 370–383.

[8] Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning* (06–11 Aug 2017), D. Precup and Y. W. Teh, Eds., vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 933–941.

[9] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Diab, M., and Bhutada, P. Verb noun construction mwe token classification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications (MWE 2009)* (2009), pp. 17–22.

[11] Doogan, S., Ghosh, A., Chen, H., and Veale, T. Idiom savant at semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)* (2017), pp. 103–108.

[12] d'Arcais, G. B. F., and Giovanni, B. The comprehension and semantic interpretation of idioms. *Idioms: Processing, structure, and interpretation* (1993), 79–98.

[13] Ghorpade, J., Parande, J., Kulkarni, M., and Bawaskar, A. Gpgpu processing in cuda architecture. *arXiv preprint arXiv:1202.4347* (2012).

[14] Haagsma, H., Bos, J., and Nissim, M. MAGPIE: A large corpus of potentially idiomatic expressions. In *Proceedings of the 12th Language Resources and Evaluation Conference* (Marseille, France, May 2020), European Language Resources Association, pp. 279–287.

[15] Hinton, G., Vinyals, O., Dean, J., et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531 2*, 7 (2015).

[16] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[17] Liu, C., and Hwa, R. A generalized idiom usage recognition model based on semantic compatibility. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 6738–6745.

[18] Liu, P., Qian, K., Qiu, X., and Huang, X.-J. Idiom-aware compositional distributed semantics. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (2017), pp. 1204–1213.

[19] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[20] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[21] Odijk, J. Identification and lexical representation of multiword expressions. In *Essential speech and language technology for Dutch*. Springer, Berlin, Heidelberg, 2013, pp. 201–217.

[22] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. nips autodiff workshop. In *Website https://openreview. net/forum* (2017).

[23] Peng, J., and Feldman, A. Experiments in idiom recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (2016), pp. 2752–2761.

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

[25] Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066* (2019).

[26] Puri, R., and Catanzaro, B. Zero-shot text classification with generative language models. *arXiv preprint arXiv:1912.10165* (2019).

[27] Sag, I. A., Baldwin, T., Bond, F., Copestake, A., and Flickinger, D. Multiword expressions: A pain in the neck for nlp. In *International conference on intelligent text processing and computational linguistics* (2002), Springer, pp. 1–15.

[28] Schick, T., and Schütze, H. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676* (2020).

[29] Schick, T., and Schütze, H. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118* (2020).

[30] Shwartz, V., and Dagan, I. Still a pain in the neck: Evaluating text representations on lexical composition. *Transactions of the Association for Computational Linguistics 7* (2019), 403–419.

[31] Sporleder, C., and Li, L. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)* (2009), pp. 754–762.

[32] Sun, Y., Zheng, Y., Hao, C., and Qiu, H. Nsp-bert: A prompt-based zero-shot learner through an original pre-training task–next sentence prediction. *arXiv preprint arXiv:2109.03564* (2021).

[33] Tai, K. S., Socher, R., and Manning, C. D. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

[34] Tam, D., Menon, R. R., Bansal, M., Srivastava, S., and Raffel, C. Improving and simplifying pattern exploiting training. *arXiv preprint arXiv:2103.11955* (2021).

[35] Trinh, T. H., and Le, Q. V. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847* (2018).

[36] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[37] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations* (2020), pp. 38–45.

[38] Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848* (2019).

[39] Yamamoto, K., Kudo, T., Konagaya, A., and Matsumoto, Y. Protein name tagging for biomedical annotation in text. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine* (2003), pp. 65–72.

Appendices

Appendix A

Training Details

A.1 Implementation

A.1.1 PET, iPET and ADAPET

The implementation of PET, iPET and ADAPET is based on the Transformers library[37] (Wolf et al., 2020) and PyTorch[22] (Paszke et al., 2017). The implementation of BERT classifier is based on TensorFlow library[1] (Abadi et al., 2016). The CUDA[13] toolkit (Ghorpade et al., 2012) is used in the implementation.

A.2 Hyperparameters

A.2.1 BERT Classifier

The learning rate of the classifier is set to $\alpha = 1e - 5$, the choice of the optimizer is Adam, with the epsilon $\epsilon = 1e - 8$. The number of train, dev and test batch sizes are set to 2, 4 and 2. The epoch of training phase is 3.

A.2.2 PET and iPET

Same as in the BERT classifier, the learning rate of the classifier is set to $\alpha = 1e - 5$, the optimizer is Adam, with the epsilon set to $\epsilon = 1e - 8$. The PET repetition is 1. For PET, iPET and their final classifiers, the train batch size is 2, the unlabelled batch size is 4, with an accumulation step of 8. The temperature of final distillation is 2 (Hinton et al., 2015)[15]. The reduction strategy for merging predictions from multiple PET models is wmean, weighting with their accuracy. The epoch of the final classifiers and PET is 3.

A.2.3 ADAPET

The learning rate again is set to $\alpha = 1e-5$. The train batch size is 2 and the evaluation (dev) batch size is 4, with the accumulation factor of 8, evaluating every 2 steps. Since ADAPET takes number of batches as a hyperparameter so for tasks with different amount of train data, it is calculated using the Equation A.1:

$$batch\_number = \frac{amount\_of\_data}{grad\_accumulation\_factor \times batch\_size} \times epoch \tag{A.1}$$

A.3 Hardware

The PET (excluding iPET) and ADAPET (excluding ADAPET with 10000, 15000, 19280 labelled examples) experiments were run on ShARC (Sheffield Advanced Research Computer), with a single NVIDIA Tesla K80 (only 12GB onboard VRAM) GPU. ADAPET with 10000 and 15000 labelled examples were also run on ShARC but with 2 GPUs. The iPET experiments were run on Bessemer, the University of Sheffield's newest High Performance Computing cluster, with a single NVIDIA Tesla V100 (32GB VRAM) GPU, ADAPET tasks with 19280 labelled data were also trained on Bessemer but with 4 GPU at a time. The BERT Classifier experiments were conducted using a personal computer, equipped with a single Gigabyte GeForce RTX 2080 SUPER GAMING OC (8GB VRAM) GPU.

A.4 Training Time

All BERT Classifier training only took 15 hours on the personal computer. Each sPET task took approximately 8 hours to finish running and each PET-all task took approximately 12 hours to finish running. For iPET, each of them took 1 to 5 days to finish running on Bessemer, depending on the amount of their labelled data. For ADAPET tasks, each of them took from less than 2 hours to 5 days on ShARC, depending on the amount of their data used for training, apart from ADAPETs with 4 GPUs only taking approximately 1 day to finish training.