3. Experiment Setup

In this chapter, we describe our selection of datasets, framework, and automatic evaluation we used for the experiments. We start by describing our set of training and test data, our reasoning for choosing the dataset, and the tokenization algorithm in Section 3.1. We then move forward to the framework we use to implement the neural network, training, and evaluation phase in Section 3.2. Finally, we discuss the automatic evaluation we use during the experiments in Section 3.3.

3.1 German-to-English Dataset

The scope of our experiment is in a single language pair: German→English. We only select a single language pair because we want to focus our experiment on understanding the behaviour of BERT and the adapters in machine translation, not on generalization across multiple language pairs. We select IWSLT14 and WMT19 as our primary datasets. IWSLT14 will be mainly used as the dataset for fine-tuning and testing the final performance of the model, while WMT19 is used for the additional dataset in pre-training as well as in training some of our baselines.

3.1.1 IWSLT 2014

The 2014 IWSLT evaluation ([22]) is the fourth instance of a shared task started in 2010. This task is mainly focusing on the translation of TED Talks. The task is based on a collection of public speeches that cover various topics. All the talks in the collections have English captions. These captions are then translated into many languages by various volunteers worldwide. As TED talks are recorded events of speakers sharing their thoughts and experience, we need to deal with spoken language rather than written language. Spoken language is expected to be less complex and formal than written language.

Both the in-domain training and development data are available through the WIT3 website.[1] There is also out-of-domain training data provided through the workshop website. We focus only on the English German pairs in this work and ignore the other available languages. The evaluation dataset (tst2014) comprises talks from the previous years, and the 2014 talks are included in the training sets. Furthermore, to improve the reliability of assessing the MT progress over the years, evaluation sets from previous years (tst2013) are also distributed together with tst2014. On the other hand, development sets (dev2010, tst2010, tst2011, and tst2012) are kept intact from the past editions.

Evaluation sets tst2014 for the German language (De→En) are derived from the ASR task. Therefore, it is ensured that no overlap exists with other tasks that employ TED talks. In addition to the TED talks, there is a series of independent talks called TEDx. The difference lies in the location of the events. TED talks mainly focus on the North American region, while TEDx can be held in various areas worldwide. The TEDx-based corpus was proposed in 2013 for De→En as an additional test set to put more rigour into the evaluation process. Finally, a concatenation of the TEDx and TED-based development sets consisting of dev2010, tst2010, tst2011 and tst2012 sets were released. The complete statistics of the dataset can be seen in Table 3.1.

[1] `https://wit3.fbk.eu/`

	set	sentences	tokens	
			En	De
	train	172k	3.46M	3.24M
dev	TED.dev2010	887	20,1k	19,1k
	TED.tst2010	1,565	32,0k	30,3k
	TED.tst2011	1,433	26,9k	26,3k
	TED.tst2012	1,700	30,7k	29,2k
test	TED.tst2013	993	20,9k	19,7k
	TED.tst2014	1,305	24,8k	23,8k
	TEDx.dev2012	1,165	21,6k	20,8k
	TEDx.tst2013	1,363	23,3k	22,4k
	TEDx.tst2014	1,414	28,1k	27,6k

Table 3.1: Statistics of IWSLT 2014 German→English dataset.

3.1.2 WMT 2019

WMT19 dataset was first introduced in The Fourth Conference on Machine Translation (WMT) held at ACL 2019 ([5]). The primary objectives of this conference are to evaluate the state-of-the-art models in MT, advertise public access to the MT models' performance and the common test sets, and improve the methods to evaluate and estimate machine translation. There are various shared tasks within the conference that evaluate different machine translation aspects. This conference has been conducted 13 times and the current conference is built on top of the previous editions ([43, 16, 17, 20, 18, 21, 19, 7, 8, 12, 10, 9, 13]).

The dataset was collected from various news sources on the Internet. As mentioned before, we use WMT for the pre-training dataset and an additional dataset to train our baseline models. For this reason, we are not utilizing the dev and test set. Therefore, we show the statistics of the dataset in Table 3.2 only for the training set. Apart from a large number of sentences, another reason for choosing WMT19 as our additional dataset is that it contains sentence pairs from various domains.

corpus	sent	tokens	
		De	En
Europarl Parallel Corpus	1,8M	48,1M	50,5M
News Commentary Parallel Corpus	0,3M	8,3M	8,2M
Common Crawl Parallel Corpus	2,3M	54,5M	58,8M
ParaCrawl Parallel Corpus	31,3M	559,3M	598,3M
EU Press Release Parallel Corpus	1,4M	29,4M	30M
WikiTitles Parallel Corpus	1,3M	2,8M	3,2M

Table 3.2: Statistics of WMT 2019 German→English dataset.

3.1.3 Tokenization

BERT uses the subword tokenization algorithm called WordPiece ([70]) to construct the list of vocabularies. The algorithm is very similar to Byte-Pair Encoding (BPE; [72]), where it relies on a pre-tokenizer to split words within the training data, such as simple whitespace tokenization.

After the pre-tokenization, the set of words with their frequency is gathered. BPE then starts building a symbol vocabulary that consists of all symbols within the corpus. The symbol can be anything from alphabetical, numeric, and other symbols. BPE then learns a set of rules to merge and form a new symbol from two other symbols from the existing vocabulary. This process is repeated until the size of vocabulary items matches the desired vocabulary size.

To provide an example, let us assume that we have the following words and their frequencies[2]:

("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

From these words, we have the set of initial symbols: "["b", "u", "n", "p", "h", "g", "s"]". BPE then starts the merging process by using the total frequency of each possible symbol pair. The pair that occurs the most often will be picked as a new vocabulary item. In our example, we have "h" followed by "u" with a total of 15 occurrences (10 times in `hug` and 5 times in `hugs`) and "u" followed by "g" with

[2]The example is taken from `https://huggingface.co/docs/transformers/tokenizer_summary`

a total of 20 occurrences. Therefore, we pick "ug" and append the new symbol to the vocabulary. We repeat this process until we meet the desired number of vocabulary items.

During the decoding process and assuming the following set of unique symbols: "["b", "u", "n", "p", "h", "g", "s", "ug"]", we now perform the tokenization process by matching the sub-word of the input word to the existing vocabulary. For example, if the incoming word is "mug", we will have ["[UNK]", "ug"] as our tokenization as we do not have "m" in our vocabulary. "[UNK]" is introduced as a special symbol to handle tokens/symbols that do not exist in the vocabulary. On the other hand, the word "bug" will be tokenized into ["b", "ug"].

3.2 Framework

3.2.1 HuggingFace Transformers

Transformers ([85]) is a library created by the Huggingface team that implements various transformer-based architectures. This library's primary aim is to implement transformer models specifically designed for research and production. This implies that the library is easy to read, extend, and supported by the industrial-strength implementation for deployment in production. The library also aims to facilitate transformer-based pre-trained model distribution to the public. Under the same foundation, this library supports the distribution and re-use of various pre-trained models in a centralized hub. This includes the configuration, such as the hyperparameters used to instantiate the models and the pre-trained weights. This improves research reproducibility as numerous users can now re-use and improve their experiments based on the pre-trained models.

The fact that the library is continuously maintained by the Huggingface team and supported by over 400 external contributors is one of our reasons for choosing Huggingface to conduct the experiments in this work. The library is released under

the Apache 2.0 license and is freely available on GitHub and their official website.[3] Furthermore, the website also provides easy-to-understand tutorials and detailed documentation of the API.

3.2.2 AdapterHub

Another reason we choose Huggingface is the availability of AdapterHub ([57]). Despite adapter's simplicity and achieving strong results in multi-task and cross-lingual transfer learning ([56, 58]), reusing and sharing adapters was not straightforward. Adapters are rarely released independently due to their subtle differences in architecture and strong dependence on the base model, task, and language. AdapterHub was created to facilitate the easiness of training models with adapters and share the fine-tuned adapters in various settings to mitigate these issues.

3.3 Automatic Evaluation

Bilingual Evaluation Understudy or `BLEU` ([52]) is one of the evaluation metrics that is widely used to evaluate MT models. It works by evaluating the output of an MT system (the hypothesis) with a set of manually translated references.

BLEU works by measuring the percentage of the matching n-grams in the hypothesis and taking the difference in length of the hypothesis and references as a form of penalty. The percentage of n-grams is often interpreted as a measurement of precision. However, in some cases, simply calculating the matching n-grams may lead to a misinterpretation of the output. In the case of unigram ($n = 1$), we can see the matching n-grams as the number of tokens available in the hypothesis and the references divided by the total number of tokens. See the following example:

Hypothesis: you you you you

Reference: I think you should know that you are right

[3] `https://github.com/huggingface/` and `https://huggingface.co`

The above example will lead to 100% unigram precision of the hypothesis despite the result only sharing the word **you**. To mitigate the precision problem, the shared number of n-grams in the hypothesis and the reference must be clipped by the number of n-grams in the reference. The following is the updated BLEU formula after incorporating the n-gram clipping:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{x \in C} Count_{clip}(x)}{\sum_{C' \in \{Candidates\}} \sum_{x' \in C'} Count(x')} \tag{3.1}$$

Where x is the set of possible n-grams from a particular sentence used when calculating the score.

We mentioned that besides n-grams, BLEU also considers the length of the hypothesis as a penalty score. This is useful when dealing with short candidates within the hypothesis. The following is the example that shows the problematic output:

Hypothesis: you

Reference: I think you should know that you are right

Despite having a 100% precision score, the hypothesis does not represent the correct translation compared to the reference. We want to elude such a problem by introducing a penalty called **brevity penalty**. The penalty works by measuring the length of the hypothesis relative to the reference length and $e^{(1-r/c)}$ when the candidate is shorter than the reference length. To be more specific, we refer to the equation below.

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{3.2}$$

In the case of more than one reference, r is the length of the reference with the closest length to the hypothesis, called the **effective reference length**. One must note that the choice of reference will vary between different implementations

of BLEU. To be more precise, take the following example that shows the difference between two references is equal to 1, where one reference is one word shorter than the hypothesis and the other is one word longer:

Candidate: I eat fish on the beach with

Reference 1: I eat fish on the beach with friends

Reference 2: I eat fish on the beach

Combining the above components, BLEU is defined as follows:

$$BLEU = BP \cdot exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3.3}$$

We define w_n as the weights for each n-grams scores (p_n) where $\sum_{i=1}^{\infty} w_i = 1$. w_n is usually set uniformly across all n depending on the number of n-grams used in the calculation. When 4-grams is used, it is set to $\frac{1}{4}$.

By default, BLEU computes the n-gram precision from unigrams to 4-grams. We perform the final score computation by calculating the average for each n-gram score and multiplying them by the brevity penalty (BP).

In this work, we follow the implementation of sacrebleu[4] ([60]) that is wrapped under Huggingface Transformers framework.

[4] `https://github.com/mjpost/sacreBLEU`