Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Improving Neural Machine Translation Robustness via Data Augmentation

Author:
Zhenhao Li

Supervisor:
Prof. Lucia Specia

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science / [Artificial Intelligence] of Imperial College London

September 2019

Abstract

Neural Machine Translation (NMT) models have shown strong ability in translating clean texts, but they are susceptible to noises in the input. Improving NMT models robustness can be seen as a form of "domain" adaptation to noise. The recent Machine Translation on Noisy Text (MTNT) corpus provides noisy parallel data for a few language pairs, but this data is very limited in size and diversity. In this thesis, we compare different data augmentation methods on noisy data. Specifically, we use back-translation, forward-translation, and fuzzy match techniques. Besides, we propose to use parallel transcripts from speech to improve robustness of neural machine translation models. We compare our best-performing systems with the submissions of WMT19 Robustness Task and show that the system with our data augmentation methods could achieve a competitive position.

Acknowledgments

Many people helped me in this thesis and I would like to acknowledge their effort and help to this thesis.

First of all, I would like to thank my supervisor Prof. Lucia Specia for her great guidance. She is always willing to answer my questions and give me constructive advice. She also shared me with the GPU resource, without it this project would not be possible.

Besides, I want to thank my colleagues at Imperial College London, who helped me with technical issues. During the communication with them, I could learn from other's work and keep my mind updated.

Finally, I would also like to thank my girlfriend Wenqing Peng and my family. Their encouragement helped me during my hard time. With their support, I could fully focus on my work and devote myself to this project.

Contents

Chapter 1

Introduction

1.1 History of Machine Translation

The abilities to read and write are often regarded as human's distinguished intelligence. In the 7111 living languages, 3995 languages have a developed writing system [34]. Natural language processing (NLP), the field of accessing and processing human languages with computers, is becoming an emerging area in Artificial Intelligence (AI). Unlike images or acoustic signals, natural language is symbolic, ambiguous, and context-related, thus posing challenges for NLP. With the development of civilization, it is necessary to build systems that can aid humans in communication across languages.

Machine Translation (MT), referring to using computer software or system to translate one language into another automatically, is a branch area of natural language processing. Different levels of machine translation are commonly divided into translation by words, syntax, semantics, and finally the interlingua, as shown in Figure 1.1. Similar to human's translating behavior, MT systems first analyze source language into interlingua, which represents the languages in an abstract and universal meaning, and then generate target language based on it.

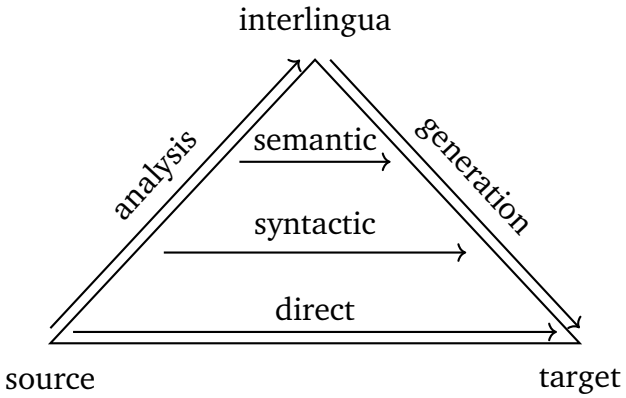

Figure 1.1: The Vauquois triangle that illustrates different levels of machine translation.

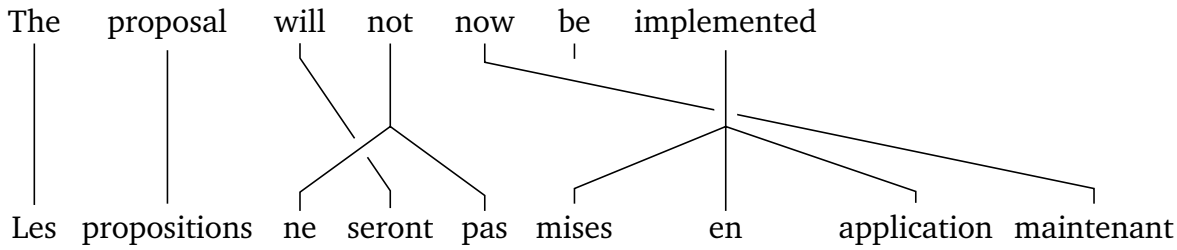

Figure 1.2: Word alignment example [4].

The origin of machine translation could date back to 60 years ago when Warren Weaver first proposed the idea of translation using computers [41]. Early approaches to MT involved using human-crafted rules to format grammar and syntactic structures. However, these rule-based machine translation systems were complex and dependent on linguistic knowledge. After that, statistical machine translation became mainstream. Instead of formalizing syntactic rules and using a pre-built bilingual dictionary, Statistical Machine Translation (SMT) utilizes translation examples in parallel corpus and learns the translation model in a probabilistic aspect. Standard statistical machine translation learns an alignment model from the parallel corpus, which finds a mapping between source and target words [5], as illustrated in Figure 1.2. When the alignment table is learned, the model goes through a process of choosing alignment, choosing target words, and reordering to produce the target sentence. In the subsequent research, word phrases are used to replace the individual word in the alignment learning [23] and currently, most statistical machine translation systems are phrase-based.

Although the SMT system could outperform the rule-based system, it suffers from the complicated pipeline and long-term dependency missing. Since translations are processed phrase by phrase, the long-distance context in sentences might be ignored. With the increase in data size and computational efficiency, neural methods are becoming popular in machine translation. Standard deep neural network (DNN) could only apply to tasks with fixed-length inputs and outputs. In machine translation, where inputs and outputs are sentences with flexible length, it might be difficult to use simple DNN models. To overcome the problem of flexibility, sequence-to-sequence (seq2seq) model is popular in neural machine translation (NMT) [37]. The seq2seq model comprises of an encoder-decoder structure. The encoder represents the input sentence into a context vector, and the decoder uses it to generate a corresponding sentence in the target language(See Figure 1.3).

The sequence-to-sequence model is strong in dealing with both sequential input and output. Besides, the seq2seq is flexible in that the encoder and decoder can be any network, either recurrent network, convolutional network or transformer [12, 40, 42]. This flexibility encourages research to explore the effect of different model architectures.

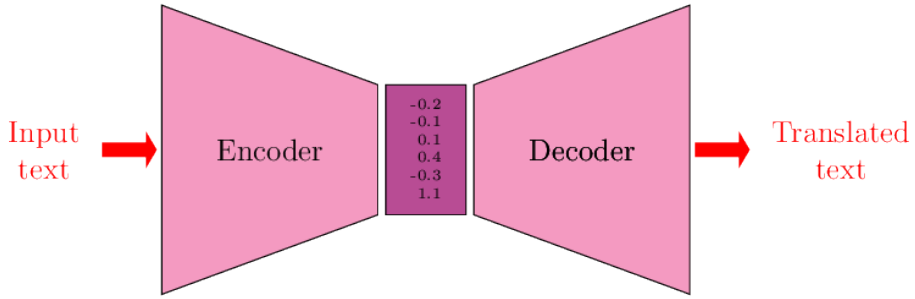

Figure 1.3: Sequence-to-sequence structure

The proposal of attention mechanism further strengthens the NMT model's ability to learn long-term contexts [1]. The original seq2seq model uses the last hidden state in the encoder (which is an LSTM network [16]) as the context. However, after a long sequence of recurrent structure, the last hidden state might loss context information at the beginning of the input sentence. To overcome this issue, an attention layer is added in the encoder, taking a weighted average over all encoder hidden states to form the context vector.

Based on the attention and seq2seq structure, NMT models could achieve similar performance to SMT models [1]. In recent years, new model architectures such as convs2s [12] and transformer [40] improve NMT model's performance, and thus outperform the SMT models. Considering its flexibility in training as well, neural methods have now become the mainstream in machine translation.

1.2 Robustness Problem

Despite the success of using neural methods in machine translation, NMT still has its weakness. Lacking robustness is one of the fatal problems. Current NMT systems are trained on clean texts which source from news reports, official documents and so forth. However, natural languages are not always clean and formal. For example, speaking languages might contain informal words such as "wanna" and "gonna" that hardly appear in clean training data; comments on social media contain noises such as abbreviations, spelling errors, etc. Those NMT systems trained with clean data can hardly generalize and translate precisely informal texts with noise [2]. Therefore, it is still a long distance for NMT systems to translate human language well.

The tasks of improving robustness can be seen in two aspects. Firstly, a robust translation model should be able to keep a similar level of noises between the source and target texts. Social media comments usually contain some tokens that rarely occur in formal corpora, such as emojis and emoticons. For these kinds of noise, the NMT model is expected to keep them remained in the translations, rather than treating as unknown words. In this thesis, we name this task as "*preserving informal text*". On the other hand, we would expect the NMT model to be robust and could denoise the target sentences. For example, when the input sentence contains typos or abbrevi-

ations, we want the model to recognize these kinds of noise and translate a clean sentence without such noises. In short, we define this task as "*denoising noisy text*". Since the model is supposed to denoise some noises in the source text, we assume that the source sentence should be noisier than the target sentence, which we define as "*noise gap*".

To improve the NMT model's robustness to noise, parallel data from noisy text is needed. Synthetic noise is used to construct noisy parallel data from clean data [2]. Although models trained on perturbed sentences could improve robustness, they can hardly generalize on noisy sentences from other ways of perturbation. Compared to synthetic noise, parallel corpus with natural noise could help to deal with different types of noisy human texts.

Michel et al. [28] proposed the Machine Translation on Noisy Text (MTNT) dataset, which contains natural comments crawled from Reddit[1] and corresponding human translations. The MTNT dataset has two language directions, namely Fr↔En and Ja↔En, and covers various types of noises, including emoji/emoticon, jargon, spelling errors, etc. Although the MTNT dataset provides resources for improving NMT Robustness, the size of this corpus is very limited. For example, the training set of MTNT on Fr→En contains only 19,161 sentence pairs, which is far from enough to train NMT models with millions of parameters.

In the WMT19 Robustness Task[2] [25], improving NMT robustness was treated as a domain adaptation problem. Models trained on large clean corpus were adapted to the noisy domain by fine-tuning on the small noisy corpus. In the shared task, the MTNT dataset was used as in-domain data. The domain adaptation was commonly conducted in two methods: fine-tuning on in-domain data [8, 31] and mixed training with domain tags at the beginning of sentences [3, 43]. Due to the limited size of the MTNT dataset, most approaches participating in the task performed noisy data augmentation using back-translation [3, 15, 43], with some approaches also adding synthetic noise [3]. In addition to adapting models on noisy parallel data, other techniques have been used to improve performance, generally measured according to BLEU [30] against clean references. For example, Berard et al. [3] proposed inline-casing by adding special tokens before each word to represent word casing. In [29], placeholders were used to help to translate sentences with emojis.

Although the submissions in the shared task have achieved good performance on noisy translations, we believe that the model could perform even better with more noisy data in high quality. As mentioned before, the robustness of an NMT model can be seen as a combination of "preserving informal text" and "denoising noisy text". Based on these assumptions, back-translation on clean texts, which is broadly used in the shared task, might be limited since it removes all noises from the translations, despite it provides a large volume of extra data. Therefore, in this thesis, we ex-

[1] `https://www.reddit.com/`

[2] `http://www.statmt.org/wmt19/robustness.html`

plore the effect of different data augmentation methods and generate noisy data to improve NMT model's robustness.

1.3 Contribution

In this thesis, we explore data augmentation techniques for robustness in neural machine translation. We follow the WMT19 Robustness Task and experiment under constrained and unconstrained data settings on Fr↔En language pairs. Under the constrained setting, we only use datasets provided by the shared task and propose new data augmentation methods to generate noises from the given data. This thesis has mainly three contributions:

(1) We compare different techniques of domain adaptation on noisy data. We compare the method of fine-tuning on noisy data with domain-sensitive training. Strength and weakness of both methods are discussed in this thesis.

(2) We explore effects of various data augmentation methods on robustness, namely the back-translation(BT), forward-translation(FT) and fuzzy match method adapted from [6]. Results show that our methods of data augmentation could extend limited noisy parallel data and improve model robustness. In addition, we experiment under the unconstrained setting using external dataset. We propose for the first time the use of speech datasets, in two forms: a) the IWSLT [7] and MuST-C [11] human transcripts as a source of spontaneous speech data, and b) automatically generated transcripts for the MuST-C dataset as another source of noise. We show that using informal language from spoken language datasets can also help to increase NMT robustness, both on noisy and clean texts.

(3) We combine the methods and in addition, apply with some tricks used in the shared task. By submitting our best-performing systems to the WMT19 Leaderboard, we show that our proposed methods could achieve competitive positions. Although our system do not beat the state-of-the-art system, it is trained with a smaller model and less data, which might be more efficient for training.

1.4 Thesis outline

This thesis is structured as follows. In Chapter 2, we review the background of this thesis and provide a general understanding of this work. In Chapter 3, we provide the information on the experiment settings, including the corpora we used, how we augment noisy data and the models we use. Chapter 4 presents the result and analysis of the experiments. We evaluate and show the effect of noisy parallel data, our augmented data, and external data on improving robustness. Finally, in Chapter 5, we give the conclusion and discuss briefly future work in NMT robustness.

Chapter 2

Background

In this chapter, we will give an overview of the background and related work. Neural Machine Translation has been a popular realm in machine translation within the recent five years. In the first section, we will discuss the development of NMT and some relevant architectures, ranging from the original sequence-to-sequence to the Transformer. Besides, we will also describe some data augmentation techniques in NMT, from which we got our inspiration. In the second section, we will go further into the issue of machine translation robustness and introduce recent works in this topic, precisely the WMT19 shared task.

2.1 Neural Machine Translation

Machine Translation (MT) is an important topic in the realm of Natural Language Processing. A good machine translation model can translate an input sentence $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ by consulting the parallel corpus and finding out a sentence in the target language with the most likelihood. In mathematical aspect, this can be represented as $\max_y P(\mathbf{y}|\mathbf{x})$. During the training stage, the machine translation model learns from all sentence pairs from the parallel corpus and maximizes the likelihood over the whole corpus. Traditional machine translation systems were rule-based, which required complex and language-oriented linguistic knowledge. Due to the inflexibility, rule-based MT was soon replaced by statistical machine translation (SMT). Popular SMT system is built on phrase-based model [23], and involves a sophisticated pipeline of building alignment table, doing phrase translation and re-ordering the output sentence. After sequence-to-sequence model [37] and attention mechanism [1] were proposed, neural machine translation (NMT) models could achieve similar or even better performance than SMT models, but with a simpler building procedure. In recent years, more architectures on NMT are explored and thus improving the NMT models' performance [12, 40]. Now, NMT has become the mainstream in machine translation.

Sutskever et al. [37] proposed the sequence-to-sequence (seq2seq) structure. The seq2seq model contains an encoder and a decoder. As is shown in Figure 2.1, the encoder (red) receives the input sequence after it is passed through an embedding

layer. The source sequence is passed through a recurrent layer and encoded into a context vector (green). The decoder (blue) takes the context vector as input and generates the translation token by token.

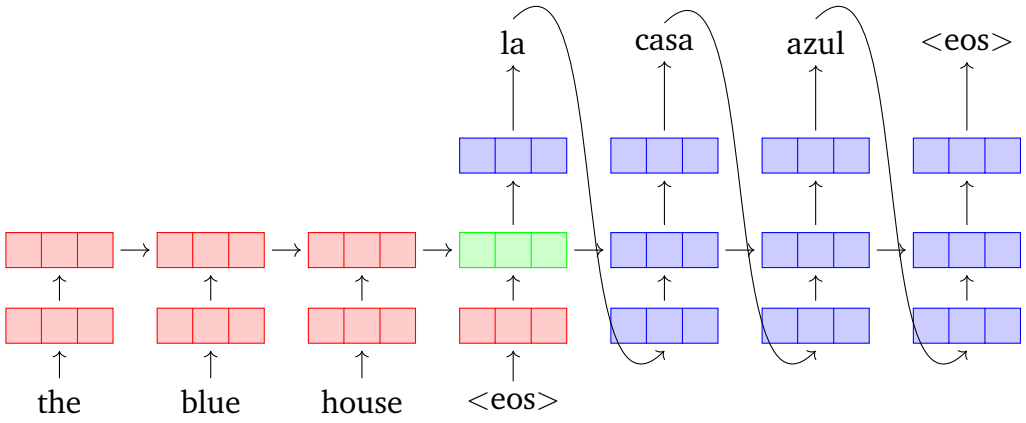

Figure 2.1: An illustration of seq2seq model structure

In the seq2seq model, both encoder and decoder use an LSTM network [16], with the context vector being the last hidden states in the encoder. This architecture could better handle various-length input and output sequence, compared to standard deep neural networks. Moreover, the encoder-decoder structure in this model provides a flexibility since it can be replaced with any other networks. The proposal of seq2seq model, which could achieve similar performance as best SMT models, marked a beginning of NMT.

2.1.1 Attention-based model

Although the sequence-to-sequence model shows a good ability in dealing with flexible sequence, it still faces the problem with long input sentences. The recurrent structure of encoder might cause loss of context, especially when the input sequence is in a high length. To cope with the long-term context problem, Bahdanau et al. [1] proposed an attention mechanism based on seq2seq model. They introduced an attention layer in the decoder to construct the context vector dynamically. At each decoding step, the model attends to all hidden states in the encoder and forms a context vector, instead of only using the last hidden states. Figure 2.2 shows how attention works.

In Bahdanau's model, a bidirectional LSTM network was used as an encoder to model input context from left-to-right and right-to-left. Therefore, the encoder hidden states h_j was calculated as a concatenation of the two hidden on both direction

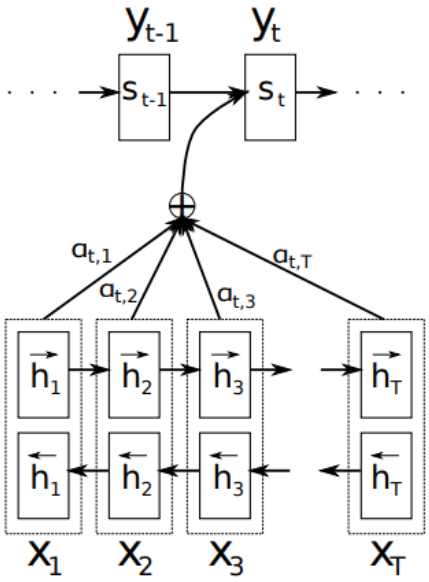

Figure 2.2: Attention-based encoder[1]

(see Eq 2.1).

$$h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}] \tag{2.1}$$

At each decoding stage i, an alignment score e_{ij} is calculated over all input tokens through an alignment function (see Eq 2.2). The alignment function gives a score of how well the previous output y_{i-1} is aligned with the source input x_j. In Bahdanau's experiment, the alignment function was computed using a feed-forward neural network.

$$e_{ij} = a(s_{i-1}, hj) \tag{2.2}$$

A softmax layer is applied to on top of it to transform the alignments into a probability distribution a_{ij}. In this step, an attention matrix is created (see Eq 2.3). Each element in the attention matrix a_{ij} represents the importance of input token x_i when decoding output y_j.

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_z} \exp(e_{ik})} \tag{2.3}$$

Since the attention is in a probability distribution that sums up to one, it can be treated as weights of all encoder hidden states. Therefore, the context vector c_i is calculated dynamically by averaging on all encoder hidden states based on the attention weights (see Eq 2.4).

$$c_i = \sum_{j=1}^{T_z} a_{ij} h_j \tag{2.4}$$

With the introduction of attention layer, the model could take the whole input sequence as the context and learn the importance of each token when decoding. The attention-based model shows a better performance than the seq2seq model without

attention, especially in sentences with large length. In addition, by learning the alignment function, the attention can be visualized and thus shows how the model learns to align.

Luong et al. [26] further investigated the effects of different types of attention. They compared models using global attention (see Figure 2.3) and local attention (see Figure 2.4). With global attention the decoder forms context vector using all tokens from the source sequence while local attention only attends to a smaller window of source tokens. Both types of attention were shown to be beneficial to NMT models.

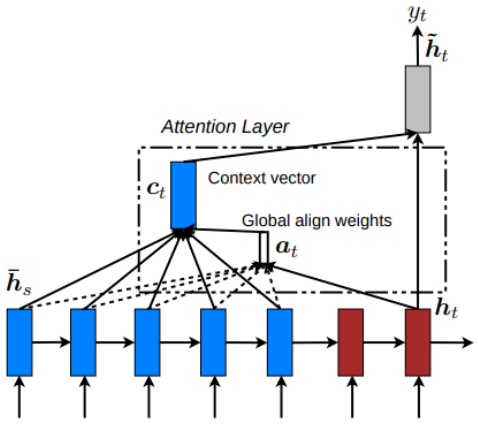

Figure 2.3: Global Attention Model [26]

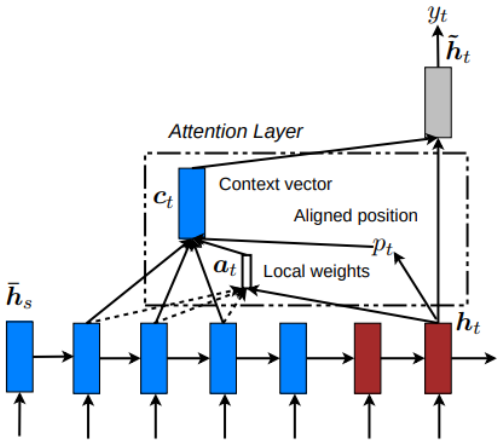

Figure 2.4: Local Attention Model [26]

Besides, they also explored the use of different alignment functions in the process of calculating attention. In Bahdanau's work, a feed-forward network was used to learn the alignment function. Instead of this, Luong et al. tried different approaches (as shown in Eq 2.5)[1]. The alternatives to the feed-forward network showed similar performance but resulted in a more straightforward and effective computation.

$$e_{ij} = \begin{cases} \mathbf{h}_{i-1}^T \mathbf{h}_j & \text{dot} \\ \mathbf{h}_{i-1}^T \mathbf{W_a} \mathbf{h}_j & \text{general} \\ \mathbf{W_a}[\mathbf{h}_{i-1}; \mathbf{h}_j] & \text{concat} \end{cases} \tag{2.5}$$

2.1.2 ConvS2S

The recurrent architecture has its strength in representing the sequence order and context, while it suffers from the drawback of non-parallel computing. To avoid this flaw, Gehring et al. [12] proposed the ConvS2S model, which is based on a multi-layer convolutional architecture. The encoder and decoder of the ConvS2S model are comprised of several convolutional layers with non-linearity function applied (See Figure 2.5).

[1]We changed the annotations from the paper in order to keep a same set of annotations as used before.

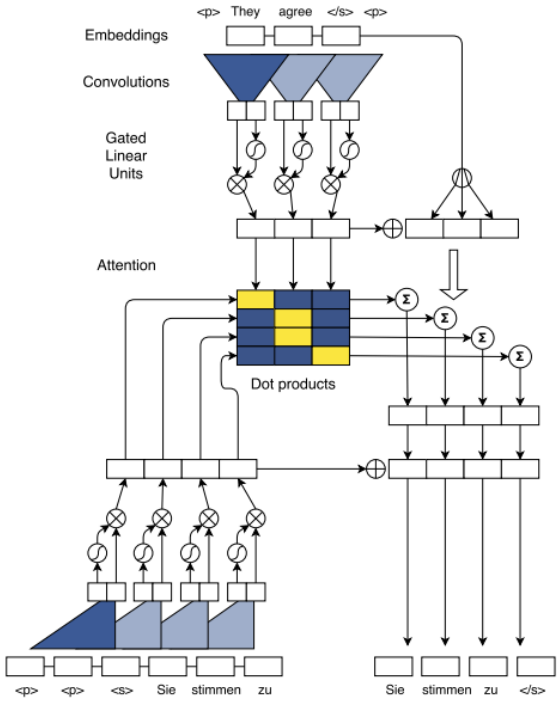

Figure 2.5: ConvS2S architecture [12]

During training, the source and target sequences are first mapped into the vector space using the embedding layers on both sides. After that, the word embedding sequence $\mathbf{w} = (w_1, w_2, ..., w_m)$ is added with a position embedding $\mathbf{p} = (p_1, p_2, ..., p_m)$ to form a new word representation $\mathbf{e} = (w_1+p_1, w_2+p_2, ..., w_m+p_m)$, which encodes the absolute position of each word into the vector representation. This step is to add information related to word positions since the convolutional network does not model word orders as the recurrent network. Unlike the recurrent network which takes the whole sequence as the context, the convolution layer computes context within a fixed-length window, and the resulting vector after convolution functions plays the same role as the hidden state in the recurrent neural network. The hidden states are passed into a Gated Linear Unit Layer (GLU) [9] to perform non-linearity. The convolution and GLU network are stacked with multiple layers, with residual connections [14] between the convolution layers. Unlike the traditional attention-based model, the ConvS2S also applied a multi-step attention, which computes attentions at each decoder layer.

The ConvS2S model was evaluated on three WMT translation tasks, namely English-Romanian, English-German, and English-Franch. The model outperformed the RNN-based models, and also, reduced the computation complexity. It showed an advantage over RNN-based models in terms of training and translating speed on both GPU and CPU. In the evaluation of English-German translation task, both the introduction of position embedding and the multi-step attention were shown to benefit the model.

2.1.3 Transformer

Either the recurrent network (RNN) or convolutional network (CNN) is dependent on the sequential order of the input sentences, and this might hinder parallel computation. To substitute the RNN or CNN, Vaswani et al. [40] proposed the Transformer architecture, which uses only attention in the encoder and decoder. In the standard transformer model, both encoder and decoder use a stack of 6 identical layers (see Figure 2.6). Each layer contains a multi-head attention layer connected with a position-wise feed-forward neural network. On top of the two sub-layers, layer normalization and residual connection are added to avoid overfitting. Since the transformer model does not contain any information regarding sentence orders, positional encoding is added after the embedding layer as well.

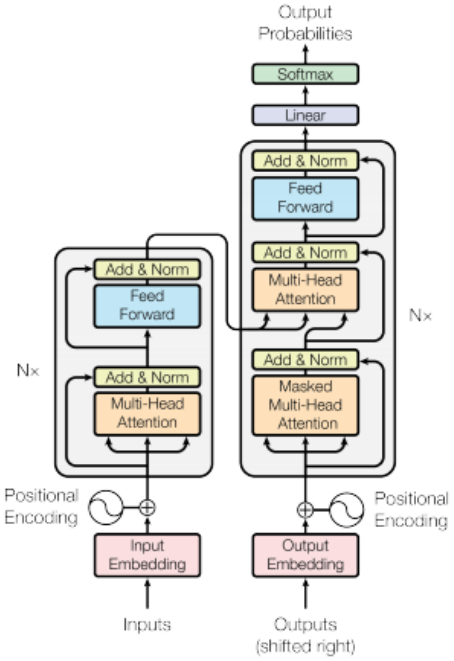

Figure 2.6: Transformer architecture [40]

Figure 2.6 shows the architecture of a multi-head attention layer. As presented in the transformer paper, the attention mechanism can be described as "mapping a query and a set of key-value pairs to an output" [40]. Based on this intuition, after the embedding layer, the input sequence is duplicated into three vectors, namely the Query q, Key k and Value v vectors of different dimensionalities. Then the query, key and value vectors are packed together over the sequence to form matrix Q, K and V. After softmaxing the multiplication of query and key, the attention matrix is obtained. Finally, the attention is multiplied with the value V, resulting in the context matrix of each token's attention to all other tokens in the sentence. The computation

process is showed in Eq 2.6 and Figure 2.8.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.6}$$

Since the scaled dot-product attention outputs a matrix that represents each token's attention to all tokens in the sentence, it is named as self-attention. Instead of doing a single scaled dot-product attention, the set of query, key, value vectors is linearly projected into different dimensionalities to perform the multi-head attention (as shown in Figure 2.6). The outputs after scale dot-product attention are concatenated and fed into a linear layer, thus the output of multi-head attention would keep the same dimensionality as the input.

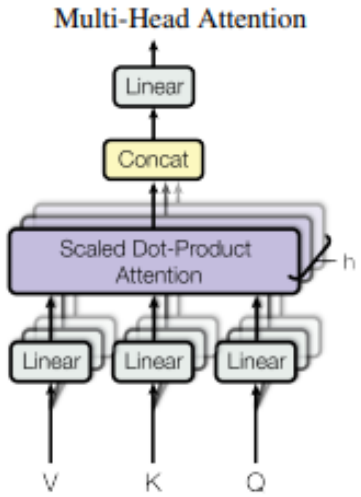

Figure 2.7: Multi-head attention [40]

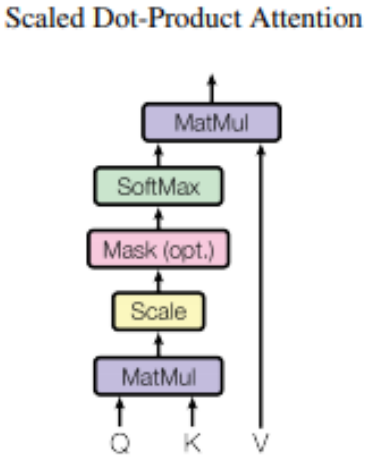

Figure 2.8: Scaled dot-product attention [40]

In the transformer model, all computations can be finished in parallel after replacing the recurrent network with multi-head attention. In Vaswani et al.'s paper [40], eight heads were used in the multi-head attention. The transformer outperforms other models in neural machine translation and has become a popular choice of model architecture in NMT.

2.1.4 Data augmentation in NMT

While models training on human-translated parallel data could achieve state-of-the-art performance in NMT, large volumes of monolingual data might be neglected. Sennrich et al. [32] first proposed the back-translation technique to utilize monolingual data in neural machine translation. The procedure of back-translation involves first training a target-to-source model on parallel data and then using it to generate synthetic parallel data by translating target language monolingual data. The synthetic parallel data from back-translation can be mixed with the raw parallel data

and used to train a better model. Although the quality of back-translation might not be as precise as human translations, it provides an option to utilize monolingual data and improves model performance. Another method to utilize monolingual data is the dummy source sentences [32]. With this technique, the monolingual sentences are treated as parallel examples with empty source pairs. Therefore, the source sentence is attached with a "<null>" token to denote that it is empty. During the training of monolingual data with dummy source sentences, the encoder parameters are frozen. This step plays a role of training a language model on the target language, and thus the decoder could benefit from the extra monolingual data.

Bulte et al. [6] proposed the method of finding fuzzy matches to augment parallel data. The fuzzy match algorithm finds similar sentences from a parallel corpus and matches the counterpart's translations to generate new sentence pairs. The algorithm loops through the training data (S, T) and compares each source sentence $s_i \in S$ with all other source sentences $s_j \in S(s_i \neq s_j)$. If the similarity between the two sentences is higher than a threshold λ, then these two sentences are recognized as fuzzy matches. The similarity between sentences was computed with Leveshtein edit distance [24] on token level, in Bulte et al.'s work [6]. Since looping through the whole corpus to extract fuzzy matches is costly, they first used a Python library `SetSimilaritySearch`[2] to select possible similar candidates before calculating the edit distance. To further boost the speed, they applied a multi-threading and selected only the top_n candidates from the result of similar sets. The speeding methods could improve the algorithm speed by approximately 300 times without losing much data. From their experiment, 41.3% sentences in the test data could find a fuzzy match in the training data, with a similarity threshold of 50%. This result suggests the similar distribution of sentences in the same corpus and the feasibility of using the fuzzy match approach to augment parallel data.

2.2 Machine Translation Robustness

Noisy language is another problem in neural machine translation, and even in natural language processing. The human's natural language contains noises such as typos, abbreviations, etc. Human can easily recognize the noisy texts while the current NMT models might fail. Considering the following sentences [10]:

> *Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.*

Human can read and comprehend the disordered sentences while the current NMT models show poor performance and fail to translation the above paragraph [2].

[2] `https://github.com/ekzhu/SetSimilaritySearch`

Despite the promising performance of Neural Machine Translation in common translation tasks, the current architectures are far from handling complex, nonstandard natural languages. As illustrated in [22], the current NMT models show poor performance in dealing with out-of-domain data, for examples the out-of-vocabulary words or rare words. Moreover, the thrive of current NMT models is based on a large amount of human-translated, standard parallel corpus, while human languages could be noisy, informal, and arbitrary. For example, we selected an example from the test set of MTNT corpus and used Google Translate[3] to translate it into English. The raw sentence, human translation, and hypothesis translation from Google are listed in Table 2.1.

src	Aaaaaaaah.... 8 ans aprs, je viens de percuter.... :o 'tain je me disais bien que je passais ct d'un truc vu les upvotes.
ref	Aaaaaaaah.... 8 years later, I've just realized.... :o damn I had the feeling that I was missing something considering the upvotes.
hyp	Aaaaaaaah .... 8 years later, I just hit ....: o 'tain I told myself that I was next to something seen the upvotes.

Table 2.1: Illustration of a noisy sentence translated by Google Translate.

The source sentence is a comment on social media and contains several types of noise, such as spelling errors ("Aaaaaaaah"), spoken language ("tain" with similar pronunciation as "damn") and jargons ("upvote"). Looking into the hypothesis translation, we might see that with the noise injected, the translation quality is heavily hindered. The current state-of-the-art machine translation systems are built on parallel corpora with formal languages; hence, they are sensitive to noise in the input sentences. Therefore, improving the robustness of current NMT models to input noises has become a promising and popular topic in machine translation.

To come up with criteria for evaluation, Michel et al. [28] proposed a parallel corpus, Machine Translation of Noisy Text (MTNT), which contains noisy sentences on both source and target languages. The corpus was extracted from Reddit[4] and translated by human translators. Two language pairs, namely English-French and English-Japanese, in both directions were provided. They also provided a non-exhaustive list of different types of noise on social media, as shown in Table 2.2.

As mentioned before, the task of improving machine translation robustness is to filter out noises in the target sentences while keeping a similar level of informality to the source sentences. Some noises in the source sentences should be kept in the target sentences to render the translation on a similar informal level. These types of noises, for examples, emojis/emoticons and capitalized words, often have low frequency in clean parallel corpora. In this thesis, we define the task of keeping these types of noise as "preserving informal texts". On the other aspect, the input might

[3] `https://translate.google.com/`
[4] https://www.reddit.com/

Type	**Examples**
Spelling errors	"across" → "accross", "receive" → "recieve"
Word perturbation	"je n'aime pas" → "j'aime pas", "je pense" → "moi je pense"
Grammatical errors	"a ton of" → "a tons of", "fewer people" → "less people"
Spoken language	"want to" → "wanna", "je ne sais pas" → "chais pas"
Internet slang	"to be honest" → "tbh", "shaking my head" → "smh"
Improper casing	"just do it" → "JUST DO IT"
Code switching	"This is cute" → "This is kawaii"
Jargon	"upvote", "downvote"
Emoji	❤️, 😭, 😂, 😉
Profanities/slurs	"f*ck" ...

Table 2.2: Different types of noise on social media [28], with examples in the format of "clean words" → "noisy words".

contain noises needed to be removed in the target sentence, which often comes from misspelling, masked words, etc. We define the task of removing such types of noise as "denoising noisy text". To deal with the noises from errors, a domain adaptation on noisy data could benefit the robustness. As shown in Michel et al.'s [28] work, by simply fine-tuning on the noisy parallel corpus, the BLEU score on noisy texts could improve to a large extent without harming model performance on clean texts.

Since noisy texts such as emojis and emoticons have low frequency in the training corpus, the task of "preserving informal texts" can be seen as dealing with the out-of-vocabulary words/rare words. To denoise noisy texts, the model could be adapted to the noisy domain. In the following subsections, we will introduce methods of improving robustness in the above two aspects, and then review the WMT19 Robustness Task.

2.2.1 OOV/rare words

Out-of-vocabulary word has always been a problem in NLP, and it is not an exception in machine translation. A common practice in neural machine translation is to uses a limited-size vocabulary, ranging from 30k to 80k. When the model translates words out of the vocabulary, it merely treats it as an unknown token and thus outputs a <UNK> token in the translation, meaning that the source token is unknown to the model. Therefore, this sets a limitation that the training and test dataset should be in the same distribution (similar vocabulary). Otherwise, the model might perform poorly on test data that contains numerous rare or OOV words [1, 37]. To ease this problem, many methods are used to deal with the OOV problem, including introducing an extra dictionary and segmenting smaller tokens.

Jean [17] et al. proposed a model-specific method and trained the neural machine translation model on a large target vocabulary. A sampling method was used to reduce the computation efficiency while increasing the target vocabulary. During each

gradient update, only a small subset of the whole target vocabulary was sampled, which approximated the negative term in the gradient. This method was evaluated on English-German and English-French translation task, and when the target vocabulary covered the whole training set, the OOV rate in the test set was largely deceased to less than 3%.

Different from the model-based method, Luong [27] et al. followed the methods of dealing with OOV in statistical machine translation. The phrase-based SMT models use a phrase table (which can be much larger than the vocabulary in NMT) and explicit alignments to resolve the problem. Luong et al. applied this method in NMT models. With an extra phrase dictionary at the post-processing step, the model could replace the <UNK> token in the output with corresponding target words. Each time the model encountered an unknown source word, it would look up the dictionary to find the corresponding phrase translation, or copy the source word with the highest attention to the output if no dictionary provided. Besides, they proposed three annotation strategy on the training data, namely Copyable, PosAll, and PosUnk, to let the NMT model learn the post-processing step automatically. In the Copyable annotation, each unknown target word is aligned with a source word, and word with no alignment is represented with a special $<\text{unk}_{\emptyset}>$. With the PosAll annotation, a token representing the relative position is added to each unknown word, while in PosUnk only aligned unknown word is annotated. Among all the three annotating strategies, the model based on PosUnk gived the best performance and showed a robust translation on less frequent sentences. Their ensembled system yielded an improvement of 2.8 BLEU score on the WMT14 en-fr translation task against non-annotated models.

Another method to alleviate rare words problem is to segment words into smaller tokens. Following this intuition, Sennrich et al. [33] proposed the use of Byte Pair Encoding (BPE) in NMT to learn subwords in the corpus. The procedure of learning BPE is unsupervised. The algorithm first loops through the corpus and separates all words into characters. Then an iterative merge operation is conducted, with each iteration the algorithm merges the most frequent subword pairs. After the merge operations, a subword vocabulary is generated. Using this subword vocabulary, words in the parallel corpus are segmented into subwords, with a special delimiter to represent the segmentation. Therefore, by merging the subwords connected by the delimiter, standard word-level translations can be restored. The applying of BPE is completed in the preprocessing step, thus it can generalize to all model architectures. During training, rather than learn word context from the corpus, the model learns subword context instead. Although separating words into subwords might increase the sequence length and cost more computational resource, the introduction of BPE could reduce the vocabulary to a large extent, helping to alleviate rare words problem. Sennrich et al. compared BPE-based models with models using dictionary back-off and showed that the BPE could improve the performance by $+1.1$ and $+1.3$ BLEU [30] on En$\rightarrow$DE and EN$\rightarrow$RU translation tasks.

2.2.2 Domain adaptation on noisy text

Denoising noisy texts is different from treating the rare words, since the errors in sentences could be random and arbitrary. Not only should the model translate normal clean words, but it also has to recognize the correct form of the wrong words. A possible solution is to train on texts with such types of noise so that the model could generalize on noisy texts with similar errors. Therefore, improving the robustness can be seen as domain adaptation on the "noisy" domain.

To address this problem, some research focus on data augmentation methods that inject synthetic noise into the parallel corpus. Belinkov et al. [2] conducted research based on the above meme [10], and introduced a structure-invariant word representation by averaging all character representations to solve the word disorder problem. When testing current state-of-the-art NMT models on noisy test data, almost all models suffered from a severe performance decrease. In order to verify the importance of the word order, they experimented on different kinds of strategies of injecting synthetic noise (See Table 2.3).

Swap	randomly swapping two letters in a word (e.g. noise→nosie)
Mid	randomize letters order with first and last letters fixed (e.g. noise→nisoe)
Rand	randomize all letters in a word (e.g. noise→isoen)
Key	replace one letter with an nearby keyboard letter (e.g. noise→noide)

Table 2.3: Different kinds of synthetic noise in [2]

They introduced an order-invariant model named meanChar operating on character-level. The model represents words by taking an average over all its characters, thus rendering it robust to character disorder. However, the meanChar model expressed an inconsistent performance concerning different kinds of noise despite its better performance than order-sensitive models. To further increase the model robustness to noise, Belinkov et al. injected both synthetic and natural noises to clean data for training. They used the IWSLT parallel corpus [7] as it is transcribed from TED talks and might contain noises related to spoken languages. Results revealed that the injection of noises in the training data would help increase model robustness, but only on the same type of noise. Moreover, the natural noise showed a different pattern with the synthetic ones, which illustrated the need for more natural noisy data in the robustness task.

Likewise, Khayrallah et al. [19] explored the impact of different kinds of noise as well. Different from the work on synthetic noise, they used crawled web data which contains human-generated noise as training data. The comparison of statistical machine translation model with neural machine translation model showed different results on noisy data. The NMT model's performance showed a significant decrease of -9.9 BLEU score on noisy texts while the SMT model showed an increase of +1.2

BLEU score after adding noisy crawled data. The NMT models learned to copy unknown source words into the target output, which might be beneficial in translating name entities but harmful when translating noisy texts.

Sperber et al. [35] utilized artificial noise in speech recognition. Different from writing texts, oral languages might involve more abbreviations and informal uses. Therefore, the transcripts for speech might be a better choice for selecting noisy parallel corpus. Simulating the errors in speech-generated texts, they proposed a noise generating model that could corrupt the clean data and introduce noises such as character insertion, substitution, and deletion. In their experiments, introducing more noises in the training data was found to help translate noisy texts, but harmful to clean texts. Therefore, it might be an issue to trade off between clean and noisy data.

Not only using rule-based synthetic noisy data, Vaibhav et al. [39] tried data augmentation method using back translation [32]. By building an NMT model on the opposite direction, pseudo source texts could be attained by passing the target sentences through the back-translation model (since the translation cannot be perfect, it might contain noise). The texts generated by the back-translation model were used as source texts, and this might simulate the noises in informal languages (See Fig 2.9). They used a tagged back-translation technique by attaching a tag in front of each sentence, representing the dataset where this sentence sources from. The tagged back-translation could improve the model robustness and outperform the untagged method. The tag at the beginning of each sentence plays a role of "domain" constraint, rendering the model to translate with specific styles (e.g., noisy text).

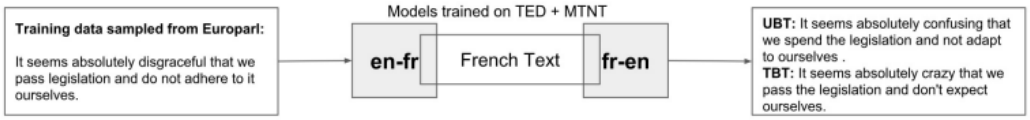

Figure 2.9: Pipeline for back-translation based noise injection [39]

Karpukhin et al. [18] used a Transformer model with a CNN-based character encoder. Therefore, the model could be invariant to word order. The model was trained on various types of noise and evaluated on the corresponding test sets. Results showed the benefit of noise-introduced training data while the model performance was inconsistent when evaluated on MTNT test set. This would suggest that synthetic noises might differ from natural noises, and model trained simply on the dataset with synthetic noises might not perform well on the natural noisy texts.

2.2.3 WMT19 Robustness Task

The WMT19 Robustness Task [25] is a shared task focusing on utilizing noisy parallel corpus to improve NMT model's performance on noisy texts. The task has

constrained and unconstrained settings. Under the constrained setting, only provided corpora are allowed for training. For example, in Fr↔En direction, the MTNT dataset is used as in-domain data (noisy) while the out-of-domain data is the data from the WMT15 news translation task. Apart from parallel data, monolingual data from both clean and noisy corpora is also allowed under the constrained setting. Instead of using the test data in the MTNT corpus, the shared task provided another set of test data as a blind test, named as MTNT2019. The MTNT2019 test set was extracted following the same way of genegerating the MTNT data. Size of the MTNT2019 test data is presented in Table 2.4.

	En-Fr	Fr-En	En-Ja	Ja-En
# sentence	1,401	1,233	1,392	1,111
# source tokens	20.0k	19.8k	20.0k	18.7k
# target tokens	22.8k	19.2k	33.8k	13.4k

Table 2.4: Size of MTNT2019 test data.

In the shared task, improving NMT robustness was treated as a domain adaptation problem. Most participating teams used data augmentation methods such as back-translation, considering the limited size of noisy parallel data. Moreover, novel techniques were proposed regarding improving performance on the noisy test set.

Inline casing The submission from Naver Labs Europe [3] won the first title in Fr↔En and Ja→En language directions. Works on processing the training were done to improve the performance. Considering that the noisy corpus contains words with improper casings, they applied a "inline casing" technique by adding special tokens after each word to denote the word casing. For example, they used the token "<T>" to represent title case and "<U>" to represent uppercase. Examples of sentences processed with inline-casing are shown in Table 2.5. The inline casing method first recognizes the word case and adds a casing token after each word, and then it lowercases all words in the sentences. Note that this process was done together with the application of BPE. Since NMT models treat words with different casing as different entities, such an approach could reduce the vocabulary size and data sparsity of the model. With the introduction of word casing tokens, the model could learn and preserve the casing information in the translations.

Raw	**inline-casing**
They were so TASTY!!	they <T> were so <U> tasty <U> !!
MacDonalds	mac <T> donalds <T>

Table 2.5: Examples after processed with inline-casing [3]

Placeholder The placeholder method was applied to handle entities such as emojis, emoticons, and user names [3, 29]. This technique replaces all such tokens (e.g. emojis) in both source and target texts with a certain placeholder token. Therefore during training, the model learns a copy-and-paste strategy when it comes to

the placeholder tokens in the input. At the translation step, a postprocessing work needs to be done to replace the placeholders in outputs with the original tokens in the inputs. With this method, the model could learn the position of placeholder in the translation but without the need to recognize each special token in the raw input.

Data filtering The data filtering method is commonly used in the news translation task. In the WMT19 Robustness task, data filtering was also applied to clean the noisy parallel corpora. The filtering criteria used in the shared task is listed below [3, 29].

- Identical sentence pairs are removed.
- Sentence pairs with length ratio over threshold are removed (the threshold is set to be 1.8 in Berard et al.'s work [3]).
- Sentence pairs with words in other languages are removed.
- Sentence pairs with attention score (calculated with a baseline model) lower than threshold are removed.

This process could remove training examples with low quality and filter out unrelated noises (e.g. words in other languages). The data filtering could help improve model robustness on noisy texts as well, even for the baseline model that was not fine-tuned on noisy data.

Data Augmentation Various data augmentation methods were used to improve model performance, considering the limited number of noisy samples. Since the shared task provides monolingual data with both noisy and clean texts, Back-translation [32] was applied to utilized these data. Most participating teams used this data augmentation method. Berard et al. [3] back-translated both noisy and clean monolingual data in the target language. To acquire more back-translated noisy data, they back-translated MTNT monolingual data at each epoch during training. For out-of-domain monolingual data, they back-translated part of the corpus at each epoch.

Helcl et al. [15] used back-translation iteratively to improve model performance. They first trained a baseline model, which was used to back-translate monolingual data. Then the back-translated parallel data was combined with the parallel data from corpus to train a new model. The new model was again used to provide a precise version of pseudo parallel data. The back-translation took three iterations and could improve the lower bound of the baseline performance.

Zheng et al. [43] proposed back-translation methods to inject noise from clean monolingual data. They mentioned that in the MTNT corpus, source texts from end-user should be much noisier than the target human translations, although it still keeps some types of noise. Standard back-translation, which generates pseudo parallel data from target language corpus, might reverse the noise level and thus provides noisier target sentences. Considering this constraint, they fine-tuned the

baseline model (trained on clean data) with the MTNT training data in the opposite language direction. For example, the En→Fr data in MTNT corpus was fine-tuned on a Fr→En model. Using this method, the model could learn to inject noise in the translations.

While the clean monolingual data is large, most participating team focused on utilizing the noisy monolingual data. Murakami et al. [29] back-translated monolingual data in the MTNT corpus and applied a filtering process to remove pseudo parallel data that is too noisy. Zhou et al. [44] applied a source-target-source translation to generate a noisy version from the clean source texts.

Apart from back-translation, synthetic noise was applied to generate more noisy parallel data. Berard et al. [3] injected synthetic noise to MTNT-train, Common-Crawl and News Commentary dataset by replacing clean words with noisy variants. The noisy variants were made by rules, such as letter swapping, punctuation substitution, etc. The injection of rule-based synthetic noise showed a minor improvement on the noisy test.

Domain adaptation methods The domain adaptation was conducted mainly in two methods: fine-tuning on noisy data and domain-sensitive training with tags. Similar to the MTNT baseline, most systems were trained first on the clean out-of-domain data, and then fine-tuned on the noisy in-domain data [15, 31, 44]. To avoid overfitting on the noisy domain, a mixed fine-tuning on both noisy and clean data was applied [8, 29]. Considering the significantly different corpus size between clean and noisy data, the noisy data could be upsampled during the mixed fine-tuning. Others used a domain-sensitive training approach by adding a "domain" tag at the beginning of the source sentence to denote where this sentence sources from. For example, in the submission from Baidu-OSU [43], tags like "<clean_s>" and "<noisy_s>" were used to denote a noisy or clean sentence. Berard et al. [3] used double tags to denote both the dataset and noise level. Results showed improvement of the domain-sensitive training, compared to a simple mixed training.

External data One unconstrained system was submitted by FOKUS [13]. They used a baseline model trained under the setting of WMT19 Biomedical Translation Task[5]. Since the dataset for biomedical translation contains much more jargons and rare words than news translation dataset, using this external dataset could extend vocabulary coverage. Results showed that the baseline model for biomedical translation could outperform the MTNT baseline model by 2 to 4 BLEU score. This would suggest that using extra data with different types of noise could improve model robustness, even without a specific in-domain data.

The methods mentioned above handle the robustness problem well on both aspects of "preserving informal text" and "denoising noisy text". For example, the placeholder and inline-casing techniques target in keeping the special tokens and cas-

[5] `http://www.statmt.org/wmt19/biomedical-translation-task.html`

ing information in the translations. The data augmentation and domain adaptation methods are used to adapt the model on noisy domain so that it could handle the noisy texts. The submissions were evaluated by human judgements and the results are shown in Table 2.6 [25].

Systems	Human judgement scores (Rank)			
	En-Fr	Fr-En	En-Ja	Ja-EN
Constrained				
Baidu+OSU [43]	71.5 (2)	80.6 (3)	–	–
CMU [44]	–	58.2 (6)	–	–
CUNI [15]	66.3 (3)	82.0 (2)	–	–
FOKUS [13]	–	–	–	48.5 (5)
JHU [31]	–	76.3 (4)	58.5 (3)	65.4 (3)
NaverLabs [3]	**75.5 (1)**	**85.3 (1)**	63.9 (2)	**74.1 (1)**
NTT [29]	–	–	**66.5 (1)**	71.3 (2)
NICT [8]	–	–	44.7 (4)	49.1 (4)
Unconstrained				
FOKUS [13]	52.5 (4)	62.6 (5)	–	–

Table 2.6: Human judgement evaluation score for WMT19 Robustness Task. This table is extracted from [25].

The human judgement scores show a similar ranking as BLEU score. The NaverLabs's submission applied placeholders, inline-casing, back-translation and synthetic noise. With more augmented noisy data and sophisticated processing, they achieved the state-of-the-art on three language directions.

Chapter 3

Experiments

We conduct experiments on different sets of data to explore the effects in improving NMT model's robustness on Fr↔En language directions. 1) We first use a small size of training data, including the europarl and news commentary dataset. Different model architectures are trained and the performance of there models are compared on the noisy test set. 2) We propose new data augmentation methods to extend the limited noisy data. We experiment with back-translation (BT), forward-translation(FT), and fuzzy match (FM) data. Improvements of the augmented data are compared. 3) We explore the effect of data from speech transcripts. Both human transcripts and automatic speech recognition transcripts are used in this experiment. We show that introducing a different type of noise from speech transcripts could also benefit robustness. 4) Finally, we train models using techniques from WMT19 Robustness Task submissions, with our augmented data applied. We submitted our best-performing systems to the WMT19 Robustness Leaderboard, with both constrained and unconstrained systems. We compare our models with other systems in the shared task and show how our augmentation method could improve over the state-of-the-art systems.

3.1 Corpora

Data in the experiments can be divided into two categories: noisy and clean. The noisy parallel data is treated as in-domain data while the clean parallel data is out-of-domain. Following the criteria of WMT19 Robustness Task, we use datasets provided by the shared task as well as some external datasets.

As for datasets within the shared task constraint, we use all available corpora, both parallel and monolingual. For out-of-domain training, we used the WMT15 Fr↔En News Translation Task data[1], including Europarl v7, Common Crawl Corpus, UN Corpus, News Commentary v10 and Gigaword Corpus. In the following sections, we represent the combination of these corpora as "clean data". The MTNT dataset [28] is used as our in-domain data for fine-tuning. In terms of the monolingual data, we use noisy monolingual data in both English and French from the MTNT dataset. We

[1] http://www.statmt.org/wmt15/translation-task.html

only use the clean monolingual data provided by the shared task in French, which includes News Crawl data from 2008-2014 and News Discussion data. We do not use monolingual data in English because the size of the data is too large. Therefore, the system utilizing clean monolingual data is only in the En→Fr direction.

Under the unconstrained data setting, we experiment with external corpora, namely the IWSLT2017[2] and MuST-C[3] corpora[4], to explore the effect of informal spoken languages in human transcripts from speech. Since the MuST-C dataset provides the raw audio files, we use automatic speech recognition (ASR) system to generate automatic transcripts. The size of parallel and monolingual data is shown in Table 3.1.

Corpus	**Sentences**	**Words**	
		EN	**FR**
Gigaword	22.52M	575.67M	672.07M
UN Corpus	12.89M	316.22M	353.92M
Common Crawl	3.24M	70.73M	76.69M
Europarl	2.01M	50.26M	52.35M
News Commentary	200k	4.46M	5.19M
IWSLT	236k	4.16M	4.34M
MuST-C(en-fr)	275k	5.09M	5.30M
MTNT(en-fr)	36k	841k	965k
MTNT(fr-en)	19k	634k	661k
News Crawl 08-14(en)	41.99M	850.58M	–
News Discuss(en)	3.84M	66.17M	–
MTNT(en)	81k	3.41M	–
MTNT(fr)	26k	–	1.27M

Table 3.1: Size of parallel and monolingual training data. The last four rows are statistics for monolingua corpora.

We use the development set in MTNT and the *newsdiscussdev2015* for validation. Models with best performance on the validation set are evaluated on both noisy (MTNT and MTNT2019 test set) and clean (*newstest2014* and *newsdiscusstest2015*) test datasets. The validation and test data can be obtained from the prepared data in MTNT corpus[5]. Statistics of validation and test set are listed in Table 3.2.

For prepossessing, we tokenize the data with Moses tokenizer [21]. We apply Byte Pair Encoding (BPE) [33] with `subword-nmt`[6] tool to segment subwords. The BPE

[2] `https://wit3.fbk.eu/mt.php?release=2017-01-trnted`

[3] `https://ict.fbk.eu/must-c/`

[4] The data from IWSLT has same sentences for both directions, so we reversed the En→Fr data on the Fr→En direction. The MuST-C data is only used on En→Fr direction.

[5] `https://github.com/pmichel31415/mtnt/releases/download/v1.1/clean-data-en-fr.tar.gz`

[6] `https://github.com/rsennrich/subword-nmt`

Data	Sentences	Words	
		EN	FR
newsdiscussdev	1,500	24.2k	24.9k
MTNTdev(en-fr)	852	13.9k	15.8k
MTNTdev(fr-en)	886	32.9k	34.6k
newstest	3,003	62.3k	68.1k
newsdiscusstest	1,500	23.6k	25.1k
MTNTtest(en-fr)	1,020	15.9k	18.4k
MTNTtest(fr-en)	1,022	16.0k	16.6k
MTNT2019(en-fr)	1,401	19.9k	22.7k
MTNT2019(fr-en)	1,233	19.2k	19.8k

Table 3.2: Size of validation and test sets.

is learned from the combination of both clean and noisy training data. We experiment with a large vocabulary size to include noisy as well as clean subwords. The subword vocabulary size is set to 50k. In the first experiment, which uses a small training data, we set the subword vocabulary to 16k. Sentence after applying BPE would be a sequence of subwords, with a special delimiter "@@" denoting connection within words. For example, the word "phantom" is separated into subword form of "phan@@ tom". We exclude sentences with more than 70 subwords to avoid out-of-memory problem during training. We postprocess by removing the delimiter to restore word-level translations. Upon evaluation, we detokize our hypothesis translation files with Moses detokenizer. We use `multi-bleu-detok.perl`[7] to evaluate the BLEU score on the test sets.

3.2 Models

In the first experiment, we try with different model architectures, based on recurrent neural network (RNN), convolutional neural network (CNN) and Transformer. In the rest experiments, we fix our model to Transformer to achieve a better performance.

We follow the hyperparameters of the RNN model with attention in [28]. We use two layers of LSTM network for both encoder and decoder. The hidden layer size of the LSTM is 1024 and the embedding size is 512. We apply a copy bridge layer between the encoder and decoder. We use Adam as the optimizer and initiate the learning rate with 0.001. The batch size is 4096 tokens (subwords) per batch. To avoid overfitting, we use Dropout [36] of 0.3 and label smoothing [38] of 0.1. We apply a learning rate decay, with each epoch the learning rate halves. We validate and save checkpoints every 5000 training iterations. An early stopping technique is used to save training time, we stop the training once the perplexity on validation

[7] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu-detok.perl`

set no longer improve by 5 checkpoints. The checkpoint with lowest perplexity on validation set is evaluated on the test sets.

For convolutional model and transformer model, we simply adapt the hyperparameters from [12] and [40]. The batch size is 4096 tokens (subwords), which is the same as RNN-based model. Our models are trained with OpenNMT-py [20] on a single GTX 1080 Ti. The training stops until convergence on validation set (nearly 5 epochs). We use the Transformer model as our baseline in the following experiments.

As mentioned at the beginning of this chapter, we conduct experiments on different range of data. Besides, different fine-tuning methods are used. Therefore, to clarify the models and training data, we list the information of the models in Table 3.3.

Models	**Training data**	**Fine-tuning data**
RNN-s Conv-s Transformer-s	Europarl+News Commentary	MTNT
Baseline	WMT15 Fr↔En news translation data	—
Tune-S	—	MTNT, augmented data
Tune-R	—	MTNT (reverse direction)
Tune-B	—	MTNT (both direction), augmented data

Table 3.3: Notations for important models in the experiments. The first three models with a suffix "-s" are trained on a smaller range of clean data. The baseline is training on WMT15 news translation data in Fr↔En. The last three models are fine-tuned on the baseline model, with different fine-tuning options. The suffix "-S", "-R", and "-B" represent fine-tuning with MTNT parallel data on the same direction, reversed direction and both directions.

In our first experiment, we use a small range of training data, namely the Europarl and News Commentary Copora. The three models used are attached with as lowercased suffix "-s" to represent that they are trained on small clean data. In the rest three experiments, we use the whole training data provided by the shared task.

In our second experiment, we use data augmentation methods to improve robustness, under the constraint of the shared task. The baseline model, which is a standard Transformer model, is trained on the WMT15 Fr↔En news translation data (clean data). We list three models with different fine-tuning options. The "Tune-S" model fine-tunes the baseline model with MTNT parallel data on the same direction (e.g. fine-tuning Fr→En data on Fr→En model). The "Tune-R" model fine-tunes the baseline model with reversed direction data (e.g. fine-tuning Fr→En data on En→Fr model). The "Tune-B" model fine-tunes with MTNT parallel data on both language directions (e.g. fine-tuning Fr↔En on Fr→En model).

In the third experiment, we explore the use of parallel transcripts from audios. We use both human transcripts and ASR generated transcripts and show the effect of injecting noise from spoken language.

Finally, we build models using techniques proposed in the shared task submissions. We introduce our data augmentation methods and the extra speech transcript data and compare with the state-of-the-art systems. We experiment with both fine-tuning on noisy data and mixed training with "domain" tags indicating where the sentences source from. We used different tags for clean data, MTNT parallel data(same and reversed directions), forward translation, back translation, ASR data and fuzzy match data and human transcripts parallel data (IWSLT and MuST-C). Tags are added at the beginning of each source sentences.

3.3 Data Augmentation Methods

3.3.1 Back-translation

Back translation [32] is a popular technique for utilizing monolingual data in neural machine translation. The common practice is to build a target-to-source model and translate the target language monolingual data. The synthetic parallel data from back-translation is combined with the raw parallel data to train a new model. This technique is shown to be effective in training on clean texts while it might not benefit performance of robustness. Consider the two aspects of improving robustness, back-translation on noisy monolingual data could only improve the data amount while it does not fulfill the task of "Denoising noisy text". Since the noisy monolingual texts contain a large proportion of noises, after reversing the direction it would inject noises into the translation. This might disobeys the "noise gap" between source and target texts and model trained with such data might produce noisy output.

Therefore, we could only use back-translation on clean monolingual data. Following Zheng et al.'s approach [43], we train the Tune-R model by fine-tuning the baseline model on MTNT data in reversed direction. The reversed fine-tuning would teach the model to generate noises in the translations. An example is shown in Figure 3.1. We fine-tune the baseline Fr-En model with reversed source and target texts in MTNT En-Fr data. By doing so, the Tune-R model is able to generate noisy translations from clean input texts. Then we translate the French monolingual data into English with Tune-R model. Therefore, we can obtain a pseudo parallel data that has noises on the English side. The parallel data generated with this method would keep the "noise gap" and can be used to train En-Fr models.

3.3.2 Forward-translation

Albeit back-translation can be used to utilize clean monolingual data, we would like to make use of the noisy monolingual data, which is ideal for improving robustness. The MTNT corpus provides noisy monolingual data from the same domain as the parallel data. We propose a forward-translation method to generate synthetic parallel data from noisy monolingual data. As shown in Figure 3.2, we fine-tune the baseline model on merged MTNT parallel data in both directions, resulting in the

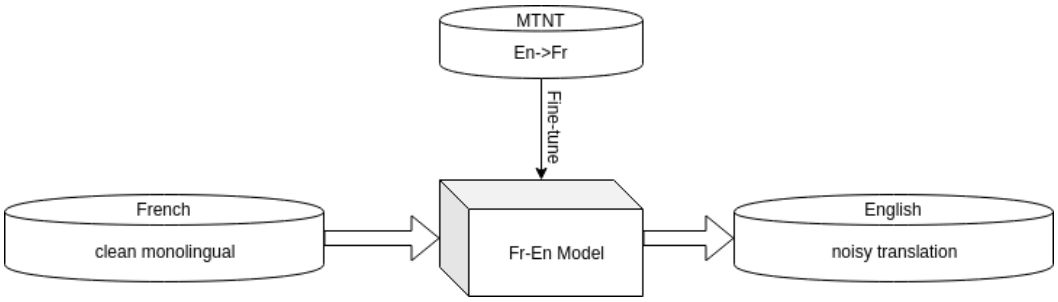

Figure 3.1: Example of the back-translation on clean monolingual data.

Tune-B model. Noisy monolingual data in MTNT corpus is translated into the target language but with fewer noises. The parallel data from FT can be used to fine-tune the same Tune-B model. Since we use merged noisy data on both directions, the model could learn information in the opposite direction, and thus it will not overfits the translation generated by itself.

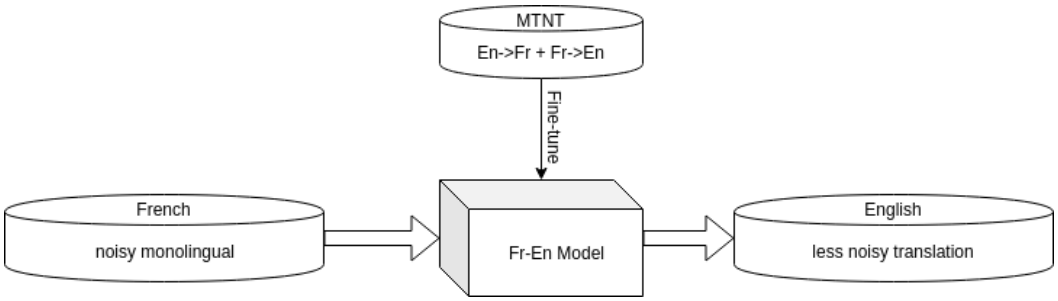

Figure 3.2: Example of the forward-translation on noisy monolingual data.

3.3.3 Fuzzy match

We adapt the method to augment data from parallel corpus from [6]. The original method aims to find similar source sentences to those in the parallel corpus (S, T) using a monolingual corpus, and then reuse the translation of the source sentences as translations for the similar source sentences found. We adapt this method and use it to expand the small noisy corpus. For each source sentence $s_i \in S$ in the training set, all other source sentences $s_j \in S(s_i \neq s_j)$ are compared with this sentence by measuring string similarity $Sim(s_i, s_j)$. If the similarity of the two sentences is above a threshold λ, the two sentences are mapped to each other's corresponding target sentence and the two new sentence pairs $(s_i, t_j), (s_j, t_i)$ are added into our augmented training data. The similarity is measured with Levenshtein distance [24] on token level. The similarity score is calculated as the edit distance divided by the minimum length of the two sentences (as shown in Equation 3.1). The pseudo-code for finding fuzzy matches is shown in Algorithm 1.

$$Sim(S_i, S_j) = \frac{editdistance(s_i, s_j)}{min(len(s_i), len(s_j))} \tag{3.1}$$

In addition to finding fuzzy matches in the parallel corpus, we experiment with the monolingual corpus by mapping sentence m_i in monolingual corpus to its fuzzy match's target sentence t_j (If $Sim(m_i, s_j) > \lambda$, we add a new sentence pair (m_i, t_j) to the training augmented data). The pseudo code finding fuzzy matches from extra monolingual data is shown in Algorithm 2.

To boost the speed of finding matches, we follow the approaches in [6] and use a Python library `SetSimilaritySearch`[8] to select similar candidates before calculating edit distance. For each source sentence, only the top 10 similar candidates are selected to calculate the edit distance score.

Algorithm 1: Find fuzzy matches from parallel corpus

Input : Parallel data (S, T), threshold λ
Output: Fuzzy match parallel data (S', T')
Function `find_parallel`(S, T, λ)**:**
  $candidate\_pairs \leftarrow$ `SetSimilaritySearch`(S, λ) ;
  foreach $(s_i, s_j) \in candidate\_pairs$ **do**
    $similarity \leftarrow$ `editdistance`$(s_i, s_j) \div$ `min`(`len`(s_i), `len`(s_j)) ;
    if $similarity > \lambda$ **then**
      add$(s_i, t_j), (s_j, t_i)$**to**(S', T') ;
    end if
  end foreach
  return (S', T')

Algorithm 2: Find fuzzy matches from parallel corpus and a monolingual corpus

Input : Parallel data (S, T), monolingual data M, threshold λ
Output: Fuzzy match parallel data (S', T')
Function `find_parallel`(S, T, M, λ)**:**
  foreach $m_i \in M$ **do**
    $candidates \leftarrow$ `SetSimilaritySearch`(m_i, S, λ) ;
    foreach $s_j \in candidates$ **do**
      $similarity \leftarrow$ `editdistance`(m_i, s_j) ;
      if $similarity > \lambda$ **then**
        add(m_i, t_j)**to**(S', T') ;
      end if
    end foreach
  end foreach
  return (S', T')

Considering the large size of our clean data, we only find noisy matches within

[8] `https://github.com/ekzhu/SetSimilaritySearch`

MTNT parallel and monolingual data. The parallel data in both directions are merged to get more combinations. We also notice that the MTNT training data contains duplicated source sentences, and we filter out the same sentences when finding fuzzy matches to avoid duplicated data generation. With the threshold setting to 0.5, we have 7,290 new sentence pairs on En→Fr and 7,154 on Fr→En.

3.3.4 Automatic speech recognition

Since the MuST-C dataset provides audio files, we use ASR system to transcribe the audio files, so that noises related to pronunciation could be created. We use the Google Speech-to-Text API[9] and transcribe 2,461 audios of TED talks in the MuST-C dataset. Looking into the ASR transcripts, we find that the ASR system tends to skip some sentences due to the fast speaking speed. Therefore we filter the ASR data based on the length ratio of the human transcripts and the ASR transcripts. We measured the noise level of ASR transcripts by evaluating Word Error Rate (WER) and Word Recognition Rate (WRR) with `asr_evaluation`[10] library, comparing to the human transcripts in MuST-C. As can be seen from the table 3.4, with a more strict filtering policy, the quality of ASR transcripts produce less error and achieve a higher recognition rate.

Data	**WER**	**WRR**
ASR($\lambda = 1.5$)	36.41%	65.38%
ASR($\lambda = 1.2$)	31.70%	70.54%

Table 3.4: Noise level of ASR generated data filtered with different length ratios.

[9] `https://cloud.google.com/speech-to-text/`
[10] `https://github.com/belambert/asr-evaluation`

Chapter 4

Results and Analysis

In this chapter, We will show results of the experiments and give analyses on these results. In the first section, different models trained on small range of data are compared. In the second section, we show the effects of our data augmentation methods, including back-translation, forward-translation and fuzzy match methods. In the third section, we will evaluate the use of external data from parallel speech transcripts. We compare the ASR data, IWSLT dataset and MuST-C dataset and show the improvement from these data. In Section 4.5, we build models using techniques from the shared task. We apply our methods and compare our systems with the state-of-the-art systems. In the last section, we list some translation samples and conduct qualitative analysis.

4.1 Fine-tuning on Noisy Text

We compared the robustness of RNN, CNN and Transformer models trained on small range of clean data. We evaluated the BLEU score of these models perfomance on MTNT, MTNT2019, *newstest2014* and *newsdiscusstest2015* test sets. We conducted this experiment in Fr→En direction. We fine-tuned these models with the MTNT training data and the results are shown in Table 4.1.

Models	**MTNT**	**MTNT2019**	**newstest**	**newsdiscusstest**
RNN-s	22.54	25.47	29.2	29.95
RNN-s(tune)	34.50(**+11.96**)	35.52(+10.05)	29.23(+0.03)	32.84(+2.89)
Conv-s	24.49	26.77	28.96	30.02
Conv-s(tune)	31.98(+7.49)	35.38(+8.61)	30.67(**+1.71**)	34.37(+4.35)
Transformer-s	25.12	27.41	30.67	30.6
Transformer-s(tune)	**36.52**(+11.40)	**37.52**(**+10.11**)	**31.6**(+0.93)	**34.89**(**+4.39**)

Table 4.1: BLEU scores of Fr→En models performance on clean and noisy test sets. We report BLEU score of each model as well as the improvement after fine-tuning on noisy training data.

From the table we can see that Transformer model outperforms the other two on both noisy and clean test set. While the CNN model shows better result on MTNT and MTNT2019 testsets than the RNN model, they show a similar performance on

clean test sets. After fine-tuning, the Transformer model still achieves the highest BLEU score on all four test sets. This would suggest that the model performance might be consistent when evaluated on noisy and clean test sets. Therefore, the better model evaluated on clean data would achieve a better result after fine-tuned on noisy data.

From the comparison of models before and after fine-tuning on MTNT training data, we can see that all models show an improvement in terms of robustness to noisy texts. The fine-tuning results in an increase of 7-12 BLEU score on noisy test sets. Meanwhile, the models' performance on the clean test sets improve as well. The introduction of domain adaptation on noisy texts improves the BLEU score on `newsdiscusstest` more than `newstest`, because languages in the former test set is less formal hence it might benefit more from the fine-tuning.

We found that the improvements of RNN and Transformer model exceed the CNN model, evaluating on MTNT and MTNT2019 test sets. However, on the two clean test sets, the CNN model could improve more than the other two after the fine-tuning. We also found that different models take different epochs to finish the fine-tuning. As stated in Table 4.2, the RNN model takes 54 epochs to converge on noisy data while the CNN model takes 25 epochs. It is worth mentioning that the Transformer quickly converges after 6 epochs, which is only 100 training iterations. Although the learning rate might have influence to the convergence speed since we continued the learning rate during fine-tuning, it still shows the need of more parallel data to feed big models such as Transformer.

Model	**Epochs**
RNN-s(tune)	54
Conv-s(tune)	25
Transformer-s(tune)	6

Table 4.2: Numbers of epoch of different models to converge on noisy data.

4.2 Data Augmentation

Since the size of parallel noisy data is small, we used various data augmentation methods to generate synthetic noisy data. We used the Transformer model as our baseline for better performance, and the clean training data was expanded to all WMT15 Fr↔En news translation data. In this experiment, the use of data for training and fine-tuning was under the constraints of WMT19 Robustness Task.

4.2.1 Back Translation

We generated parallel data using back-translation on both clean and monolingual data. The data from clean texts was used for training the baseline while the data

from noisy texts was used for fine-tuning. We followed Zheng et al.'s [43] method and used the Tune-R model to generate back-translation from clean texts. Regarding the MTNT monolingual data, we used the target-to-source Tune-S model to produce back-translations. The results are shown in Table 4.3.

Models	**MTNT**	**MTNT2019**	**newstest**	**newsdiscusstest**
		fr-en		
Baseline	34.41	36.14	**36.51**	34.43
Tune-S	**40.16**	**41.58**	35.94	**36.75**
+BT(noisy)	37.93	39.19	34.03	34.00
		en-fr		
Baseline	30.12	29.57	35.51	35.93
Tune-S	36.15	32.25	36.77	37.16
+BT(noisy)	35.61	31.01	36.28	36.96
Baseline+BT(clean)	—	—	37.38	37.58
+MTNT	**37.51**	**32.66**	**39.03**	**39.32**

Table 4.3: BLEU score for models with back-translation. We show the effect of both back-translation data from clean and noisy monolingual data.

Using a large range of training data, the baseline model could already achieve a competitive performance. The Tuns-S model, which is fine-tuned on MTNT parallel data, outperforms the baseline by +6 and +3 BLEU score on MTNT and MTNT2019 test sets. However, the introduction of BT data from noisy texts does not improve model performance. In both directions, after combining noisy BT data with MTNT data, the performance drops compared to the Tune-S model. This might be caused by the imbalanced noises in the BT data. Unlike the MTNT parallel data, the noisy BT data has noises in the target sentences, which would teach the model to output noisy translations. Therefore, back-translation on noisy monolingual data might not help improve robustness.

In terms of clean back-translation data, from the table (see the last two rows) we can see that the performance of the baseline is improved on the clean test sets. The performance of the noisy tests was not measured since this model was trained with the "domain" tags. Therefore, without training on the noisy tag, this model could not translate the noisy test sets. However, after fine-tuned on the MTNT data, the model performance improves to a large extent, outperforming the Tune-S model. The introduction of clean back-translation data could improve the potential of the baseline model and achieve a better performance with noisy data injected.

4.2.2 Forward Translation

Since the MTNT data is limited in each direction, we tried merging the MTNT data in both directions to utilize more noisy parallel data. We name the model fine-tuned on the merged data as Tune-B. We used the Tune-B model to generate forward-translation data, which is used to fine-tune the same model. We report BLEU score

of models utilizing the FT data in Table 4.4.

Models	MTNT	MTNT2019	newstest	newsdiscusstest
		fr-en		
Tune-S	**40.16**	41.58	**35.94**	**36.75**
+FT	39.28	40.82	35.47	36.32
Tune-B	38.25	40.12	35.36	35.15
+FT	39.38	**41.82**	35.56	35.00
		en-fr		
Tune-S	**36.15**	**32.25**	**36.77**	37.16
+FT	36.11	31.36	36.16	37.77
Tune-B	35.64	31.27	36.67	37.21
+FT	35.98	31.37	36.3	**37.91**

Table 4.4: BLEU score for models with forward-translation from MTNT monolingual data.

Comparing the Tune-S and Tune-B models, we found that the former performs better. This is because the merged data includes sentence pairs in the opposite direction, which might be noisier on the target side. Using data in the same direction could better improve robustness than using merged data. However, after we added forward-translation data for fine-tuning, we found that the performance of Tune-S model decreases while the Tune-B model improves. In Fr→En direction, the Tune-B model with FT data outperforms Tune-S model on MTNT2019 set.

The forward-translation data was generated using the Tune-B model, which includes information in the opposite direction, and this might benefit forward-translation and prevent the model from overfitting itself. However, the Tune-S model, if fine-tuned together with FT data, would simply overfit the data in the same direction since it does not contain opposite direction information.

Although the Tune-B model with FT data still lags behind the Tune-S model marginally, it could utilize the noisy monolingual data. Compared to noisy parallel data, the monolingual data is much easier to obtain. Therefore, if provided with a large volume of noisy monolingual data[1], the Tune-B model might benefit more from FT data. Thus we believe that the proposal of forward-translation is still beneficial.

4.2.3 Fuzzy Match

We used the algorithms mentioned in section 3.3.3 to generate fuzzy match data from MTNT parallel and monolingual corpus. As did in BT and FT experiments, we compared the Tune-S and Tune-B models with extra fuzzy match data. The results in BLEU score are shown in Table 4.5.

[1]The size of noisy monolingual data in MTNT corpus is only 3-4 times of the parallel data

Models	**MTNT**	**MTNT2019**	**newstest**	**newsdiscusstest**
		fr-en		
Tune-S	40.16	**41.58**	35.94	36.75
+FM	**40.18**	41.55	**35.95**	**36.78**
Tune-B	38.25	40.12	35.36	35.15
+FM	38.91	40.85	35.62	34.58
		en-fr		
Tune-S	36.15	**32.25**	36.77	37.16
+FM	**36.24**	32.07	**36.84**	**37.34**
Tune-B	35.64	31.27	36.67	37.21
+FM	36.03	31.37	36.58	37.13

Table 4.5: BLEU score for models with fuzzy match data.

The performance of Tune-S model does not change too much after introducing the FM data. In Fr→En direction, the BLEU score of Tune-S model with FM data almost stays the same while in En→Fr direction, the FM data improves BLEU scores slightly except on MTNT2019 test set. Regarding the Tune-B model, the FM data shows a positive effect on noisy data. In both directions, the Tune-B model with FM data improves slightly on MTNT and MTNT2019 test sets. However, this would also sacrifice the performance on clean test sets to a small extent. It is worth mentioning that although the FM data could benefit the Tune-B model, it still lags behind the Tune-S model without any extra data.

4.2.4 Augmented data combination

Although the FM and FT data could benefit the Tune-B model, the model with these augmented data does not outperform the Tune-S model, which only fine-tunes the baseline on MTNT parallel data. Therefore, we took a combination of the FM and FT data as well as the merged MTNT parallel data to see the effect on robustness. We report the BLEU scores in Table 4.6.
We noticed that the combination of FM and FT data improves robustness in both directions. The combination of FT and FM data could benefit model performance more than using only one of them. From the table, we can see that the Tune-B model with FM and FT combined achieves similar performance as the Tune-S model. In Fr→En direction, it could even outperform the Tune-S model on the MTNT2019 test set. The Tune-B model, although lags behind the counterpart Tune-S model, could utilize more extra data and improve with these data. The use of noisy parallel data with merged direction could increase the model's potential to learn from extra data, which might not be in the same domain.

Considering that the Tune-B model with augmented data does not outperform the Tune-S model to a large extent, we hypothesize it is because the model is not adapted to the specific noisy domain. Since the introduction of the opposite di-

Models	MTNT	MTNT2019	newstest	newsdiscusstest
		fr-en		
Tune-S	40.16	41.58	35.94	36.75
+FM	40.18	41.55	35.95	**36.78**
Tune-B	38.25	40.12	35.36	35.15
+FM+FT	40.13	**42.80**	35.82	35.88
+double tune	**40.57**	42.55	**36.06**	36.53
		en-fr		
Tune-S	36.15	**32.25**	36.77	37.16
+FM	36.24	32.07	**36.84**	37.34
Tune-B	35.64	31.27	36.67	37.21
+FM+FT	36.21	31.25	36.37	37.76
+double tune	**36.78**	32.10	36.72	**37.84**

Table 4.6: BLEU score for models with a combination of the augmented noisy data. For comparison, we only list Tune-S model with FM data added, because the FT data might hinder the performance.

rection decreases the Tune-B model initially, we conducted a second fine-tuning step to compensate for this loss. For example, the Tune-B model is first fine-tuned with MTNT(merged), FM and FT data, and then it is fine-tuned with the MTNT data in the corresponding direction. This step would compensate for the loss from the use of merged MTNT data. The models with the double fine-tuning outperform the others on the MTNT test set and achieve similar performance on MTNT2019 as the Tune-S model. Besides, this step improves model performance on clean test sets as well. The double fine-tuning increases by +0.24 and +0.65 BLEU score in Fr→En direction, and +0.35, +0.08 in the opposite direction.

4.3 External Data

To explore the effect of other types of noise, we fine-tuned our baseline model on different external datasets (See Table 4.7). We used the parallel text data in IWSLT and MuST-C corpora. These data were generated from human transcripts of audio files and corresponding translations to the transcripts. Besides, since the MuST-C dataset provides the raw audio files, we used ASR systems to generate transcripts that might contain noises in the data. As for the translations, we used the corresponding sentences in human translations from the MuST-C data.

By fine-tuning only on IWSLT data, the BLEU score (Fr→En) on MTNT2019 increases by +2.14 over the baseline, while the results on the other three test sets decrease. In the En→Fr direction, fine-tuning on IWSLT improves the model performance on all four test sets, and with MTNT data added, the BLEU score on noisy data shows an additional increase on the noisy test sets. The results suggest that the introduction of the external dataset could help to improve robustness. Even without using the in-domain data, by fine-tuning only on IWSLT data, the model performance on noisy

Models	**MTNT**	**MTNT2019**	**newstest**	**newsdiscusstest**
		fr-en		
baseline	34.41	36.14	**36.51**	**34.43**
tune-IWSLT	34.22	38.28	35.96	32.95
tune-S+IWSLT	37.52	**41.22**	36.33	34.07
tune-B+IWSLT	**37.84**	41.12	36.03	34.02
		en-fr		
baseline	30.12	29.57	35.51	35.93
tune-IWSLT	33.66	30.58	36.89	37.34
tune-S+IWSLT	35.44	31.24	36.73	37.90
tune-B+IWSLT	**35.47**	**31.27**	**36.76**	**38.03**
tune-ASR($\lambda = 1.5$)	30.53	28.66	36.46	35.61
tune-ASR($\lambda = 1.2$)	31.09	29.48	36.59	35.54
tune-MuSTC	34.15	31.09	36.27	37.61

Table 4.7: BLEU score for models with external datasets from speech transcripts. We used both human transcripts and ASR generated transcripts in this experiment.

test sets would also improve. Similar trend shows with the MuST-C data, with an increase on all four test sets. The use of datasets from speech transcript could improve model robustness on noisy texts because noises related to spoken languages are injected. The benefit of speech transcripts might come from informal languages such as slangs, speaking languages and domain-related words. Apart from this, we also kept the indicating words (e.g. "[laughter]" and "[applause]") in the transcripts, which could also play a role of noise.

When using ASR data generated from the audio files in MuST-C dataset, we found that the ASR system tended to skip some phrases due to the fast speaking speed. Some sentences in the ASR transcripts were broken. Therefore, we did a filtering work to clean the broken sentences out. The filtering criteria are based on the sentence length ratio of human transcripts to the ASR transcripts. We first filtered ASR data by removing sentences where the original transcript length is over 1.5 times that of the ASR transcript. The model fine-tuned on ASR data shows a slight decrease on MTNT2019 and `newsdiscuss`. Further removing more broken sentences, we reduced the length ratio threshold to 1.2, and with the strict filtering policy, the model achieves similar performance as the baseline model. Evaluated on *newstest*, the ASR-tuned model improves +1.08 BLEU score over the baseline. Although the ASR data could increase robustness compared to the baseline, it might inject more noises and thus corrupt the fine-tuning data. Better ASR system might be needed to provide more accurate transcripts.

4.4 Domain-sensitive training

Simply concatenating all data and training in one step might not be an ideal approach since this might mix data from different domains. To deal with this, domain

tags were added to distinguish sentences from different domains. The step of adding domain tags is named as "Domain-sensitive training" (or "Domain-aware training"). In this section, we experimented with a domain-sensitive training in Fr→En direction.

We added different tags to represent domains. The "<clean_s>" tag is used to represent clean sentences. The "<MTNT_s>" is used to represent sentences from MTNT parallel data in the corresponding direction while the "<MTNT_rev>" tag represents sentences from MTNT data but in the opposite direction. In this case, the model could utilize noisy parallel data in both directions but without experiencing performance loss like the Tune-B model. The "<IWSLT_s>" tag is used to denote sentences from the IWSLT dataset. Data generated with forward-translation is added with a "<FT_s>" tag while sentences generated with back-translation on noisy monolingual data is added with a "<BT_s>" tag. We also included fuzzy match data and used "<FM_s>" to denote fuzzy match data. We tagged the fuzzy match data in the opposite direction with "<FM_rev>".

We conducted a leave-one-out experiment to compare the effect of data from different domains. We first fine-tuned the baseline model with data from all domains. For each time, we removed data with a certain tag and fine-tuned the baseline model with the rest. If the performance decreases after leaving a certain tag, this data is considered as beneficial. The results are shown in Table 4.8.

Removed tag	**MTNT**	**MTNT2019**	**newstest**	**newsdiscusstest**
None	40.20	42.67	34.21	32.17
<MTNT_s>	37.79(**-2.41**)	41.66(**-1.01**)	34.37(+0.16)	32.20(+0.03)
<MTNT_rev>	39.94(-0.26)	42.44(-0.23)	34.59(+0.38)	33.17(**+1.00**)
<FT_s>	39.64(-0.56)	41.85(-0.82)	34.72(**+0.51**)	33.03(+0.86)
<BT_s>	40.34(+0.14)	42.62(-0.05)	34.17(-0.04)	32.53(+0.36)
<FM_s>	39.98(-0.22)	42.87(+0.20)	33.77(-0.44)	32.15(-0.02)
<FM_rev>	40.42(**+0.22**)	43.10(**+0.43**)	34.20(-0.01)	32.57(+0.40)
<IWSLT_s>	40.03(-0.17)	42.75(+0.08)	32.64(**-1.57**)	30.55(**-1.62**)

Table 4.8: Results of the leave-one-out experiment in Fr→En. Each time we remove a certain type of data and see the effect of it.

We first fine-tuned the baseline model with all domain tags, except the "<clean_s>" itself. The model shows a good result with all data combined. However, we also found that the "domain-sensitive training" might help the model in translating noisy texts but drags the performance on clean texts down. Compared to previous results on clean test sets (e.g. in Table 4.6), the BLEU score of model trained with domain tags decreases by around 2 on both `newstest` and `newsdiscusstest`. This is because with the domain tags the model treats clean training data and noisy data as different types of data. Thus the benefit from noisy texts, such as the MTNT training data, might not help to optimize the parameters that matter in translating clean texts.

The removal of "<MTNT_s>" tag results in a sharp decrease in the noisy test set performance. The BLEU score drops by -2.41 on MTNT test set and -1.01 on MTNT test set. However, the BLEU score on both clean test sets increases marginally. Contrary to what we found in previous experiments, the noisy parallel data might not help to improve model performance on clean texts, if the model is trained with the domain tags. Similar trend appears after removing the "<MTNT_rev>" tag when we excluded the MTNT data in the opposite language direction. Performance on noisy test sets drops while the BLUE score increases on clean test sets. This suggests that when using "domain-sensitive training", the noisy data plays a role in helping to improve noisy text performance but corrupting translation on clean texts. It is worth noticing that the harm from noisy data in the opposite direction is more than that in the same direction. This is due to the fact that the reversed data might break the "noise gap", meaning that it does not follows the task of "Denoising noisy text". Thus the injection of noises in target sentences would corrupt the translations.

Removing the forward-translation data causes the BLEU score to decrease on noisy texts but increase on clean texts. This result is similar to what we found in section 4.2.2, and again confirms the positive effect of FT data in improving robustness to noisy data. Similar to previous findings, the BT data does not help in robustness since the removal of "<BT_s>" tag does not change the performance too much.

The removal of fuzzy match data results in decreases on MTNT and `newstest` sets. However, the performance on MTNT2019 test set improves by +0.2 BLEU score. We hypothesize that this is caused by the slight difference between the MTNT and MTNT2019 test sets. The fuzzy match data is sourced from the MTNT training data and therefore could benefit MTNT test set performance but not guaranteed to improve that on MTNT2019. We also tried with fuzzy match data in the opposite direction, but we found that the opposite FM data shows no help in neither noisy nor clean texts. The fuzzy match data could be noisy since it allows similar sentences to swap translations. Reversing the direction would further increase the noises and therefore harm the robustness.

IWSLT data plays an important role in clean texts performance. Removing it results in decreases of -1.57 and -1.62 BLEU scores on `newstest` and `newsdiscusstest` sets. Among all other types of data, removing IWSLT data causes the most decreases in BLEU score on clean test sets. In terms of the noisy test sets, the IWSLT data could help marginally on the MTNT test set.

Compared to simple mixed training, the "Domain-sensitive training" could absorb more noisy data and thus improve model robustness to noises. However, as a sacrifice, this method might hinder the performance on clean texts. Models trained with this method could hardly benefit from the noisy data when they are evaluated on clean texts.

4.5 WMT19 Leaderboard

We submitted our best performing systems in previous sections to the WMT19 leaderboard and compared with the state-of-the-art systems. We submitted both constrained and unconstrained systems in both language directions. The systems in Fr→En are the models with "Domain-sensitive training" in section 4.4. In En→En direction, we submitted the model trained with back-translation from clean monolingual data, as shown in section 4.2.1. To achieve a better performance, we applied the inline-casing technique [3] in the En→Fr systems, which were trained with domain tags as well. We included both IWSLT and MuST-C data when training the unconstrained system in En→Fr direction. The information of the four submissions are shown in Table 4.9.

Direction	Tag	Inline-casing	Train	Fine-tune
fr-en	yes	no	WMT15	MTNT(both)+FM+FT
fr-en	yes	no	WMT15	MTNT(both)+FM+FT+IWSLT
en-fr	yes	yes	WMT15+BT(clean)	MTNT(both)+FM+FT
en-fr	yes	yes	WMT15+BT(clean)	MTNT(both)+FM+FT+IWSLT+MuST-C

Table 4.9: Infomation of our submissions to WMT19 Leaderboard. We submit both constrained and unconstrained systems on both directions. In en-fr direction, we applied clean text back-translation and inline-casing.

We list the WMT19 Robustness Leaderboard ranking in Table 4.10 and Table 4.11. Our System could achieve the 4th position in Fr→En direction, improving by +17.6 BLEU score over the MTNT paper baseline model. In En→Fr direction, our system achieves the 3rd place in the leaderboard. The state-of-the-art systems are based on the Transformer-big model while limited by GPU resource, our systems are built on the standard Transformer model. This might account for the difference with the SOTA results. The unconstrained systems, with IWSLT and MuST-C data added, improves over the constrained system by only +0.1 BLEU score. Chances are that the external datasets could help more when translating clean texts, as stated in the previous section.

System	BLEU-uncased
Berard et al.[3]	48.8
Helcl et al.[15]	45.8
Zheng et al.[43]	44.5
Ours(Unconstrained)	43.9
Ours(Constrained)	43.8
Post et al.[31]	41.8
Zhou et al.[44]	36.0
Grozea et al.[13]	30.8
MTNT paper baseline	26.2

Table 4.10: WMT19 Robustness Leaderboard on Fr→En.

System	BLEU-uncased
Berard et al.[3]	42.0
Helcl et al.[15]	39.1
Ours(Unconstrained)	38.1
Ours(Constrained)	38.0
Zheng et al.[43]	37.0
Grozea et al.[13]	24.8
MTNT paper baseline	22.5

Table 4.11: WMT19 Robustness Leaderboard on En→Fr.

4.6 Translation Samples

Now looking back to the example in Table 2.1, we compared the translation of our system to the Google translation result (see in Table 4.12). Similar to Google Translate, our baseline system copies the word “o ’tain” to the output and misinterprets the rest part of the sentence as well. We listed the translation of the “Domain-sensitive training” model in Fr→En. It can be seen from the table that after the domain adaptation on noisy texts, the translation could be more precise. Firstly, the second half of the source sentence could be translated correctly. Moreover, the word “o ’tain” is translated with a synonym word but in a stronger emotion, though it contains profanity. This would suggest that the domain adaptation on noisy texts could help the model to recognize rare words and thus avoid corrupting the translation sentence.

src	Aaaaaaaah.... 8 ans aprs, je viens de percuter.... :o ’tain je me disais bien que je passais ct d’un truc vu les upvotes.
ref	Aaaaaaaah.... 8 years later, I’ve just realized.... :o damn I had the feeling that I was missing something considering the upvotes.
Google	Aaaaaaaah .... 8 years later, I just hit ....: o ’tain I told myself that I was next to something seen the upvotes.
Baseline	Aaaaaaaaah.... 8 years later, I just hit....: o ’tain I was saying well that I was passing next to a trick in view of the upvotes.
Fine-tuned	Aaaaaaaaah.... 8 years later, I just hit...: o ’fuck I was saying that I was missing something given the upvotes.

Table 4.12: A translation example from a noisy French sentence. Our domain adapted model shows an improvement over the baseline and Google translate.

We show the attention map of the last decoder layer to the input in Figure 4.1. In the attention map, a brighter color represents more attention is paid to the input token. As the beginning of the sentence, the attention map shows a good alignment between the source and target tokens. The continuous “a” tokens could be attended to the corresponding tokens in the source sentence. It is also worth noticing that the source token “tain”, receives a large proportion of attention from target tokens “o”, “&apos”(a symbol of punctuation) and “f@@”(the double @ is BPE delimiter). This confirms the importance of this special token in the source sentence and could account for why the translations from baseline and google might be messy after mistranslating this “tain” token. Another interesting phenomenon from the attention map is that the full stop punctuation in the source sentence is attached with much importance by many target tokens. This is contrary to intuition since the punctuation is not supposed to be attached with such importance during translation.

Table 4.13 illustrates another example of noisy sentence. The source sentence contains some terminologies such as “Valeur Actuelle Nette” and “Net Presente Value”. Due to the interfere of these jargons, the baseline translation would ignore the first half of the sentence. Nevertheless, the fine-tuned model could translate the whole

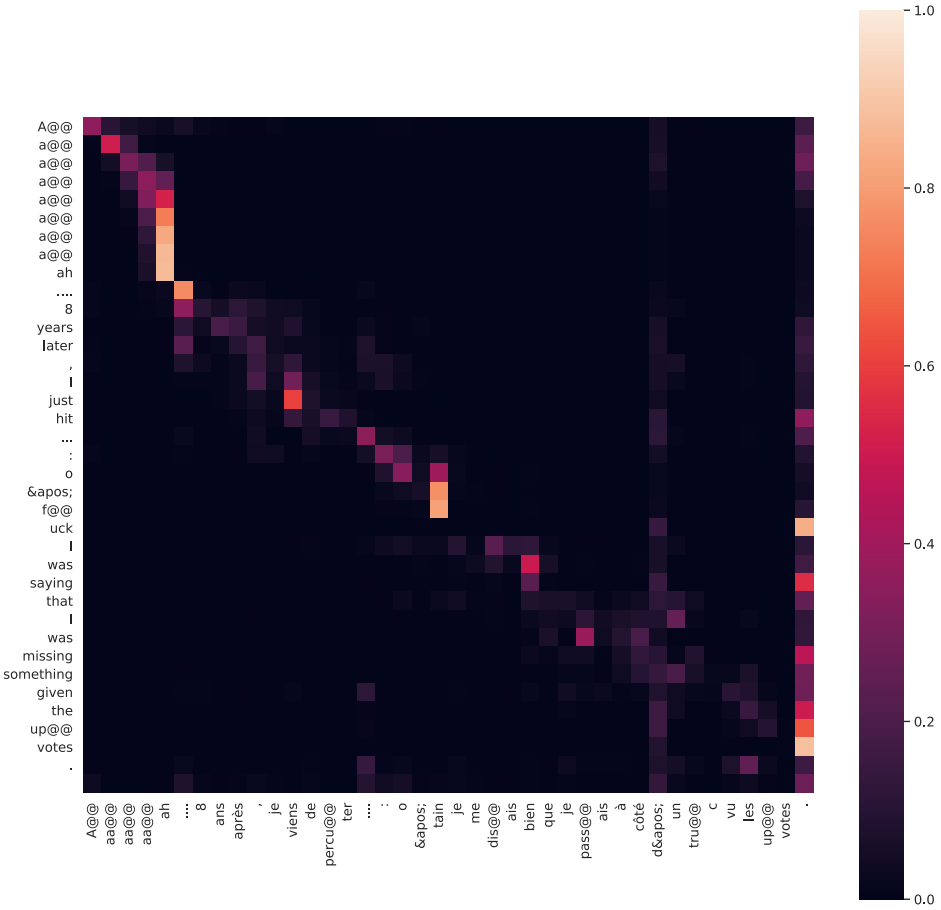

Figure 4.1: Attention map of the translation example.

sentence correctly. Although there is a difference of the word "Valeur Actuelle Nette" between the reference and model translation, it is still acceptable since this word means present net value. Besides, the context of this sentence is explaining a French jargon, and thus human translation keeps the "Valeur Actuelle Nette" in French. However, the model could not understand and might translate the words "Valuer" and "Actuelle" into English.

src	>> On calcule la Valeur Actuelle Nette (VAN), en anglais on l'appelle la Net Present Value (NPV).
ref	>> You calculate the Valeur Actuelle Nette (VAN), in English it's called the Net Present Value (NPV).
Baseline	>> The Net Present Value (NPV) is calculated.
Fine-tuned	>> You calculate the Current Value Nette (VAN) in English we call it the Net Present Value (NPV).

Table 4.13: An example of jargon/terminology in noisy sentence.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we detailedly described the topic of robustness in neural machine translation and explored the effect of different data augmentation methods. We conducted experiments under the setting of WMT19 Robustness task in Fr↔En directions. Our contributions can be summed as follows:

- We explored different techniques to improve robustness of NMT models. We compared mixed fine-tuning on noisy data with Domain-sensitive training and analysed the strength and weakness of both methods.
- We used data augmentation methods to extend limited noisy parallel data and showed the effect of these augmented data. In the experiments, we tried with back-translation, forward-translation and fuzzy match methods. We proposed to use parallel data from speech transcripts and found that it could improve model robustness, especially on clean texts. We first proposed to use automatic speech recognition to generate noisy data from audios and showed both positive and negative effects of this type of data.
- We submitted our best-performing systems to the WMT19 Leaderboard. We used techniques proposed in the shared task, such as inline-casing and back-translation on clean monolingual data. Our submissions could achieve competitive positions compared to the state-of-the-art systems. Although our proposed systems can not outperform the SOTA result, they are trained with smaller models and less data, which might be more efficient for training.

5.2 Future work

This thesis mainly focused on the data aspect since currently noisy parallel data is very limited. However, the data size is guaranteed to increase in the future. Therefore, more work could be done to improve robustness of neural machine translation.

Model Architecture One possible solution is to propose better model architectures

that could perform well on noisy texts. Currently, character level model might ease the problem of sentence perturbation, but only limited to this type of noise. To deal with all types of noses, as well as fullfill the two task of improving robustness, better model architectures should be proposed.

Better data augmentation methods Although we showed the effect of various data augmentation methods, more augmentation methods are demanded. Generative adversarial network might be a possible direction, and ideally the model could learn from the noisy data and generate similar noisy parallel sentences. However, the issue with discontinuity of NLP might still be a problem to be resolved. Another direction might focus on unsupervised neural machine translation since noisy monolingual data is much easier to be sourced. The unsupervised NMT model could possible benefit from the large volume of noisy monolingual data in both source and target languages.

Transfer learning from other tasks Many other tasks in NLP also involves data with noisy inputs. For example, the input sequence of Error Correction contains spelling and grammatical errors; the input of sentiment analysis could be sourced from social media and therefore contains noises as well. Transfer learning might be a way to utilize the robust models in other NLP tasks and apply to machine translation. Besides, it is also a good method to conduct multi-task learning and benefit robustness of machine translation model from other tasks.

To sum up, the robustness problem is an important task in natural language process, not only limited to machine translation. How to adapt the state-of-the-art models from clean texts to noisy texts is a promising topic, and more research is needed to propel the development of NLP.

Appendix A

Legal and Ethic Considerations

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		

continued on next page

continued from previous page

Does your project involve developing countries?	✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?	✓
Could the situation in the country put the individuals taking part in the project at risk?	✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY	
Does your project involve the use of elements that may cause harm to the environment, animals or plants?	✓
Does your project deal with endangered fauna and/or flora /protected areas?	✓
Does your project involve the use of elements that may cause harm to humans, including project staff?	✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?	✓
Section 8: DUAL USE	
Does your project have the potential for military applications?	✓
Does your project have an exclusive civilian application focus?	✓
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?	✓
Does your project affect current standards in military ethics e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?	✓
Section 9: MISUSE	
Does your project have the potential for malevolent/criminal/terrorist abuse?	✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?	✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?	✓
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?	✓
SECTION 10: LEGAL ISSUES	

continued on next page

continued from previous page

Will your project use or produce software for which there are copyright licensing implications?	✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?	✓
SECTION 11: OTHER ETHICS ISSUES	
Are there any other ethics issues that should be taken into consideration?	✓

This project does not involve human participants nor animals. Human embryos or cells are not involved in this project. The datasets in the experiments are only used for research and all datasets and corpora are cited with reference. The data does not contain privacy information. This project focuses on improving robustness of neural machine translation, so it is general and does not involve developing contries. The data is in form of texts so this project would not do harm to the environment and human body. The topic of this project is in machine translation and thus it does not have the potential of misuse. All softwares and tools used in this project are open-sourced and for non-business use, and they have been cited in this thesis. Overall, this project is legal and does not have potential ethic issues.

Bibliography

[1] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. pages 3, 6, 7, 8, 15

[2] Belinkov, Y. and Bisk, Y. (2018). Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. pages 3, 4, 13, 17

[3] Berard, A., Calapodescu, I., and Roux, C. (2019). Naver labs europe's systems for the wmt19 machine translation robustness task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 526–532, Florence, Italy. Association for Computational Linguistics. pages 4, 19, 20, 21, 22, 40

[4] Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational linguistics*, 16(2). pages 2

[5] Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311. pages 2

[6] Bulte, B. and Tezcan, A. (2019). Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809, Florence, Italy. Association for Computational Linguistics. pages 5, 13, 28, 29

[7] Cettolo, M., Girardi, C., and Federico, M. (2012). Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy. pages 5, 17

[8] Dabre, R. and Sumita, E. (2019). Nict's supervised neural machine translation systems for the wmt19 translation robustness task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 533–536, Florence, Italy. Association for Computational Linguistics. pages 4, 21, 22

[9] Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org. pages 10

[10] Davis, M. (2003). Aoccdrnig to a rscheearch at cmabrigde uinervtisy. pages 13, 17

[11] Di Gangi, M. A., Cattoni, R., Bentivogli, L., Negri, M., and Turchi, M. (2019). MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics. pages 5

[12] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org. pages 2, 3, 6, 9, 10, 26

[13] Grozea, C. (2019). System description: The submission of fokus to the wmt 19 robustness task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 537–538, Florence, Italy. Association for Computational Linguistics. pages 21, 22, 40

[14] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. pages 10

[15] Helcl, J., Libovick, J., and Popel, M. (2019). Cuni system for the wmt19 robustness task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 539–543, Florence, Italy. Association for Computational Linguistics. pages 4, 20, 21, 22, 40

[16] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. pages 3, 7

[17] Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics. pages 15

[18] Karpukhin, V., Levy, O., Eisenstein, J., and Ghazvininejad, M. (2019). Training on synthetic noise improves robustness to natural noise in machine translation. *arXiv preprint arXiv:1902.01509*. pages 18

[19] Khayrallah, H. and Koehn, P. (2018). On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics. pages 17

[20] Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*. pages 26

[21] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics. pages 24

[22] Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics. pages 14

[23] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics. pages 2, 6

[24] Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Dokl.*, 10:707–710. pages 13, 28

[25] Li, X., Michel, P., Anastasopoulos, A., Belinkov, Y., Durrani, N., Firat, O., Koehn, P., Neubig, G., Pino, J., and Sajjad, H. (2019). Findings of the first shared task on machine translation robustness. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 91–102, Florence, Italy. Association for Computational Linguistics. pages 4, 18, 22

[26] Luong, T., Pham, H., and Manning, C. D. (2015a). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics. pages 9

[27] Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015b). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics. pages 16

[28] Michel, P. and Neubig, G. (2018). MTNT: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 543–553, Brussels, Belgium. Association for Computational Linguistics. pages 4, 14, 15, 23, 25

[29] Murakami, S., Morishita, M., Hirao, T., and Nagata, M. (2019). Ntts machine translation systems for wmt19 robustness task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages

544–551, Florence, Italy. Association for Computational Linguistics. pages 4, 19, 20, 21, 22

[30] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. pages 4, 16

[31] Post, M. and Duh, K. (2019). Jhu 2019 robustness task system description. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 552–558, Florence, Italy. Association for Computational Linguistics. pages 4, 21, 22, 40

[32] Sennrich, R., Haddow, B., and Birch, A. (2016a). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics. pages 12, 13, 18, 20, 27

[33] Sennrich, R., Haddow, B., and Birch, A. (2016b). 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)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics. pages 16, 24

[34] Simons, G. F. and Fennig, C. D., editors (2018). *Ethnologue: Languages of the World*. SIL International, Dallas, TX, USA, twenty-first edition. pages 1

[35] Sperber, M., Niehues, J., and Waibel, A. (2017). Toward robust neural machine translation for noisy input sequences. In *International Workshop on Spoken Language Translation (IWSLT)*. pages 18

[36] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958. pages 25

[37] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112. pages 2, 6, 15

[38] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826. pages 25

[39] Vaibhav, V., Singh, S., Stewart, C., and Neubig, G. (2019). Improving robustness of machine translation with synthetic noise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages

1916–1920, Minneapolis, Minnesota. Association for Computational Linguistics. pages 18

[40] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. pages 2, 3, 6, 11, 12, 26

[41] Weaver, W. (1955). Translation. *Machine translation of languages*, 14:15–23. pages 2

[42] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., ukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144. pages 2

[43] Zheng, R., Liu, H., Ma, M., Zheng, B., and Huang, L. (2019). Robust machine translation with domain sensitive pseudo-sources: Baidu-osu wmt19 mt robustness shared task system report. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 559–564, Florence, Italy. Association for Computational Linguistics. pages 4, 20, 21, 22, 27, 33, 40

[44] Zhou, S., Zeng, X., Zhou, Y., Anastasopoulos, A., and Neubig, G. (2019). Improving robustness of neural machine translation with multi-task learning. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 565–571, Florence, Italy. Association for Computational Linguistics. pages 21, 22, 40