applied sciences

MDPI

Article

High-Performance English–Chinese Machine Translation Based on GPU-Enabled Deep Neural Networks with Domain Corpus

Lanxin Zhao [1,*], Wanrong Gao [2] and Jianbin Fang [2]

[1] School of International Business, Hunan University of Information Technology, Changsha 410151, China
[2] School of Computer, National University of Defense Technology, Changsha 410073, China; gaowanrong@nudt.edu.cn (W.G.); j.fang@nudt.edu.cn (J.F.)
[*] Correspondence: zhaolanxin21@yeah.net

Abstract: The ability to automate machine translation has various applications in international commerce, medicine, travel, education, and text digitization. Due to the different grammar and lack of clear word boundaries in Chinese, it is challenging to conduct translation from word-based languages (e.g., English) to Chinese. This article has implemented a GPU-enabled deep learning machine translation system based on a domain-specific corpus. Our system takes an English text as input and uses an encoder-decoder model with an attention mechanism based on Google's Transformer to translate the text to Chinese output. The model was trained using a simple self-designed entropy loss function and an Adam optimizer on English–Chinese bilingual text sentences from the News area of the UM-Corpus. The parallel training process of our model can be performed on common laptops, desktops, and servers with one or more GPUs. At training time, we not only track loss over training epochs but also measure the quality of our model's translations with the BLEU score. We also provide an easy-to-use web interface for users so as to manage corpus, training projects, and trained models. The experimental results show that we can achieve a maximum BLEU score of 29.2. We can further improve this score by tuning other hyperparameters. The GPU-enabled model training runs over 15x faster than on a multi-core CPU, which facilitates us having a shorter turn-around time. As a case study, we compare the performance of our model to that of Baidu's, which shows that our model can compete with the industry-level translation system. We argue that our deep-learning-based translation system is particularly suitable for teaching purposes and small/medium-sized enterprises.

Keywords: neural machine translation; transformer; GPUs; multi-domain corpus

check for updates

Citation: Zhao, L.; Gao, W.; Fang, J. High-Performance English–Chinese Machine Translation Based on GPU-Enabled Deep Neural Networks with Domain Corpus. *Appl. Sci.* **2021**, *11*, 10915. https://doi.org/10.3390/app112210915

Academic Editors: Jorge Martin-Gutierrez and João M. F. Rodrigues

Received: 13 September 2021
Accepted: 17 November 2021
Published: 18 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Currently, machine learning (ML) is experiencing a renaissance where deep learning (DL) has been the main driving force. Deep neural networks (DNNs) are extremely powerful machine learning models that can achieve promising performance on challenging problems such as speech recognition [1,2] and visual object recognition [3–6]. In particular, due to the capacity of capturing complex linguistic structures, DNNs have enabled great breakthroughs in natural language processing (NLP) [7–9]. Among the NLP tasks, machine translation (MT) is a successful representative, and its main task is using computer software to translate text or speech from one language to another.

It is a common belief that machine translation has experienced three major development waves: rule-based machine translation (RMT) [10], statistical machine translation (SMT) [11], and neural machine translation (NMT) [12]. SMT has been the mainstream driving force during the past two decades. However, this approach may ignore the long dependency beyond the length of phrases and thus cause inconsistencies in translation results such as incorrect gender agreements. It also suffers in separate components such as word aligners, translation rule extractors, and other feature extractors. Compared with the SMT method, NMT has a simple architecture, and it is able to capture long dependency in the sentence, which shows a great potential in becoming a new trend of language translation.

The dominant NMT approach is the "Embed-Encode-Attend-Decode" paradigm. Recurrent neural network (RNN) [13], convolutional neural network (CNN) [14], and self-attention/feed-forward network (SA/FFN) [15] architectures are the most commonly used approaches based on this paradigm. In particular, Google has proposed Transformer, which relies entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or CNNs [15]. The Transformer model aims to deal with long-range dependencies when solving the sequence-to-sequence tasks. This model has outperformed many other models, which is thus the focus of our work.

Recent works have focused on large pretrained models, which are mostly built based on the Transformer model, but with much larger capacity [15–18]. This approach utilizes a combination of pretraining and supervised fine-tuning. The capacity of transformer language models has increased significantly, from 100 million parameters [16], to 1.5 billion parameters [17], and finally 17 billion parameters [9,18]. Although each increase has brought significant performance improvements in downstream NLP tasks, training such models requires large-scale specialized computing hardware such as Google's TPUs [19]. These computing clusters are typically unaffordable for small/medium-sized enterprises. On the other hand, these models are too complicated and their capacity is too large for us to understand. That is, we know the models perform well, but we do not know the reasons. They work similar to a "black-box" and are particularly unsuitable for teaching purposes.

The dominant approach to creating machine learning systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent held-out examples. This approach can provide trained models that work like domain experts, which has been widely accepted. Many prior works have shown that trained models can yield a better prediction accuracy from using a domain-specific bilingual corpus [20–22]. For example, microblogs are an excellent linguistic resource. Ling et al. have shown that some microblog users post "self-translated" messages targeting audiences who speak different languages [21]. Based on this observation, the authors have introduced a method for finding and extracting this naturally occurring Chinese–English parallel segments. Their evaluation results have demonstrated that the automatically extracted parallel data obtain significant translation quality improvements. Tian et al. have designed UM-Corpus as a multi-domain and balanced parallel corpus [23]. It is a two million English–Chinese aligned corpus from eight different text domains, including Education, Laws, Microblog, News, Science, Spoken, Subtitles, and Thesis. Although using a domain-specific corpus for model training has yielded promising results, we believe that there is a lack of a bilingual corpus from domain experts. Therefore, determining how to leverage domain expertise and build a new corpus is largely required.

In this work, we present an easy-to-use deep learning machine translation system, built from the scratch, based on a domain corpus. The deep learning algorithm takes in English text as input and uses an encoder-decoder model with an attention mechanism based on Google's Transformer to translate the text to Chinese output. The model was trained using a simple self-designed entropy loss function and an Adam optimizer on paired English and Chinese text sentences from the news area of the UM-Corpus. The parallel training process of our model can be performed on common laptops, desktops, and servers with one or more GPUs. During training time, we not only track loss over training epochs but also measure the quality of our model's translations using the BLEU score (see Section 2.1.5).

The experimental results on the UM-corpus show that our trained model can achieve a maximum BLEU score of 29.2. We can further improve this score by tuning other hyper-parameters. We provide a web interface for users to build a domain-specific corpus and configure training parameters. We also observe that training the model on high-end GPUs is much faster than on a multi-core CPU, and thus the GPU is a very promising training platform for NMT. We conclude that our translation system is platform-portable, which is suitable for teaching purposes and use scenarios in small/medium-sized enterprises. As a case study, we compare the performance of our model to that of Baidu's and show that our model can compete with the production-level translation system.

To summarize, our contributions are as follows:

- We present a transformer-based machine translation system, which is built from scratch, based on a domain corpus.
- Our translation system is easy to use with a web interface, so that domain experts can extend the existing corpus and train the model in a fine-tuned manner.
- Our machine learning system is both portable and configurable and can be deployed on laptops, desktops, or servers with multiple GPUs.
- Our translation system trained based on a domain corpus can achieve competing performance with a production-level translation system.

2. Background and Related Work

This section introduces the background of neural machine translation (NMT) and emphasizes prior works on the English–Chinese machine translation.

2.1. Neural Machine Translation

Due to the proliferation of deep learning, using deep neural networks for machine translation tasks has gained great attention. We regard that Kalchbrenner and Blunsom proposed the first successful DNN-based machine translation model, which is a new concept for machine translation [24]. Compared with other models, the NMT model needs less linguistic knowledge while producing a competitive performance. Since then, many researchers have shown that NMT can perform much better than SMT.

2.1.1. Formulating the NMT Task

In the MT task, the language model (LM) can actually give the most important information: the emergence probability of a particular word (or phrase) that is conditioned on previous words. Thus, the key to improve the translation performance is to build a better language model. The NMT task is designed as an end-to-end learning task. It directly processes a source sequence to a target sequence. The learning objective is to find the correct target sequence given the source sequence, which can be seen as a high dimensional classification problem that tries to map the two sentences in the semantic space.

Given a parallel corpus C having a set of parallel sentence pairs (x, y), the training objective is to maximize the likelihood L in terms of θ, which is shown in Equation (1) L:

$$L_\theta = \sum_{(x,y)\in C} \log_p(y|x;\theta), \tag{1}$$

where $x = x_1, \ldots, x_n$ denotes an input sentence, $y = y_1, \ldots, y_n$ represents its translation, and θ is a set of parameters to be learned. Given the source sentence, the probability of a target sentence is calculated as shown in Equation (2):

$$p(y|x;\theta) = \prod_{j=1}^{m} p(y_j|y_{<j}, x;\theta) \tag{2}$$

where m is the number of words in y, y_j is the current generated word, and $y_{<j}$ are the previously generated words. At the inference time, beam search is typically used to find the translation that maximizes this probability.

2.1.2. The NMT Structure

The most commonly used NMT approach is the "Embed-Encode-Attend-Decode" paradigm, which is illustrated in Figure 1. When the encoder receives one source sentence, it reads the source sentence word by word and compresses the variable-length sequence into a fixed-length vector. This process is encoding, i.e., the encoder converts words in the source sentence into word embedding. These word embeddings are then processed by neural layers and converted to representations that capture contextual information. These contextual representations are called the encoder representations. The decoder uses an attention mechanism, the encoder representations, and previously generated words to generate the decoder representations, which in turn are used to generate the next target word. The encoder and decoder can be of RNN [13], CNN [14], or self-attention and feed-forward [15].

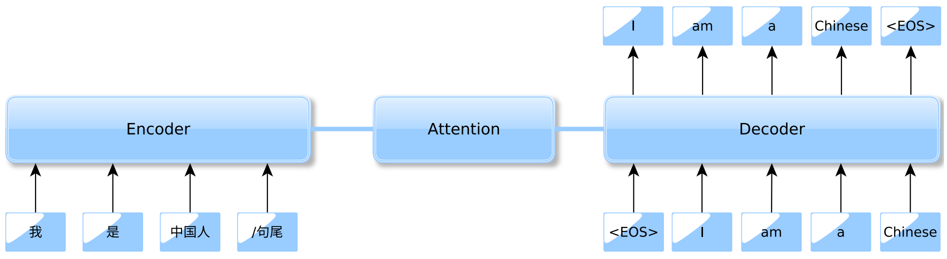


Figure 1. A standard NMT model based on the encode-attend-decode modeling approach.

While NMT has shown great potential in capturing the dependencies inside the sequence, it still suffers a huge performance reduction when the input sentences are too long. This is due to the limited feature representation ability in a fixed-length vector. Thus, the attention mechanism came into being. It works as an intermediate component between Encoder and Decoder, which facilitates the word correlation in a dynamic manner (Figure 1). As a matter of fact, the inspiration for applying the attention mechanism on NMT comes from human behavior in reading and translating text data: human beings often read text repeatedly to mine the word dependency within the sentence.

2.1.3. NMT with Attention Mechanism

Recently, fully attention-based NMT has shown promising performance. In particular, the attention mechanism has worked as a driving force in text feature extraction rather than having an auxiliary role. Among them, Transformer is one representative, which is a fully attention-based model from Google [15].

Different from prior RNN- or CNN-based models, Transformer is a complete attention-based NMT model. That is, it is of self-attention with a feed-forward connection, which can be a feature extractor allowing the entire sentence to be "read" and modeled once. It is a common practice to stack multiple layers, which leads to an improved translation quality.

Formulating Self-Attention Layers. The attention mechanism is calculated across the decoder and encoder in Equations (3) and (4):

$$e^{ji} = a(s_{j-1}, h_i), \tag{3}$$

$$a_{ji} = \frac{\exp(e_{ji})}{\sum_{k=1}^{m} \exp(e_{ki})}, \tag{4}$$

where e_{ji} is an alignment score, a is an alignment model that scores the match level of the inputs around position i and the output at position j, $s_{(j-1)}$ is the decoder hidden state of the previously generated word, and h_i is the encoder hidden state at position i. The calculated attention vector is then used to weight the encoder hidden states to obtain a context vector as

$$c_j = \sum_{i=1}^{n} a_{ji} h_i, \tag{5}$$

This context vector, is fed to the decoder along with the previously generated word and its hidden state to produce a representation for generating the current word. A decoder hidden state for the current word s_j is computed by

$$s_j = g(s_{j-1}, y_{j-1}, c_j), \tag{6}$$

where g is an activation decoder function, $s_{(j-1)}$ is the previous decoder hidden state, and $y_{(j-1)}$ is the embedding of the previous word. The current decoder hidden state s_j, the previous word embedding, and the context vector are fed to a feed-forward layer f and a softmax layer to compute a score for generating a target word as output:

$$P(y_j | y_{<j}, x) = softmax(f(s_j, y_{j-1}, c_j)), \tag{7}$$

2.1.4. NMT Model Training

When training an NMT model, the first step is to transfer the words to vectors, i.e., word embedding. The most frequently used words in one language will be chosen, and the remaining words are treated as unknown words. To overcome the problem of unknown words, the most common practice is subword tokenization with methods such as byte-pair encoding (BPE) [25], word-piece model (WPM) [26], or sentence-piece model (SPM) [27].

During the training time, the encoder-decoder model is fed by a parallel corpus. The learning objective is to minimize the cross-entropy loss between the predicted target words and the actual target words in the reference. The model parameters are initialized randomly. The training process could be formulated as updating its parameters periodically until obtaining the minimum loss of the neural network. This loss minimization is an optimization problem, and we can use gradient descent methods such as SGD, Adam, ADAGRAD, and Adafactor [28]. Among them, Adam is able to train models very fast, but it suffers in converge speed. In contrast, SGD can converge better, but it requires a long time for training. Designing a learning schedule that combines several optimizers can help train a model efficiently [29].

Training is done for a large number of iterations till the model converges. That is, the model evaluation does not change by a significant amount over iterations. In the implementation, we will refine the parameters after it processes a batch of training samples. We have to take care of the hyperparameter tuning, including learning rate, number of layers, and so on.

2.1.5. Model Evaluation

It is common to use bilingual evaluation understudy (BLEU) to evaluate NMT tasks. This metric is used to measure the differences between a model generated target sentence and its reference sentence. BLEU is defined in Equations (8) and (9):

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n), \tag{8}$$

$$BP = \{ \begin{matrix} 1, c > r \\ e^{1-\frac{r}{c}}, c \le r \end{matrix} \tag{9}$$

where p_n is n-gram corrected accuracy, w_n is $\frac{1}{n}$, c is the length of the translated sentence, r is the length of the reference sentence, and $N = 4$. The larger the BLEU is, the better.

2.2. *English–Chinese Machine Translation*

English–Chinese machine translation has been investigated for several decades. We summarize the prior NMT work in terms of designing new learning models and leveraging language features.

2.2.1. Designing New Learning Models

Hassan et al. address the problem of how to define and accurately measure human parity in translation and describe Microsoft's machine translation system [30]. They see that the translation quality of the latest neural machine translation system is at human parity. To address the issue of duplicate or missing translations, Lin et al. proposed neural machine translation improvements based on a novel beam search evaluation function [31]. They show that the proposed methods can effectively improve the English to Chinese translation quality.

SMT often performs better than NMT in translation adequacy and word coverage. Thus, it is a promising direction to combine the advantages of NMT and SMT. Zhou et al. propose a deep neural network-based system combination framework leveraging both minimum Bayes-risk decoding and multi-source NMT, which take as input the N-best outputs of NMT and SMT systems and produce the final translation [32]. This approach has been shown to significantly outperform the conventional system combination methods.

Xiong et al. propose to enhance encoding components with different levels of composition [33]. This model takes (1) the original word embedding for raw encoding with no composition and (2) a particular design of external memory in a neural turing machine (NTM) for more complex compositions. An empirical study on Chinese–English translation shows that their model can improve by 6.52 BLEU points. Wang et al. describe the Sogou neural machine translation systems for the WMT 2017 Chinese–English news translation tasks [34]. Their translation systems are built based on a multi-layer encoder-decoder architecture with attention mechanism. The best translation is obtained with ensemble and reranking techniques. Their translation system achieved the highest BLEU among all 20 submitted systems.

Tencent neural machine translation systems were designed for the WMT 2020 news translation tasks [35]. Their systems are built on deep Transformer and several data augmentation methods. They propose a boosted in-domain finetuning method to improve single models. They achieved a BLEU score of 36.8 on the Chinese–English task. In 2021, Tencent introduced a system based on the Transformer with several novel and effective variants. Their constrained systems achieve very good BLEU scores.

2.2.2. Leveraging Chinese Features

Chinese phonologic features play an important role in the sentence pronunciation. To improve the machine translation performance, Yang et al. propose a novel phonology-aware neural machine translation (PA-NMT) model where Chinese phonologic features are leveraged for translation tasks with Chinese as the target [36]. A separate recurrent neural network (RNN) is constructed in the NMT framework to exploit Chinese phonologic features to facilitate the generation of more native Chinese expressions. Experimental results on the English-to-Chinese task show that the proposed method significantly outperforms state-of-the-art baselines.

Neural machine translation (NMT) faces the challenge of out-of-vocabulary (OOV) word translation. Han et al. address this OOV issue and improve the NMT adequacy with a harder language, such as Chinese, whose characters are even more sophisticated in composition [37]. They integrate the Chinese radicals into the NMT model with different settings to address the unseen word challenge in Chinese-to-English translation. The experiments on standard Chinese-to-English NIST translation shared task data from 2006 and 2008 show that their designed models outperform the baseline model in a wide range of state-of-the-art evaluation metrics.

3. Our Methods

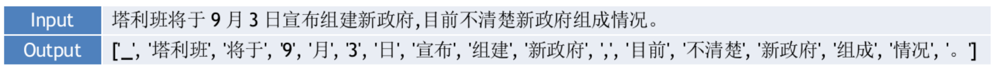
This section provides a detailed description of our methods for English–Chinese translation based on the Transformer model. We introduce our methods in terms of word segmentation, data preprocessing, model training, and deployment.

3.1. Word Segmentation

The task of word segmentation is to divide a string of written language into its component words. In English, the space is a good approximation of a word delimiter. However, the equivalent to word spacing is missing in languages such as Chinese.

Here, we use the BPE (byte pair encoding) algorithm to perform tokenization on the raw dataset [25]. Specifically, we use SentencePiece, which is an unsupervised text tokenizer for neural network-based text generation systems, where the vocabulary size is predetermined prior to the neural model training. SentencePiece allows us to make an end-to-end system that does not depend on language-specific pre-/post-processing.

For English, the segmenter splits the punctuation and separates some affixes such as possessives. For Chinese, which is written without spaces between words, SentencePiece treats the input text as a sequence of Unicode characters. Whitespace is also handled as a normal symbol. Thus, we can detokenize the text without any ambiguities. Based on this tool, we can perform word segmentation efficiently, which is shown in Figure 2.



Input	塔利班将于 9 月 3 日宣布组建新政府,目前不清楚新政府组成情况。
Output	['_', '塔利班', '将于', '9', '月', '3', '日', '宣布', '组建', '新政府', ',', '目前', '不清楚', '新政府', '组成', '情况', '。']

Figure 2. Tokenizing a Chinese sentence based on SentencePiece.

Each sentence is transformed into a sequence of integers, each integer being the index of a token in the dictionary. Only the top N frequent words will be taken into account. The N is set to 32,000 for both the English and the Chinese vocabulary.

3.2. Data Preprocessing and Loading

After training the word segmentation model, we preprocess the data with the pytorch tool, i.e., `torch.utils.data.Dataset`. The main tasks include three steps. The first is to reorder the English–Chinese parallel corpus according to the length of English sentences. In this way, we aim to ensure that the sentences within a batch are of the same length. Then, we perform word segmentation for the parallel sentences with the trained model and map each word to a unique ID in the vocabulary. The third step is that we add a starting symbol and an ending symbol for each embedding sentence, which thus generates a tensor for model training.

3.3. Model Training

We use the `PyTorch` functional API to create a Transformer model [15]. After shuffling the news dataset, we create training, validation, and test sets with a 70–10–20% split. As a result, there are 176,943 bilingual sentences for training, 25,278 for validation, and 50,556 for testing. The network is trained with a batchsize of 32, an optimizer of Adam, and a self-designed loss function for a total of 40 epochs. The training loss reduces to 2.08, and the validation loss converge reaches 4.10. The parameters of our trained model are around 400 MB. The following subsections specifically show how we train the transformer-based NMT model.

3.3.1. The Transformer Model

Rather than using the RNN or CNN structure, Transformer is the first encoder-decoder model purely relying on the self-attention mechanism [15]. To be exact, Transformer only consists of self-attention and a feed-forward neural network. A real-world neural network can have many stacked encoder layers and decoder layers.

Figure 3 shows the network architecture of the Transformer model. We see that the `encoder` consists of multi-head self-attention and a position-wise feed-forward network. It takes the sum of input embedding and positional embedding as input. The `decoder` consists of masked multi-head self-attention, multi-head self-attention, and a position-wise feed-forward network. The decoder takes the sum of output embedding and positional embedding as input. A typical transformer model has 6 layers of encoder and 6 layers of decoder. However, our trained model is much larger than this configuration and is available upon request.

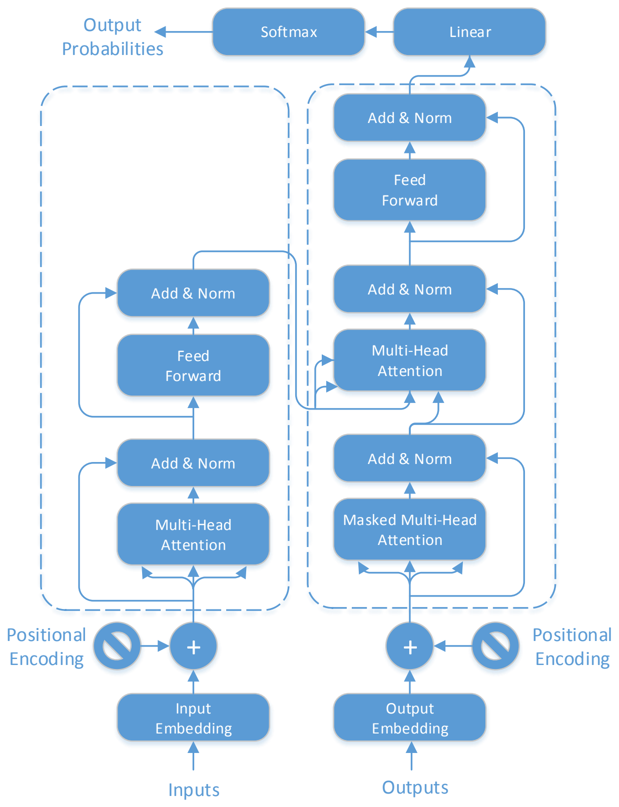


Figure 3. The Transformer model architecture.

3.3.2. Training Optimizer

We used the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate (lr) over the period of training, according to Equation (10):

$$lr = d_{model}^{-0.5} \cdot min(step\_num^{-0.5}, step\_num \cdot warmup\_step^{-1.5}), \tag{10}$$

where $step\_num$ denotes the current step number, and $warmup\_step$ is the number of steps used to warm up the training process. This corresponds to increasing the learning rate linearly for the first $warmup\_steps$ training steps and decreasing it thereafter proportionally to the inverse square root of the $step\_num$. Here, $warmup\_steps = 4000$. Figure 4 shows the implementation of the training optimizer. These parameters are selected in a trial-and-error approach.

3.3.3. Parallel Training

Training deep learning models with a large amount of training data is not a trivial task, which is performed in a high-performance computing infrastructure with a large number computing nodes or accelerators. Training NMT models also consumes a lot of computing resources. Thus, we choose to train our transformer model in a parallel way.

Training NMT models comes with many forms of parallelization, including data parallelism, model parallelism, pipeline parallelism, and hybrid forms of parallelism. In data parallelism, a number of workers load an identical copy of the deep learning model. The training data are partitioned into non-overlapping chunks and fed into the model replicas of the works for training. In model parallelism, the NMT model is partitioned, and each worker loads a different portion of the NMT model for training. The workers that hold the input layer of the model are fed with the training data. By contrast, pipeline parallelism combines the two aforementioned forms of parallelism.

```
class TrainOpt:
    """Optim wrapper that implements rate."""

    def __init__(self, model_size, factor, warmup, optimizer):
        self.optimizer = optimizer
        self._step = 0
        self.warmup = warmup
        self.factor = factor
        self.model_size = model_size
        self._rate = 0

    def step(self):
        """Update parameters and rate"""
        self._step += 1
        rate = self.rate()
        for p in self.optimizer.param_groups:
            p['lr'] = rate
        self._rate = rate
        self.optimizer.step()

    def rate(self, step=None):
        """Implement 'lrate' above"""
        if step is None:
            step = self._step
        return self.factor * (self.model_size ** (-0.5) * min(step ** (-0.5), step * self.
            ↪ warmup ** (-1.5)))
```

Figure 4. The implementation of the training optimizer.

In this work, we aim to train our NMT model on diverse available computing resources such as laptop CPUs, desktop CPUs, and server CPUs with one or multiple GPUs. In addition, we mainly use data parallelism to speed up the training process. As shown in Figure 5, the entire English–Chinese parallel corpus is partitioned into a large number of batches, and each batch of the training data is distributed to a processor of a GPU or multi-core CPU. Thus, we have to ensure that we have sufficient batches so as to fully utilize the whole GPU processor or multiple GPUs. Our implementation is built based on the *DataParallel* module of the *PyTorch* framework. Note that we have to use suitable APIs to create the model and perform data movements between CPUs and GPUs. The batch size is set to be 32.

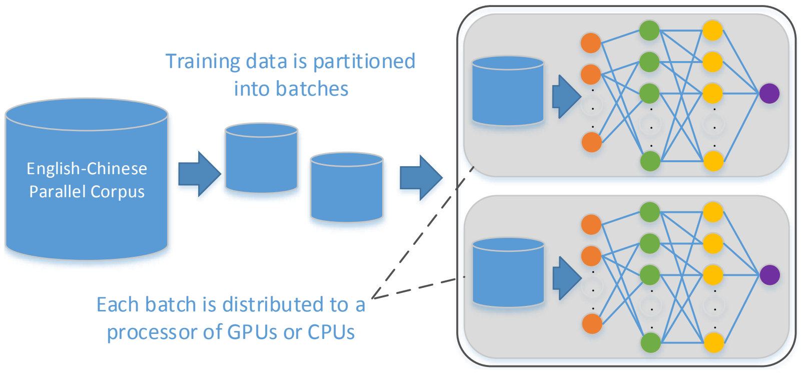

Training data is partitioned into batches
English-Chinese Parallel Corpus
Each batch is distributed to a processor of GPUs or CPUs

Figure 5. Data parallelism for training the transformer model.

3.4. Model Deployment

Once an NMT model has been trained, it can be used to translate a sentence into another language, i.e., the inference or decoding stage. Note that there is a clear distinction between training and inference: we only have access to the source sentence at the decoding time. We must initialize the transformer model and fill the model with parameters from the pytorch model data. We also must normalize the input sentence into a tensor which is taken into the decoder model (Figure 3). Then, the input sentence is translated into the output sentence with our trained model.

When doing inference, we can select the most likely word at each step in the output sequence. The simplest decoding algorithm is beam search decoding, which expands all the possible next steps and keeps the k most likely, where k is a user-specified parameter and controls the number of beams or parallel searches through the sequence of probabilities. The development set source sentences are decoded using combinations of beam size and a length penalty and the combination that gives the best evaluation metric score.

3.5. A DL-NMT Web Interface

For ease of use, we develop a Web interface for the machine translation system, which is shown in Figure 6. We aim to provide users with an easy-to-use interface for model training, corpus management, and model deployment. Thus, it has three modules for corpus, project, and model management.

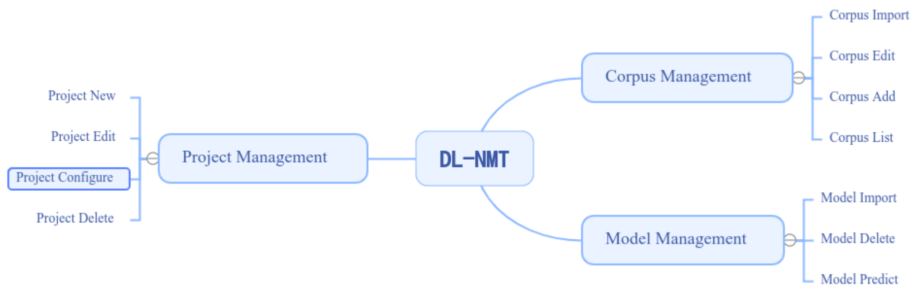


Figure 6. The DL-NMT web interface design.

3.5.1. Corpus Management

Many prior works have shown that the trained models can yield a better prediction accuracy from using a domain-specific bilingual corpus [20–22]. To this end, we provide domain experts to add or edit paired sentences. In this way, the domain experts can build their own corpus. For instance, teachers who majored in Business English can integrate their expertise of English–Chinese paired sentences into a corpus with our web interface. On the other hand, we also provide access to the existing bilingual corpus. At the initialization stage, users can choose to import the `UM-Corpus` of eight different text domains, including Education, Laws, Microblog, News, Science, Spoken, Subtitles, and Thesis, into our backend platform. Note that users can also edit the existing corpus with our web interface. With the help of this web interface, we aim to build a sufficiently large and diverse corpus from various domains to train NMT models.

3.5.2. Project Management

Our web interface provides a consequence of functions to manage the training process. We name an NMT model training process as a `project`. As a result, we can create, edit, or delete a project. To enable the ease of use, we provide a web interface to configure a model training task. The training parameters are listed in Table 1. Once ready, we use a one-stop button to start the training process. For the trained model, we can save it in a specified directory.

Table 1. The configurable training parameters.

Parameters	Notes	Defaults
src_vocab_size	English vocabulary size	32,000
tgt_vocab_size	Chinese vocabulary size	32,000
batch_size	Number of samples selected for one training	32
epoch_num	Number of epochs	40
d_model	Feature dimensions of the model	512
max_len	Max length of sentence	60
beam_size	Beam size for bleu	3
early_stop	Early stop for loss increase	5
optimizer	Use an optimizer for training	Adam
use_gpu	Whether or not to use a GPU	0

3.5.3. Model Management

Our web interface manages the trained NMT models. With the interface, users can import or delete a model. In particular, we provide an interface for users to input an English sentence, make a translation, and output a Chinese sentence. This is particularly suitable for teaching purposes, e.g., a live classroom demonstration. That is, we train the NMT model with the interface and then use the interface to demonstrate how well the trained model performs.

4. Results and Discussion

4.1. Experimental Setup

Hardware and Software. Our DL-NMT system can be deployed onto laptops, desktops, and servers with GPUs. In this work, we run the translation system on the platforms as shown in Table 2. The table also lists the frequency, the number of cores on each CPU, whether it has GPUs, the Linux kernel, and the GCC/OpenMP version.

Table 2. The hardware and system software.

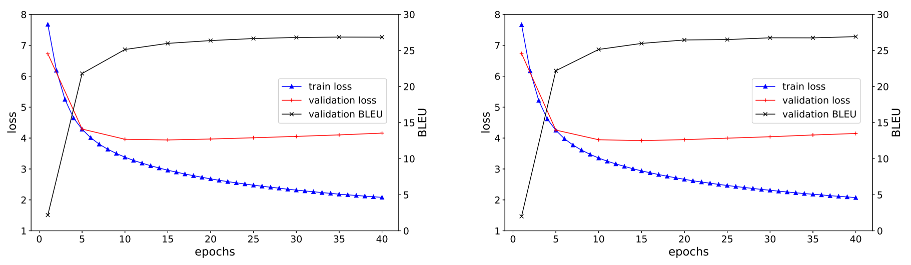
CPUs	Freq. ∣ #Core	GPUs	Category	Linux Kernel	OpenMP
Intel Core i7-7500U	2.7 GHz ∣ 2	N/A	Laptop	v4.15.0	GCC v7.5.0
Intel Core i7-7700K	4.2 GHz ∣ 4	N/A	Desktop	v4.15.0	GCC v7.5.0
Intel Xeon Platinum 9242	2.3 GHz ∣ 96	N/A	Server	v3.10.0	GCC v4.8.5
Intel Xeon Silver 4210	2.2 GHz ∣ 40	NVIDIA Titian RTX	Server	v4.18.0	GCC v8.4.1
Intel Xeon Silver 4210	2.2 GHz ∣ 40	NVIDIA GeForce RTX 2080Ti	Server	v4.18.0	GCC v8.4.1
Intel Xeon Silver 4210	2.2 GHz ∣ 40	NVIDIA GeForce RTX 2060 SUPER	Server	v4.18.0	GCC v8.3.1

Dataset Details. We use the news dataset from the UM-Corpus, which is a large English–Chinese parallel corpus. It provides a two million English–Chinese corpus from eight text domains, covering several topics and text genres, including Education, Laws, Microblog, News, Science, Spoken, Subtitles, and Thesis [23]. We train our models with the news subset of 252K sentences consisting of 10,635K Chinese words and 5672K English words. We split the training samples into 176,943 training pairs, 25,278 validation pairs, and 50,556 test pairs. Note that our web interface provides users with access to build a new corpus.

4.2. Performance Results

4.2.1. Training Accuracy

We train models on an NVIDIA RTX 2080Ti and NVIDIA Titan RTX. Since their memory is limited (11 GB and 24 GB), we use different batch sizes of 8 and 16, respectively, to avoid the out-of-memory (OOM) error. At the same time, to ensure the stability of the model, we use the gradient accumulation method to expand the batch size in another form. Gradient accumulation is to accumulate the gradients of several batches and then update the network parameters. The number of batches accumulated in each update is called the accumulation step. Figure 7 shows the training process on these two GPUs, respectively. Each curve describes the trend of loss (of the training set and validation set) and BLEU value of the validation set. We see that the training process on both GPUs is basically consistent. As the number of epochs increases, the train loss continues to decrease. In addition, the loss and BLEU values of the verification set are stable at half way. As a matter of fact, too many training iterations would lead to the problem of overfitting, which has a negative impact on the translation quality. Thus, we run the verification process for every five iterations to avoid the overfitting issue.



(a) RTX 2080Ti (b) Titan RTX

Figure 7. The training process on RTX 2080Ti (**a**) and Titan RTX (**b**).

We compare the test performance of the trained models on two GPUs in Table 3. The batch_size (bs) and the accumulation step (step) used in the gradient accumulation method for each GPU are indicated in the table. The different GPU devices basically have no impact on the model training effect. The gradient accumulation method can facilitate us in achieving the same performance with small batches as with large batches.

Table 3. The best Dev BLEU and the Test Loss and BLEU on two GPUs.

GPU	Best Dev BLEU	Test Loss	Test BLEU
RTX 2080Ti (bs = 8, step = 4)	26.86	4.15	26.86
Titan RTX (bs = 16, step = 2)	26.92	4.16	26.88

4.2.2. Decoding Stage

We use the beam search algorithm to look for the best solution in the inference stage. It expands the solution space relative to the greed search and reduces the complexity relative to the exhaustive search. The beam size is an important parameter of the beam search algorithm, affecting performance and efficiency. We compare the test BLEU and the test time based on different beam sizes (from 1 to 6) in Figure 8 and Table 4. The results indicate that the translation performance and time consumption will increase as the beam size increases. However, when the value of the beam size exceeds three, the performance improvement is negligible, while the time consumption increases steeply. To conclude, the best beam size is 3 on the Titan RTX.

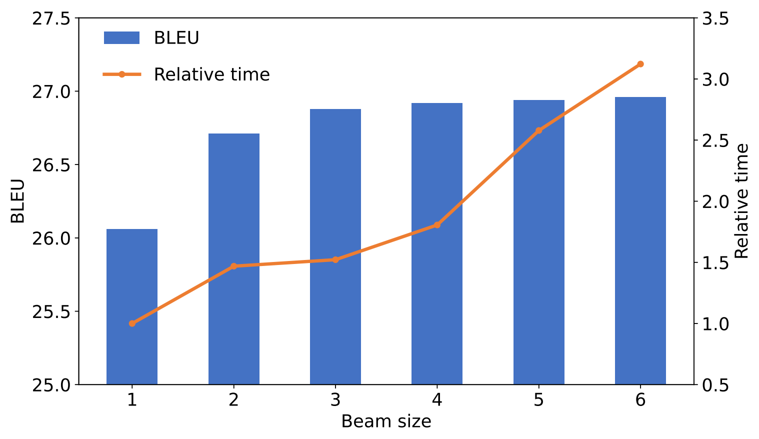


Figure 8. The test BLEU and relative test time (relative to beam size=1).

Table 4. The test BLEU and test time based on different beam sizes.

Beam Size	1	2	3	4	5	6
BLEU	26.06	26.71	26.88	26.92	26.94	26.96
Time (s)	1700	2497	2587	3070	4384	5308

4.2.3. Training Speed

The value of batch size and the compute capability of the processor affect the training speed. We first compare the training time (per epoch) when using different batch sizes on the same device. We choose the Titan RTX as the platform because it is fast enough and has enough memory. Figure 9 shows that the training speed increases with the increase in batch size, but the growth rate gradually slows down.

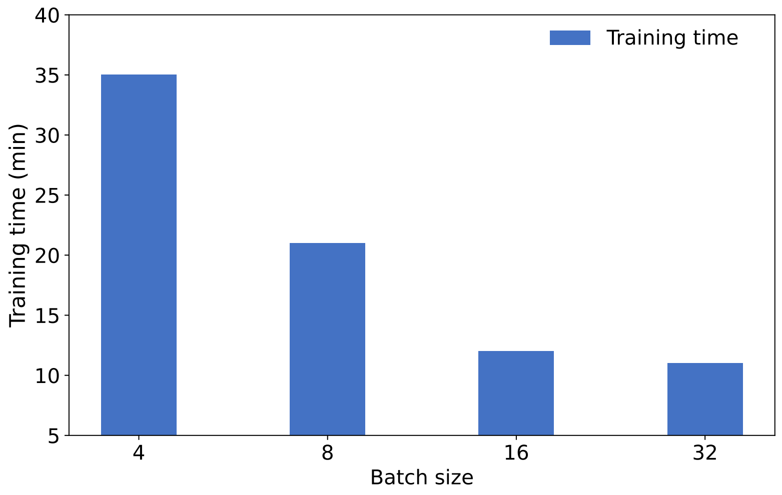


Figure 9. The training time (minutes) per epoch on Titan RTX based on different batch sizes.

Meanwhile, we compare the training speed of the CPU and GPUs in Figure 10. Due to the limitation of the memory space of the RTX 2080Ti and RTX 2060 Super, we use a batch size of 8 and 16 on them, respectively. Meanwhile, we use a batch size of 32 on the Titan RTX. The training time per epoch is 480 min, 250 min, 170 min, 22 min, 18 min, and 11 min for the Intel i7-7500U CPU (Laptop), Intel i7-7700K CPU (Desktop), Intel Xeon Platinum 9242 CPU (Xeon server), Titan RTX, RTX 2080Ti (RTX2080), and RTX 2060 Super (RTX2060). For the Xeon CPU sever, even when adopting a batch size of 96, the training time is around

15× as long as that on the Titan RTX GPU. That is, the GPU has absolute superiority in training neural networks.

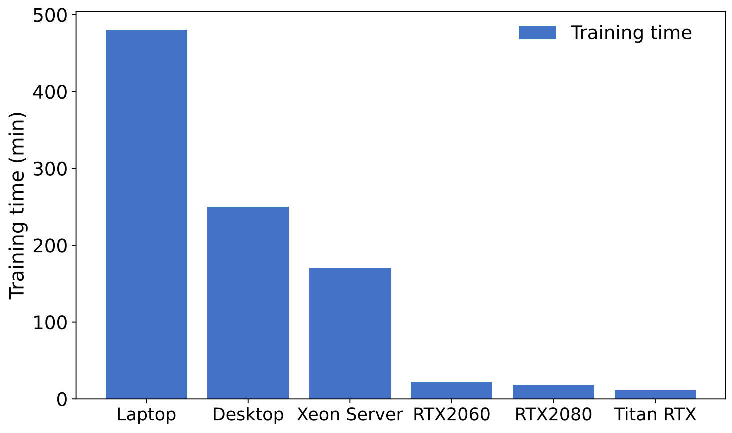
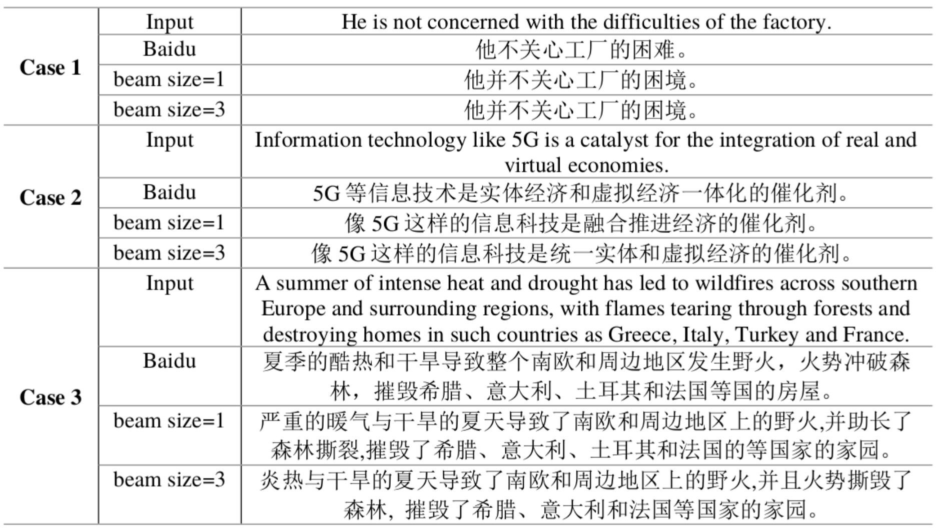


Figure 10. The training time (minutes) per epoch on different devices.

4.3. Case Study

In addition to analyzing performance parameters, we also compare the actual translation of our model to the industry-level Baidu translation system. Here, the accuracy is measured based on the options of native speakers. We randomly select three different lengths of sentences as test cases and show the comparison results in Figure 11. Overall, our model can translate English to Chinese correctly, especially for short sentences and medium-length sentences (case 1 and case 2). However, the accuracy of the translation is not sufficiently good. In addition, the beam search strategy (beam size is 3) is better than the greed search strategy (beam size is 1) when decoding the sentence. In a nutshell, by consuming a relatively short training time, our trained model is competitive with an industry-level product such as the Baidu translation system. In the future, we will use a more bilingual corpus for improved translation quality.



Case 1	Input	He is not concerned with the difficulties of the factory.
	Baidu	他不关心工厂的困难。
	beam size=1	他并不关心工厂的困境。
	beam size=3	他并不关心工厂的困境。
Case 2	Input	Information technology like 5G is a catalyst for the integration of real and virtual economies.
	Baidu	5G 等信息技术是实体经济和虚拟经济一体化的催化剂。
	beam size=1	像 5G 这样的信息科技是融合推进经济的催化剂。
	beam size=3	像 5G 这样的信息科技是统一实体和虚拟经济的催化剂。
Case 3	Input	A summer of intense heat and drought has led to wildfires across southern Europe and surrounding regions, with flames tearing through forests and destroying homes in such countries as Greece, Italy, Turkey and France.
	Baidu	夏季的酷热和干旱导致整个南欧和周边地区发生野火，火势冲破森林，摧毁希腊、意大利、土耳其和法国等国的房屋。
	beam size=1	严重的暖气与干旱的夏天导致了南欧和周边地区上的野火,并助长了森林撕裂,摧毁了希腊、意大利、土耳其和法国的等国家的家园。
	beam size=3	炎热与干旱的夏天导致了南欧和周边地区上的野火,并且火势撕毁了森林，摧毁了希腊、意大利和法国等国家的家园。

Figure 11. The translation cases of using Baidu translation system and our model.

4.4. Discussion

Implementing the Transformer-based translation system from scratch is indeed not new. However, we believe that our translation system stands out and can be applied in several scenarios. For now, large pretrained models have achieved promising results and have been widely accepted. Although each increase has brought significant performance improvements in downstream NLP tasks, training such models requires large-scale specialized computing hardware such as Google's TPUs. These computing clusters are typically unaffordable for small/medium-sized enterprises. Our translation system is portable across laptop CPUs, desktops CPU, and server CPUs with one or multiple GPUs. Such platforms are typically affordable for small/medium-sized enterprises, and our translation system can be used as a research infrastructure for such companies.

On the other hand, the large pretrained models are too complicated, and their capacity is too large for us to understand. That is, we know the models perform well, but we do not know the reasons. They work similar to a "black-box" and are particularly unsuitable for teaching purposes. Instead, our translation system can be used as a teaching demonstration tool for students majoring in translation. In particular, we have provided a web interface to manage the corpus, model training, and model prediction to ease the use of our translation system. For instance, our system provides research professors with a web interface to collect their translation expertise so as to build a new corpus.

5. Conclusions

In this work, we have implemented a deep learning machine translation system based on a news corpus. The deep learning algorithm takes in English text as input and uses an encoder-decoder model with an attention mechanism based on Google's Transformer to translate the text to Chinese output. The model was trained using a simple self-designed entropy loss function and an Adam optimizer on paired English and Chinese text sentences from the news area of the UM-Corpus. We train the model on high-end GPUs with a parallel approach. During training time, we not only track loss over training epochs, but measure the quality of our model's translations using the BLEU score. The experimental results on the UM-corpus show that our trained model can achieve a maximum BLEU score of 29.2. We can further improve this score by tuning other hyperparameters and increasing the complexity of our model, as well as by training on a larger subset of the data to avoid biased results. As a case study, we compare the performance of our model to that of Baidu's and show that our model can compete with the production-level translation system.

For future work, we plan to train our models with large-scale GPU-based clusters. We also want to incorporate language features into the model to improve its translation quality. In addition, we will use a more bilingual corpus for improved translation quality.

Author Contributions: Conceptualization, L.Z. and J.F.; methodology, L.Z., J.F., and W.G.; validation, W.G. and J.F.; writing—original draft preparation, L.Z. and W.G.; writing—review and editing, L.Z. and J.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the National Natural Science Foundation of China under Grant agreement 61972408.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data collected during this study may be obtained by contacting the corresponding author at zhaolanxin21@yeah.net.

Acknowledgments: We thank the anonymous reviewers for their constructive comments and feedback.

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

References

1. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; Kingsbury, B. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
2. Dahl, G.E.; Yu, D.; Deng, L.; Acero, A. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Trans. Speech Audio Process.* **2012**, *20*, 30–42. [CrossRef]
3. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105.

4. Ciresan, D.C.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; IEEE Computer Society: Los Alamitos, CA, USA, 2012; pp. 3642–3649.
5. Le, Q.V.; Ranzato, M.; Monga, R.; Devin, M.; Corrado, G.; Chen, K.; Dean, J.; Ng, A.Y. Building high-level features using large scale unsupervised learning. In Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, 26 June–1 July 2012.
6. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
7. Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; Chen, Z. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual, Austria, 3–7 May 2021.
8. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. 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, NAACL-HLT 2019, Minneapolis, MN, USA, 2–7 June 2019; Burstein, J., Doran, C., Solorio, T., Eds.; (Long and Short Papers); Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; Volume 1, pp. 4171–4186. [CrossRef]
9. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33, Proceedings of the Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual, 6–12 December 2020*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020.
10. Forcada, M.L.; Ginestí-Rosell, M.; Nordfalk, J.; O'Regan, J.; Ortiz-Rojas, S.; Pérez-Ortiz, J.A.; Sánchez-Martínez, F.; Ramírez-Sánchez, G.; Tyers, F.M. Apertium: A free/open-source platform for rule-based machine translation. *Mach. Transl.* **2011**, *25*, 127–144. [CrossRef]
11. Koehn, P.; Och, F.J.; Marcu, D. Statistical Phrase-Based Translation. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, AB, Canada, 27 May–1 June 2003; Hearst, M.A., Ostendorf, M., Eds.; The Association for Computational Linguistics: Stroudsburg, PA, USA, 2003.
12. Cho, K.; van Merrienboer, B.; Bahdanau, D.; Bengio, Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In Proceedings of the SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014; Wu, D., Carpuat, M., Carreras, X., Vecchi, E.M., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2014; pp. 103–111.
13. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
14. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 1243–1252.
15. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017*; Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 5998–6008.
16. Radford, A.; Narasimhan, K. Improving Language Understanding by Generative Pre-Training, 2018. Available online: https://gregraiz.com/wp-content/uploads/2020/07/language_understanding_paper.pdf (accessed on 11 May 2021).
17. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* **2019**, *1*, 9.
18. Rosset, C.; Xiong, C.; Phan, M.; Song, X.; Bennett, P.N.; Tiwary, S. Knowledge-Aware Language Model Pretraining. *arXiv* **2020**, arXiv:2007.00655.
19. Norrie, T.; Patil, N.; Yoon, D.H.; Kurian, G.; Li, S.; Laudon, J.; Young, C.; Jouppi, N.P.; Patterson, D.A. Google's Training Chips Revealed: TPUv2 and TPUv3. In Proceedings of the IEEE Hot Chips 32 Symposium, HCS 2020, Palo Alto, CA, USA, 16–18 August 2020; IEEE Computer Society: Los Alamitos, CA, USA, 2020; pp. 1–70. [CrossRef]
20. Ling, W.; Marujo, L.; Dyer, C.; Black, A.W.; Trancoso, I. Crowdsourcing High-Quality Parallel Data Extraction from Twitter. In Proceedings of the Ninth Workshop on Statistical Machine Translation, Baltimore, MD, USA, 26–27 June 2014; pp. 426–436. [CrossRef]
21. Ling, W.; Marujo, L.; Dyer, C.; Black, A.W.; Trancoso, I. Mining Parallel Corpora from Sina Weibo and Twitter. *Comput. Linguist.* **2016**, *42*, 307–343. [CrossRef]
22. Ling, W.; Xiang, G.; Dyer, C.; Black, A.W.; Trancoso, I. Microblogs as Parallel Corpora. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Sofia, Bulgaria, 4–9 August 2013; Long Papers; The Association for Computer Linguistics: Stroudsburg, PA, USA, 2013; Volume 1, pp. 176–186.

23. Tian, L.; Wong, D.F.; Chao, L.S.; Quaresma, P.; Oliveira, F.; Yi, L. UM-Corpus: A Large English-Chinese Parallel Corpus for Statistical Machine Translation. In Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, 26–31 May 2014; Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S., Eds.; European Language Resources Association (ELRA): Luxemburg, 2014; pp. 1837–1842.
24. Kalchbrenner, N.; Blunsom, P. Recurrent Continuous Translation Models. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, Grand Hyatt Seattle, Seattle, WA, USA, 18–21 October 2013; pp. 1700–1709.
25. Sennrich, R.; Haddow, B.; Birch, A. 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), Berlin, Germany, 7–12 August 2016; Volume 1, pp. 1715–1725.
26. Schuster, M.; Nakajima, K. Japanese and Korean voice search. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, 25–30 March 2012; IEEE Computer Society: Los Alamitos, CA, USA, 2012; pp. 5149–5152.
27. Kudo, T.; Richardson, J. 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, EMNLP 2018: System Demonstrations, Brussels, Belgium, 31 October–4 November 2018; Blanco, E., Lu, W., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; pp. 66–71.
28. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
29. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv* **2016**, arXiv:1609.08144.
30. Hassan, H.; Aue, A.; Chen, C.; Chowdhary, V.; Clark, J.; Federmann, C.; Huang, X.; Junczys-Dowmunt, M.; Lewis, W.; Li, M.; et al. Achieving Human Parity on Automatic Chinese to English News Translation. *arXiv* **2018**, arXiv:1803.05567.
31. Lin, X.; Liu, J.; Zhang, J.; Lim, S.J. A Novel Beam Search to Improve Neural Machine Translation for English-Chinese. *Comput. Mater. Contin.* **2020**, *65*, 387–404. [CrossRef]
32. Zhou, L.; Zhang, J.; Kang, X.; Zong, C. Deep Neural Network-based Machine Translation System Combination. *ACM Trans. Asian Low Resour. Lang. Inf. Process.* **2020**, *19*, 65:1–65:19. [CrossRef]
33. Xiong, H.; He, Z.; Hu, X.; Wu, H. Multi-Channel Encoder for Neural Machine Translation. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, LA, USA, 2–7 February 2018; pp. 4962–4969.
34. Wang, Y.; Cheng, S.; Jiang, L.; Yang, J.; Chen, W.; Li, M.; Shi, L.; Wang, Y.; Yang, H. Sogou Neural Machine Translation Systems for WMT17. In Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, 7–8 September 2017; pp. 410–415.
35. Wu, S.; Wang, X.; Wang, L.; Liu, F.; Xie, J.; Tu, Z.; Shi, S.; Li, M. Tencent Neural Machine Translation Systems for the WMT20 News Translation Task. In Proceedings of the Fifth Conference on Machine Translation, WMT@EMNLP 2020, Online, 19–20 November 2020; pp. 313–319.
36. Yang, J.; Wu, S.; Zhang, D.; Li, Z.; Zhou, M. Improved Neural Machine Translation with Chinese Phonologic Features. In *Natural Language Processing and Chinese Computing, Proceedings of the 7th CCF International Conference, NLPCC 2018, Hohhot, China, 26–30 August 2018*; Zhang, M., Ng, V., Zhao, D., Li, S., Zan, H., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11108, pp. 303–315.
37. Kuang, S.; Han, L. Apply Chinese Radicals Into Neural Machine Translation: Deeper Than Character Level. *arXiv* **2018**, arXiv:1805.01565.