Sentence and Word Weighting for Neural Machine Translation Domain Adaptation

Pinzhen (Patrick) Chen

Undergraduate Dissertation
Artificial Intelligence and Software Engineering
School of Informatics
The University of Edinburgh

2019

Abstract

Neural machine translation today has achieved state-of-the-art performance for machine translation, yet it is facing the problem of domain mismatch due to scarce data. Recently researchers have proposed many techniques to tackle this problem, including fine-tuning, building multi-domain system, sentence weighting, etc. In this project, we try to improve sentence weighting, and propose a novel *tag-and-weight* technique which uses sentence weighting when building a multi-domain system. Then, we argue that sentence weighting is a trivial case of weighting words. Thus, we move one step further to propose word level weighting for neural machine translation domain adaptation, based on word frequencies and language model scores. We evaluate our approaches using Romanian to English and English to German translation tasks with different domain specificity. Experiments show that our proposed approaches achieve improved performance. The top-performing tag-and-weight achieves on average 0.6 BLEU increase comparing to current state-of-the-art techniques. Finally, our in-depth analysis indicates that our proposed approaches are able to recall more named entities in a domain, and that tag-and-weight has a strong domain differentiating capability.

Acknowledgement

I would like to thank my supervisor Dr Kenneth Heafield for proposing and guiding me through the project, without whom I would not develop my interests in Neural Machine Translation. I also must thank him for letting me run experiments using Azure credits donated by Microsoft to the Alan Turing Institute. Finally, I want to thank my family back in China and my friends for their continuous love and encouragement.

Declaration

I declare that this dissertation was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that the work has not been submitted for any other degree or professional qualification except as specified.

Pinzhen Chen

Table of Contents

1 | Introduction

Machine translation is a sub-field of Natural Language Processing (NLP), where translation from one language to another is done automatically by a computer. The first idea was to convert words or phrases one by one using a dictionary or rules, with linguistic features taken into consideration. This is called rule-based machine translation (Nirenburg, 1989). Later, a similar example-based approach was invented, which finds template sentences in available translation examples and only deals with the new words in new sentences (Nagao, 1984). Then in 1990, machine translation entered the era of statistical machine translation (SMT), where translations are produced by a probabilistic model, parameterised on bilingual and monolingual corpora containing millions of sentences (Koehn et al., 2007). The statistical model produces a translation with the maximum probability given a sentence. Breaking down to word level, SMT models the probability of translating a word to another, and that of a word given previous words. Such probabilities in an SMT can be learned at various levels such as words, phrases, syntactic units and context-free grammars.

Since 2013, the research on machine translation has been shifting towards **neural machine translation** (NMT) using neural networks (Kalchbrenner and Blunsom, 2013; Cho et al., 2014b), and recently it has achieved the most promising performance compared to other approaches mentioned above (Bojar et al., 2016, 2017). Unlike SMT, NMT uses deep neural network trained on bilingual parallel corpora to model the translation probabilities. It is shown that NMT produces vastly more fluent results than SMT (Skadina and Pinnis, 2017).

An obvious trend we observe from the history of machine translation is that the amount of data required to build a system is growing exponentially, from rules to some template translations, and now to huge parallel corpora of millions of sentences. However, it is hard to find a large corpus in the exact same domain as we want to use the system, due to the scarcity of high-quality corpora (Chu and Wang, 2018).

As a result, one problem NMT suffers from is domain mismatch, which arises when a machine translation model learns from data in a certain domain and translates texts

in another domain (Koehn and Knowles, 2017). Several reasons can account for this problem. First, words and phrases can have different meanings in different domains (polysemy and homonymy), such as "zip", "run" and "cat" seen in daily life and computer science terms. Another reason is that texts in different domains can be in different writing styles, leading to different spelling, choice of word, word order, sentence length, etc. Also, a new domain can have words that are never seen by the model, which thus can nearly never be produced. We present in Table 1.1 a case of using online translators to translate statistics terminology "二元分类 (binary classification)" from Chinese to English. Mainstream generic translators like Google, Bing and Baidu gave results of various quality[1] due to domain mismatch.

Source	二元分类
Reference	*binary classification*
Baidu	two element classification
Bing	binary categories
Google	binary classification

Table 1.1: Online results for translating statistics terminology from Chinese to English

From the results, we see that all online translators were able to produce fluent results that convey the semantics of our original term. "Binary" essentially means something involving "two element(s)" and "category" is a synonym to "class(ification)". However, Bing and Baidu could not produce our desired terminology. A translation that only conveys the same semantics is not excellent or professional, if a domain-specific writing style is expected in certain scenarios, like academic writing, legal documents, medical prescriptions and technology patents, just to name a few.

To alleviate the above problem, many researchers have worked on adapting NMT to a specific domain from both data and model perspectives (Chu and Wang, 2018). Methods from data perspective include building a multi-domain system and controlling output domain (Sennrich et al., 2016a), synthesising domain specific parallel corpora from monolingual text (Sennrich et al., 2016c; Park et al., 2017), as well as data selection and cut-off (Wang et al., 2017a; van der Wees et al., 2017).

At model level, successful techniques include weighting objective function (cost) at sentence level (Chen et al., 2017; Wang et al., 2017b; Zhang and Xiong, 2018), combining domain specific language model and NMT system (Gulcehre et al., 2015; Domhan and Hieber, 2017), ensembling systems trained on different domains (Freitag and Al-Onaizan, 2016), neural lattice search (Khayrallah et al., 2017), fine tuning (Luong and Manning, 2015) and mixed fine tuning (Chu et al., 2017).

[1] Translations were done on 18 Mar 2019 on the three online platforms.

Following a comprehensive review of relevant approaches, we make four contributions to the NMT domain adaptation research, detailed as follows:

1. We reproduce and try to improve sentence weighting (Wang et al., 2017b) by increasing domain discrimination effect.
2. We propose a novel tag-and-weight approach, that uses source side domain tags as well as applies sentence weighting. The approach combines sentence weighting (Wang et al., 2017b) and multi-domain NMT (Sennrich et al., 2016a), and achieves substantial improvement of 0.6 and 0.85 BLEU over sentence weighting and multi-domain respectively on English to German translation.
3. We explore three word-level weighting schemes, based on word frequency and language model scores. One approach using a logarithmic frequency ratio achieves 0.6 BLEU higher than sentence weighting on Romanian to English.
4. We design another new sentence weighting by summing up the ratio of logarithmic word frequency. It outperforms sentence weighting by 0.2 BLEU on English to German and 0.8 BLEU on Romanian to English.

To the best of our knowledge, currently there is no established literature on word weighting for neural machine translation domain adaptation. The motivation for our work on word weighting is simple yet elegant. In a neural network, weighting sentences is a trivial case of weighting words because sentence loss is computed as the sum of word losses. Word weighting will lead to a weighted sum being calculated.

The rest of the dissertation starts with a background introduction in Chapter 2, where we introduce the technical background of neural machine translation and related work on domain adaptation. Then, in Chapter 3 we improve sentence weighting and propose a new domain adaptation technique. In Chapter 4, we state our motivation and propose three novel word weighting schemes based on word frequency and language model scores. In the following Chapter 5, we describe our experiments on two datasets, as well as present experiment results. We continue to analyse and find reasons behind the results in Chapter 6. Finally, in Chapter 7 we summarise our work and provide future work directions.

2 | Background

In this chapter, we describe the technical background of neural machine translation and language modelling, from mathematics to architectures. After that, we introduce the data and evaluation for machine translation. Finally, the later sections present the problem of domain mismatch and solutions related to our work.

2.1 Recurrent Neural Networks

Neural machine translation is based on Deep Learning in **neural networks**, which are formed with input, hidden and output layers of neurons. The neurons can be regarded as nodes with differentiable non-linear functions (e.g. logistic), connected to other neurons (nodes) in adjacent layers. Each neuron outputs the value calculated from its function, applied on the weighted sum of values from the previous layer's neurons (or directly from input), shown in Equation 2.1 in matrix notation:

$$y_{W,b}(x) = f(W^T x + b) \tag{2.1}$$

where y denotes the output, f denotes an activation function, W denotes weights, x denotes inputs and b denotes a bias term. The weights are initialised using a certain scheme (e.g. random or all-zero). The output from a neuron becomes the input of its connected neurons in the next layer. Hence as a whole, a neural network model is able to distort and propagate the input through its internal layers to produce an output. The process is called forward propagation and the output is the value we want, such as a word or a label.

To find the ideal weights and bias terms (parameters) for a neural network, backpropagation is used (Rumelhart et al., 1986; LeCun et al., 2012). First, a cost (error) is computed using some objective (cost) function (e.g. cross-entropy or mean squared error) on the output and expected values. Next, error derivatives with respect to each

parameter in the model are calculated. Then the parameters are increased by the product of error derivatives and a constant called learning rate. Forward propagation and backpropagation are done for many iterations on training data until a certain criterion is met. The two processes are illustrated in Figure 2.1.

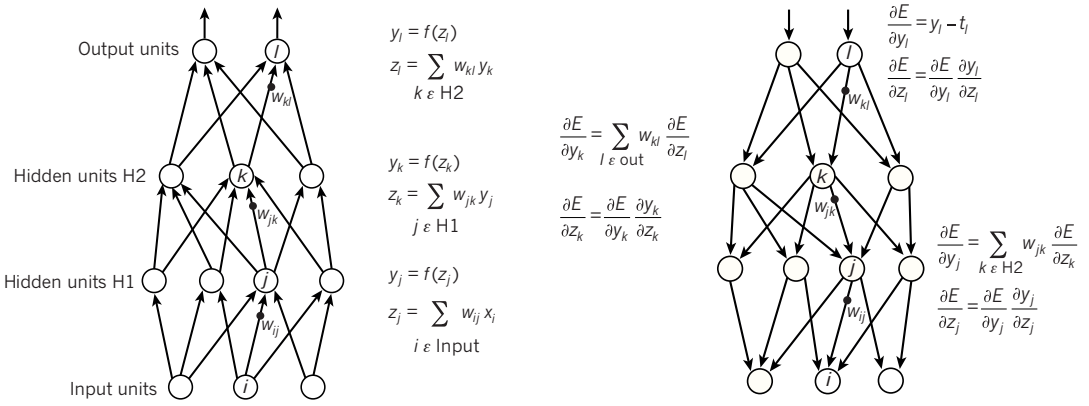

Figure 2.1: Illustrations of forward propagation (left) and backpropagation (right) by LeCun et al. (2015), where nodes are neurons, edges are connections, w are weights, z are weighted sums, f are activation functions, y are outputs, t is expected output and E is error/cost.

An **Recurrent neural network (RNN)** is a more powerful architecture. It reads in an element at a time, together with its hidden states at the previous time step. It can be visualised in Figure 2.2 as copies of the same neural network, passing down states (history) through time. Hence an RNN handles sequential data better because it remembers history (LeCun et al., 2015). Its output in matrix notation is shown in Equation 2.2, where h_{t-1} is the hidden state at time step $t-1$ (history), W's are corresponding weights and σ is the softmax function which turns numbers into probabilities that sum to 1. An RNN model is trained using backpropagation through time (Mozer, 1995), where error derivatives are backpropagated through history.

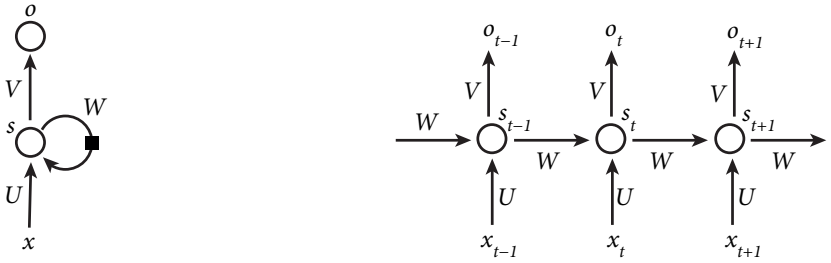

Figure 2.2: Illustrations of RNN (left) and unfolded RNN (right) by LeCun et al. (2015). x_i, o_i and s_i are the input, output and hidden states at time i, and W is history from previous state.

$$
\begin{aligned}
h_t &= f(W_h h_{t-1} + W_x x_t + b) \\
y_t &= \sigma(W_s h_t)
\end{aligned}
\tag{2.2}
$$

Vanilla RNN can suffer from vanishing gradient problem, that error derivatives tend to 0 after computation through many layers, so long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) or its variants (e.g. gated recurrent unit (Cho et al., 2014a)) are commonly used to solve the problem. An LSTM cell, shown in Figure 2.3, has three gates, namely input, forget and output gates. An input gate determines what information to keep, a forget gate determines what information to discard, and an output gate determines what information to pass forward.

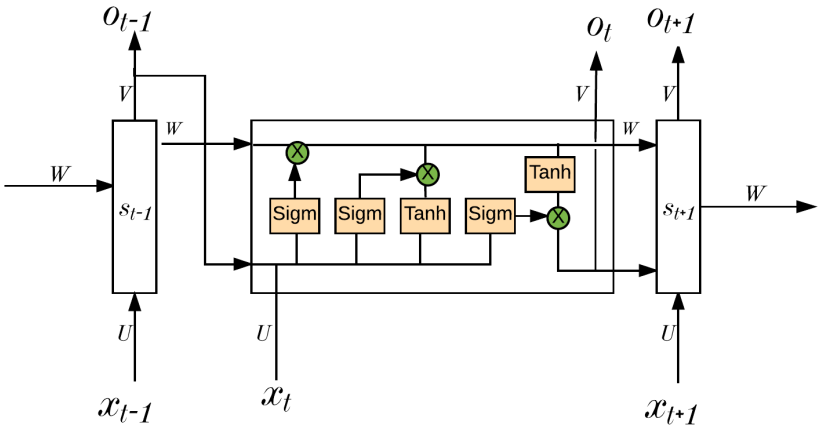

Figure 2.3: Illustration of an LSTM cell by Pawar (2017), where from left to right, the three green nodes are input, forget and output gates.

For a natural language task, a neural network typically takes a word at a time as input. As a result, input sentences are tokenised (split on space for English, for instance). Then the resulted tokens are represented as vectors called **word embeddings** (Bengio et al., 2003), usually in one-hot encoding. This is created by converting all tokens with total vocabulary size N into binary vectors of size $1 \times N$ with only one position being 1 to indicate a particular token.

2.2 Language Modelling

Besides neural networks, language models play a significant role in our project too. First, sentence weighting uses statistical language models to score training sentences in order to distinguish their domains and assign them weights (Wang et al., 2017b). Also, neural machine translation itself can be seen as a special type of neural language

model, that models the probability of a target translation given a source sentence. In this section, we introduce both statistical and neural language models, as well as two closely related measurements, entropy and perplexity.

2.2.1 *N*-gram Statistical Language Model

Nowadays ***n*-gram language model** is the most commonly used statistical language model (Chen and Goodman, 1996). In the area of probability and NLP, n-gram is defined as n consecutive items. An example of n-grams resulted from a sentence is shown in Table 2.1.

sentence	"how are you"
unigram	"how", "are", "you"
bigram	"how are", "are you"
trigram	"how are you"

Table 2.1: An example of n-grams from sentence "How are you"

An n-gram language model represents sequences of words (languages) using n-gram probabilities. It makes use of Markov assumption, that a word only depends on the $n-1$ words before it, to produce a simplified estimation of the language. For instance, a unigram (1-gram) means that the words are independent of each other, and a trigram (3-gram) means that a word depends on two previous words. Let w_i denote the i$^{\text{th}}$ word and w_i^k denote the sequence of $w_i, w_{i+1}, ..., w_{k-1}, w_k$, the probability of a given sentence $S = w_1, w_2, ..., w_n$ of length n can be calculated as Equation 2.3:

$$\begin{aligned} P(S) &= P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)...P(w_n|w_1,w_2,...w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1^{i-1}) \\ &\approx \prod_{i=1}^{n} P(w_i|w_{i-n+1}^{i-1}) \end{aligned} \tag{2.3}$$

N-gram probabilities are estimated from the language using Maximum Likelihood Estimation (MLE) (Jurafsky and Martin, 2000), which is a ratio of the observed frequency of an n-gram to the observed frequency of its prefix. Hence the previous Equation 2.3 can be further developed as shown in Equation 2.4:

$$
\begin{aligned}
P(S) = \ldots &\approx \prod_{i=1}^{n} P(w_i | w_{i-n+1}^{i-1}) \\
&\approx \prod_{i=1}^{n} P_{MLE}(w_i | w_{i-n+1}^{i-1}) \\
&= \prod_{i=1}^{n} \frac{C(w_{i-n+1}^{i-1}, w_i)}{C(w_{i-n+1}^{i-1})}
\end{aligned}
\tag{2.4}
$$

One problem of using MLE on a corpus with a finite size is that it yields zero probability for unseen events. Thus, smoothing, which assigns a small probability to unseen data, is applied in language models. Smoothing algorithms include additive smoothing, Good-Turing estimate, interpolation, backoff, and Kneser-Ney smoothing, ranging from simple to advanced algorithms (Chen and Goodman, 1996).

In practice, beginning-of-sentence symbols (BOS, denoted by "<BOS>" or "<s>") and end-of-sentence symbols (EOS, denoted by "<EOS>" or "</s>") are added to sentences to indicate the start and the end boundaries. Their probabilities are modelled like normal words in both language models and NMT systems.

2.2.2 Neural Language Model

An RNN can also model natural language (sequences of words) (Bengio et al., 2003; Mikolov et al., 2010). During training, an RNN takes each sequence without the last word as input, and the corresponding expected output is the same sequence shifted one time step to the left, thus without the first word. Hence a neural language model is trained to calculate the probabilities of next possible words given all previous words. An example of using RNN to model sentence "A Time Machine by H. G." can be visualised in Figure 2.4.

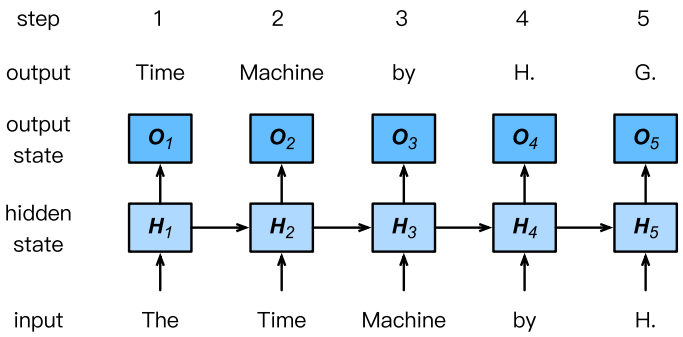

Figure 2.4: An example of neural language model by Zhang et al. (2019).

2.2.3 Cross-entropy and Perplexity

Language models can model a corpus and estimate the probability of a given sentence occurring in that corpus, besides which there are two more important metrics. The first is **cross-entropy**, which measures uncertainty of a given piece of text occurring in the corpus. A higher cross-entropy indicates a higher uncertainty, and vice versa. Intuitively, a longer sentence will have a larger uncertainty. Therefore, in our project we use **per-word cross-entropy** H (Brown et al., 1992), which is cross-entropy normalised by sentence length. It is defined as Equation 2.5 for a sentence S with a *large* length n.

$$H(S) = -\frac{1}{n}\log P(S) \tag{2.5}$$

Another metric **perplexity** shown in Equation 2.6 is defined based on cross-entropy. It reflects how much the sentence is "expected" compared to what the language model has observed in the corpus (Gamon et al., 2005). **Per-word perplexity** of a sentence indicates the average number of choices (of words) at each word position, assuming a uniform distribution (Jelinek et al., 1977). Both metrics provide good insight of how close the sentence is to the corpus.

$$\text{perplexity}(S) = 2^{H(S)} \tag{2.6}$$

2.3 Neural Machine Translation

Currently, there are two prevailing neural machine translation architectures, namely encoder-decoder (or sequence-to-sequence, seq2seq) with attention mechanism and Transformer model based solely on attention. In this project, we base all of our experiments on encoder-decoder with attention.

2.3.1 Encoder-decoder Model

An encoder-decoder model (Cho et al., 2014b) can be thought of as two RNN language models connected together. The encoder takes an inputs and represents the source sentence, and the decoder models the target sentence and outputs a translation. These are visualised in Figure 2.5 where encoder is in green and decoder is in blue. Suppose that $X = \{x_1, x_2, ..., x_n\} = x_1^n$ and $Y = \{y_1, y_2, ..., y_m\} = y_1^m$ are a pair of source and

target sentences of lengths n and m respectively. An RNN encoder is trained to read source sentence X into a representation vector c of fixed length as Equation 2.7:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}) \\ c &= q(\{h_1, h_2, ..., h_n\}) \end{aligned} \tag{2.7}$$

where $h_t \in \mathbb{R}$ is the hidden state at time step t, c is calculated from all hidden states after the input is read, and f and q are non-linear functions. Take the case of Sutskever et al. (2014), f is the LSTM described in Section 2.1 and $q(\{h_1, h_2, ..., h_n\})$ simply returns h_n.

The decoder is trained to translate X to Y, in other words, to recursively predict target side word y_t at time t, given the representation vector c of source and all previously generated words $\{y_1, y_2, ..., y_{t-1}\}$. It is the same as a neural language model on the target side, except that it takes the source language into consideration too. The translation probability is modelled and calculated as Equation 2.8, where g is a non-linear and likely multi-layered function. At time step t it calculates the probability of y_t, with s_t being the hidden state of the decoder.:

$$\begin{aligned} P(Y) = P(\{y_1, y_2, ..., y_m\}) &= \prod_{t=1}^{m} P(y_t | c, \{y_1, y_2, ..., y_{t-1}\}) \\ &= \prod_{t=1}^{m} g(y_{t-1}, s_t, c) \end{aligned} \tag{2.8}$$

2.3.2 Attention Mechanism

The previous encoder-decoder model has a bottleneck, that the fixed length vector c is poor at modelling long sentences (Sutskever et al., 2014), so the **attention mechanism** came to rescue (Bahdanau et al., 2015; Luong et al., 2015). When translating a target word, humans will take extra care of the specific source word(s) that are relevant to it. The attention mechanism, which is added between encoder and decoder, produces an analogous effect. An encoder-decoder model with attention is presented in Figure 2.5, where the attention layer is denoted by the orange blocks.

The added attentional layer computes a score of each encoder hidden state for each decoder hidden state, by reading from all encoder and decoder states. This score can

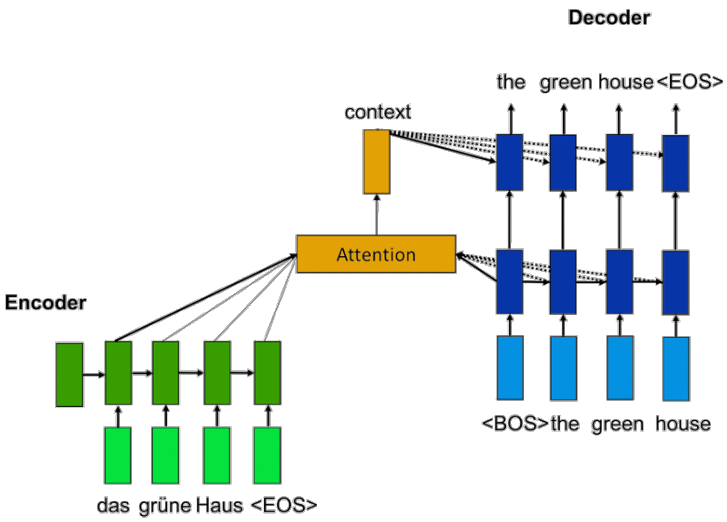

Figure 2.5: Illustration of attention by Hieber and Domhan (2017).

be computed in many different ways shown in Equation 2.9:

$$score(h_t, h_s) = \begin{cases} h_t^T W_a h_s, \text{where } W_a \text{ is a weight matrix} & \text{(Luong et al., 2015)} \\ h_t^T h_s & \text{(Luong et al., 2015)} \\ \frac{h_t^T h_s}{\sqrt{n}}, \text{where } n \text{ is source states dimension} & \text{(Vaswani et al., 2017)} \\ v_a^T \tanh\left(W_a[h_t; h_s]\right) & \text{(Bahdanau et al., 2015)} \\ \cos\left(h_t, h_s\right) & \text{(Graves et al., 2014)} \\ \ldots \end{cases} \tag{2.9}$$

Next, for each target word, an attentional (context) vector is obtained as an average of encoder hidden states, weighted by the softmax-ed attention scores. The calculations are presented in Equation 2.10, where c_t is the context vector, and $\alpha_{t,i}$ is the alignment score of target word and the i^{th} source word, at time t. Through such weighting, the connections between source and target words are not affected by their distances, so source words that are more relevant can have a higher influence directly on the target word being produced. Finally, at each time step, the attentional context vector is combined with decoder states to generate target words.

$$\alpha_{t,i} = \sigma(score(h_t, h_s))$$
$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i \tag{2.10}$$

The set of attention weights ($\alpha_{t,i}$) can be interpreted as how important the i^{th} source word is to the target word being generated at time step t. Hence, it provides an insight of word alignment (although sometimes it can be totally off). An example of attention weights of source and target word pairs is presented in Figure 2.6, and word alignment from it can be visualised in Figure 2.7.

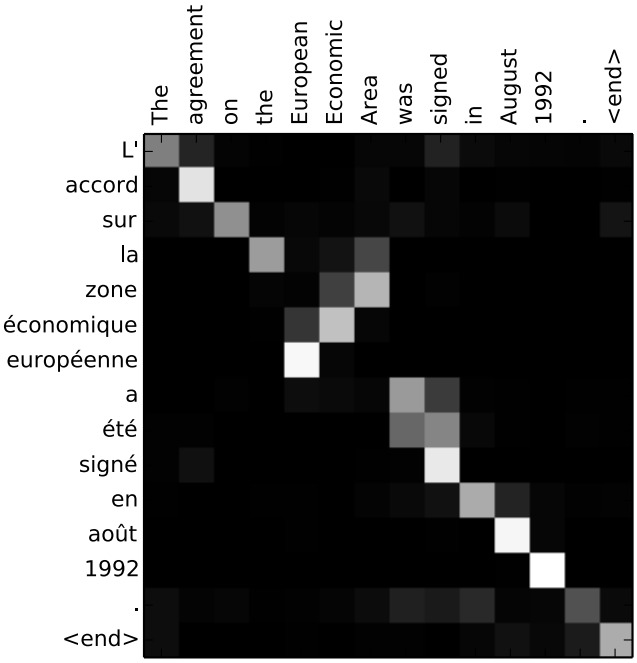

Figure 2.6: Attention weights of source (English, x-axis) and target (French, y-axis) pairs by Bahdanau et al. (2015). Each cell (i, j) indicates the score $\alpha_{i,j}$ of i^{th} target word and j^{th} source word pair, in 0 to 1 greyscale (brighter is higher).

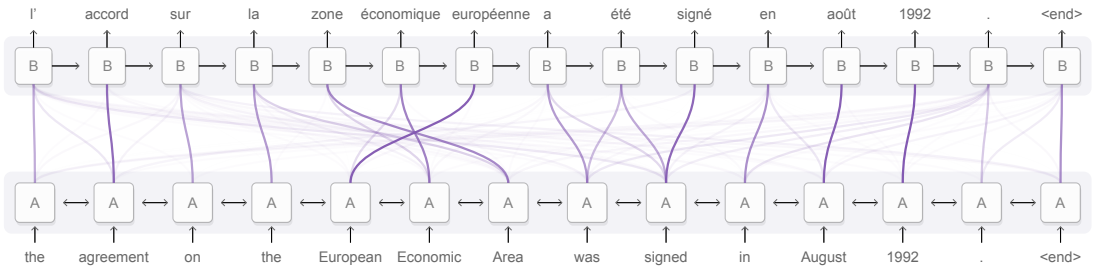

Figure 2.7: Word alignment derived from attention weights by Olah and Carter (2016)

2.4 Parallel Corpora

Until now, one might wonder what data an NMT system learns from. The answer is **parallel corpus**, which is defined as a collection of texts in two or more languages, aligned at sentence level (Williams et al., 2016). Each sentence pair express the same meaning in different languages. Words, per contra, are not necessarily aligned as languages can have different number or order of words to express the same meaning.

Besides, there is ongoing research on unsupervised NMT using only monolingual corpora (Lample et al., 2017; Artetxe et al., 2018), but it has not achieved state-of-the-art performance. Hence, for this project, we study the domain adaptation problem and train our NMT systems on bilingual parallel corpora. We will refer to the language that a system is trained on as **source** side, and the language a system translates to as **target** side. Below is an example of an English-German sentence pair if English is translated to German:

English	source:	Today is Tom's birthday.
German	target:	Tom hat heute Geburtstag.

To train an NMT model, a corpus is usually split into three subsets, specifically training, validation (or development) and test. **Training** set is the data fed to the model for learning, and it normally contains millions of sentences, taking up the majority of a corpus. **Validation** set normally contains thousands of sentences, and it is used for tuning hyperparameters or configurations like learning rate, dropout and early stopping. Finally, **test** set contains thousands of sentences for evaluation purpose.

Often in valid and test sets, the provided target side sentences are called **references** (also called gold standard or ground truth in other places), which are deemed as the correct translations of corresponding source sentences. In contrast, the generated target sentences from a model are called hypotheses, or simply (output) translations. Since valid and test sets are used to evaluate a trained system, reference translations should not be seen by the system.

There has been huge effort to gather parallel corpora from different sources, such as United Nation documents, TED talks and European Medicine Agency (Tiedemann, 2012). However, parallel corpora are commonly constructed and released by large organisations or projects (Resnik and Smith, 2003), so the texts are commonly in the domains of their specific interests. For instance, the latest United Nation corpus contains parliamentary documents and records in its six official languages (Ziemski et al., 2016). Consequently, words like "unite" and "nation" are frequent, which does not happen in the English language in general (Rafalovitch and Dale, 2009).

Thus, it is not possible to have a readily available corpus in every domain for translation tasks. As a result, domain mismatch arises and leads to degraded results, when a system performs translation on a different corpus than the one it has learned from. This problem is magnified for low resource languages because there are only high-quality corpora for limited language pairs, mainly those paired with English (Chu and Wang, 2018). Before we further explain this problem and present current solutions, we first describe how machine translation is evaluated in the next section.

2.5 Translation Evaluation

Since machine translation has emerged quickly from 1990, many machine translation evaluation methods have been proposed, which fall into two categories, human evaluation and automatic evaluation. For this project, we use **bilingual evaluation understudy (BLEU)** (Papineni et al., 2002), an automatic metric, for evaluation because it has a high correlation with human judgment. It is also quick and reproducible, so it serves as a consistent measure for comparison. Particular to our project, BLEU has been extensively used for measuring domain adaptation performance (Etchegoyhen et al., 2018).

BLEU is made up of two components, brevity penalty and weighted n-gram precisions. Brevity penalty penalises translations shorter than references and is calculated as Equation 2.11. The weighted n-gram precisions as a whole measures n-gram overlap between output and reference. Usually, n-gram precisions are calculated up to 4-grams and weighted equally as Equation 2.12. Finally, BLEU is the product of brevity penalty and weighted precisions in Equation 2.13:

$$\text{brevity penalty} = min(1, \frac{\text{output\_length}}{\text{reference\_length}}) \tag{2.11}$$

$$\text{weighted precisions} = (\prod_{i=1}^{4} \text{i-gram\_precision})^{\frac{1}{4}} \tag{2.12}$$

$$\text{BLEU} = \text{brevity penalty} \times \text{weighted precisions} \tag{2.13}$$

Throughout the project, we use an automated script called sacreBLEU[1] (Post, 2018) as an implementation of BLEU. It incorporates standard machine translation tests, and automates processing, tokenisation and scoring.

[1] https://github.com/mjpost/sacreBLEU

2.6 Domain Mismatch

Domain mismatch for machine learning refers to the situation where the distributions of training data and test data are different. For machine translation, Koehn and Knowles (2017) define it as the situation where a model learns from corpus from a specific source and translates texts from another source, which has a different topic, style, level of formality, etc. A root cause of domain mismatch is lack of data as mentioned in Section 2.4, such that a system has to be trained on non-domain-specific data. Koehn and Knowles (2017) identify this to be one of the six challenges that NMT currently faces. The results of their experiments on training (rows) and translating (columns) in different domains for both NMT and SMT, are presented in Figure 2.8. It is clear that NMT's performance (left green bars) drops more than SMT's (right blue bars) when domain changes.

System ↓	Law		Medical		IT		Koran		Subtitles	
All Data	30.5	32.8	45.1	42.2	35.3	44.7	17.9	17.9	26.4	20.8
Law	31.1	34.4	12.1	18.2	3.5	6.9	1.3	2.2	2.8	6.0
Medical	3.9	10.2	39.4	43.5	2.0	8.5	0.6	2.0	1.4	5.8
IT	1.9	3.7	6.5	5.3	42.1	39.8	1.8	1.6	3.9	4.7
Koran	0.4	1.8	0.0	2.1	0.0	2.3	15.9	18.8	1.0	5.5
Subtitles	7.0	9.9	9.3	17.8	9.2	13.6	9.0	8.4	25.9	22.1

Figure 2.8: Quality of NMT and SMT systems (BLEU) when trained (rows) and tested (columns) on different domains by Koehn and Knowles (2017). Green bars denote NMT and blue bars denote SMT.

Another interesting phenomenon we observe from their result is that, for both NMT and SMT, training on all available data sometimes significantly improves domain-specific performance (e.g. Medical for NMT and IT for SMT), whereas it harms in some other cases (e.g. Law). Thus, to blindly include or exclude extra available data for domain specific tasks does not guarantee to enhance system performance. Hence we are motivated to find sophisticated approaches that exploit underlying characteristics of datasets and benefit domain adaptation in neural machine translation.

In this report, we refer to the domain that a system is intended to translate in as **in-domain**, and other domains as **out-of-domain**. Domain adaptation approaches are used to mitigate the domain mismatch problem presented above. The approaches adapt a system to a domain-specific task, leveraging limited in-domain data and relatively more out-of-domain data. Some effective domain adaptation approaches for NMT will be introduced in the next section.

2.7 Domain Adaptation Approaches

Domain adaptation for NMT only emerged recently, and some successful techniques from both data and model perspectives have been well presented by Chu and Wang (2018) in their survey. Most up-to-date approaches are outlined in the hierarchical diagram in Figure 2.9. Overall, these can be divided into either data or model categories. Whilst data perspective approaches make use of different data to achieve domain adaptation, model-centric approaches modify training objective (cost function), architecture or decoding algorithm in a machine translation system.

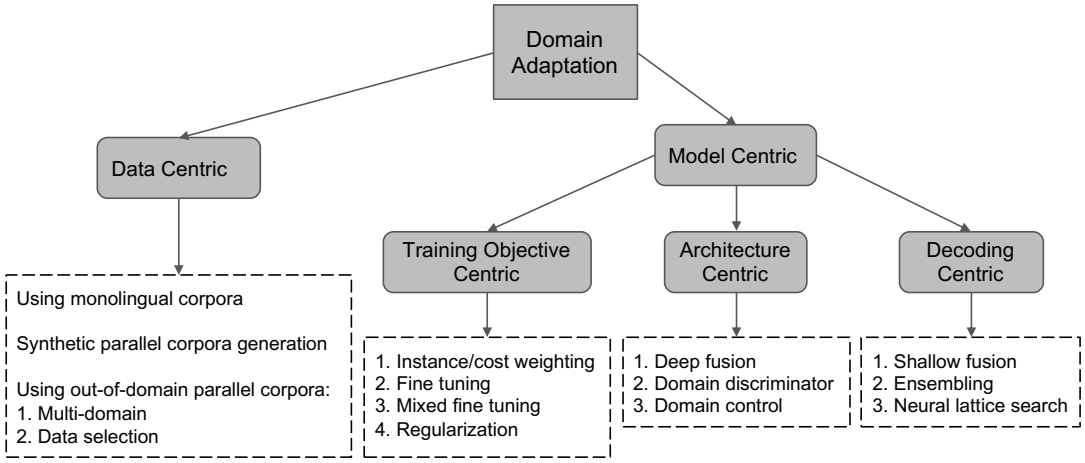

Figure 2.9: Outline of NMT domain adaptation techniques by Chu and Wang (2018).

In the following sections, we will introduce a few data-centric and training objective (cost function) centric methods, from which our project is inspired and motivated.

2.7.1 Multi-domain System

Sennrich et al. (2016a) proposed to derive domain tags from target sentences, and to add them to corresponding source sentences in a pilot study. As such, an NMT system can learn to produce translations conditioned on given domain tags, so essentially the

system becomes a multi-domain system. In particular, their research was on controlling the politeness level of English to German translation, where politeness level in English is not as obvious as that in German because German uses honorifics.

Their proposed method can be described in three steps. First, the politeness level of target sentences in training data is labelled automatically using a rule-based annotation tool. Then the tags (labels) are concatenated to the end of source sentences, to inform the model during training. Finally, at the test stage, the politeness level of target (reference) sentences is also identified and added to source sentences. As a result, the model will tend to produce translations closer to the given politeness tags.

We can consider the domain tag (e.g. a politeness tag) as an extra word in each source sentence. During training, the NMT system is able to learn the (high) probability of producing an output of the same domain, conditioned on the input containing this extra word. Moreover, with the presence of attention mechanism, this domain tag can be attended by each target word directly during translation. Thus, when a test sentence with a domain tag is fed to the NMT system, the system is more likely to produce translations in that domain.

However, we argue that there is a potential limitation to their approach, which is a lack of comprehensiveness. Their largest improvement was on "oracle experiments", where they labelled and added domains of reference translations to source sentences in a test set. This is not a conventional evaluation way adopted by current machine translation research community (Dorr et al., 2011), because their system accessed some degree of reference information before producing its own translations. We suggest that an ideal approach should be to translate a source sentence with different possible tags added to it, and compare the translations with references of corresponding politeness levels. However, it is almost impossible to obtain such a corpus. That being said, it remains unknown whether their multi-domain system can still achieve equally good results, if the source sentences are given different domains than the annotated one for reference.

Although multi-domain could be more useful or powerful than a system adapted to a single domain, another drawback worth mentioning is model complexity and data scarcity. To model multi-domain data, a system needs a deeper network with more parameters, leading to more computational resources and time required. More importantly, if the system looks at sub-corpus domains like politeness in Sennrich et al. (2016a)'s work, sufficient training data will be needed for each domain. Paradoxically, lack of data itself is an underlying reason for domain mismatch.

2.7.2 Fine-tuning

Luong and Manning (2015) proposed to **fine-tune** NMT systems for domain adaptation. It refers to adapting a general NMT system (already trained on a large amount of out-of-domain data) to a specific domain by training on in-domain data of much smaller size until the model converges. The model will gradually (over)fit to the in-domain data, backed up by out-of-domain "knowledge". Similar techniques have been widely applied in other areas of machine learning, such as computer vision (Pan and Yang, 2010) and speech processing (Yu et al., 2010).

The reason behind this is simple yet elegant. Training on a large amount of out-of-domain data followed by fine tuning on small in-domain data can be interpreted as learning a language in general followed by adapting to a certain writing style. Although data come from different domains, they still have some features and characteristics in common, such as grammar and vocabulary. Model parameters can learn to represent the language(s) in general using a large amount of out-of-domain data. Then with fine-tuning, the domain-specific data of small size can adjust parameters to capture specific information such as named entities, choice of words, sentence length, etc. The core idea is to save the best for last.

2.7.3 Mixed Fine-tuning

Inspired by, and based on multi-domain NMT and fine-tuning mentioned above, Chu et al. (2017) proposed **mixed fine-tuning**. It first trains a system on out-of-domain sentences with domain tags, similar to training a general multi-domain system. The next step is to fine-tune the model until convergence, using a mix of out-of-domain and over-sampled in-domain data, each with corresponding tags. According to their research, mixed fine tuning outperforms systems using multi-domain tags or fine-tuning separately.

In their experiments, a domain tag was simply added to a source sentence based on which corpus the sentence was taken from. Therefore, a possible improvement is to classify domain tags using a more sophisticated metric like cross-entropy (van der Wees et al., 2017) or sentence embedding (Wang et al., 2017a), to distinguish pseudo-in-domain data from an out-of-domain corpus. Nevertheless, deriving tags from source sentences is more widely applicable comparing to deriving tags from target sentences, because it no longer needs to look at references during evaluation phase.

2.7.4 Sentence Weighting

Another recently proposed method that helps domain adaptation in NMT is **sentence weighting** using a domain classifier (Chen et al., 2017), language model cross-entropy (Wang et al., 2017b) and sentence embeddings (Zhang and Xiong, 2018).

The core idea of sentence weighting is that in-domain sentences should have higher weights in the objective function than out-of-domain sentences, such that the model will be penalised more by errors made on in-domain data than out-of-domain data. As a result, the model weights can adjust to model in-domain data to a larger extent, during backpropagation (through time). This leads to a better domain-adapted model. An underlying assumption is that a lower cross entropy measured by language model is useful in reflecting that a sentence is closer to a domain, and vice versa.

In this project, we focus on weighting using language models. It has an advantage of extracting pseudo-in-domain sentences from out-of-domain corpus (Axelrod et al., 2011). For example, the sentence "A reporter says the use of amoxicillin causes cholestatic jaundice" can come from news data, but probably it will have a lower cross entropy when calculated using language model trained on biomedicine domain. Thus learning more from this sentence could benefit biomedical domain rather than news domain.

First, a raw weight w is computed from adding differences of out-of-domain cross entropy and in-domain cross entropy, for both source and target sides, shown as Equation 2.14. Then mini-max normalisation (Priddy and Keller, 2005) shown in Equation 2.15, is applied to move all raw weights into [0, 1] range to avoid negative values. Let $H_{S,D}(u)$ denote the cross-entropy between sentence u from side S domain D, and the language model trained on side S domain D, the weight λ for sentence pair $< x, y >$ is calculated as:

$$w = (H_{source,out}(x) - H_{source,in}(x)) + (H_{target,out}(y) - H_{target,out}(y)) \tag{2.14}$$

$$\lambda = \frac{w - w_{min}}{w_{max} - w_{min}} \tag{2.15}$$

After a weight λ_i is computed for the i^{th} sentence pair, it is incorporated into objective function (cost) of the sentence pair as a coefficient:

$$\text{weighted\_cost}_i = \lambda_i \times \text{original\_cost}_i \tag{2.16}$$

Consequently, when the error derivatives are calculated from the cost, the weight influences the magnitude of backpropagation. Since the weights are normalised to between 0 and 1, the direction (polarity) of backpropagation remains unchanged. Under such a sentence weighting scheme, the model parameters are updated with a greater magnitude for incorrect word produced in in-domain sentences. Hence the model parameters can be adapted to in-domain data.

We reproduce this sentence weighting method and present the distribution of weights computed in Figure 2.10. The x-axis represents the magnitude of a weight and the y-axis represents the frequency of a magnitude seen in all weights.

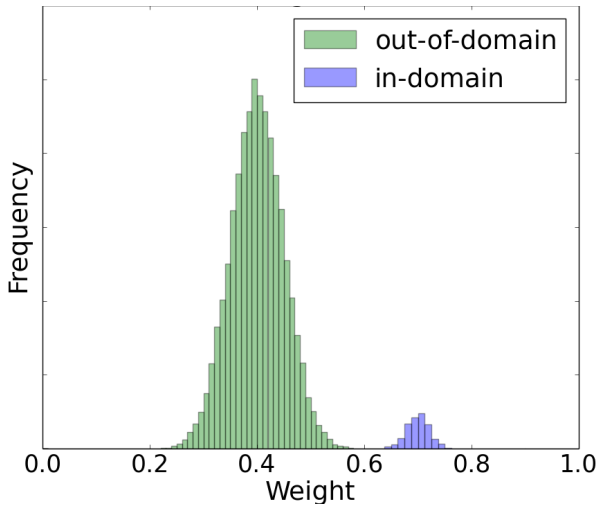

Figure 2.10: Distribution of weights resulted from Wang et al. (2017b)'s sentence weighting

3 | Improving Sentence Weighting

In this chapter, we intend to improve current sentence weighting approaches. First we try to make Wang et al. (2017b)'s sentence weighting more aggressive by increasing the gap between the distributions of in-domain and out-of-domain weights. In the second half of this chapter, we propose a novel tag-and-weight approach that weights sentences (Wang et al., 2017b) when building a multi-domain system (Sennrich et al., 2016a).

3.1 Increasing Domain Weight Difference

Without weighting, training on in-domain or out-of-domain data is simply a trivial case of assigning zero weights to unused data. Similarly, training on data from both domains means to have equal weights for all sentences. In Figure 3.1 the equivalent weightings of the three cases are visualised. Wang et al. (2017b)'s approach introduces a difference between weights of out-of-domain and in-domain data, so that their system can put more focus on in-domain sentences. As a comparison to the above three cases, weight distribution resulted from their approach is shown in the left plot in Figure 3.2.

However, one thing remains unexplored is how large the difference should be. We plan to enlarge the difference between in-domain and out-of-domain weights. We hypothesise that with increased in-domain weights and reduced out-of-domain weights, a model can not only better adapt to in-domain data, but also better distinguish pseudo-in-domain data from out-of-domain data. This follows the same logic as sentence weighting, but it is mathematically closer to training a model on in-domain data only.

From an initial attempt to compute Wang et al. (2017b)'s sentence weights, we find that the distribution of weights appears to be two normal distributions. Hence we assume the overall distribution of weights is a Gaussian mixture of two components. Then we use expectation-maximization (EM), to fit and classify the weights. The classified distribution with a smaller mean contains the out-of-domain weights and the that with

a larger mean contains the in-domain weights. To enlarge the difference between the two identified domains, we subtract 0.1 from each out-of-domain weight and add 0.1 to each in-domain weight. We ensure that weights are in between 0 and 1 by thresholding. Our proposed algorithm results in an increased difference between domain weights, shown in Figure 3.2.

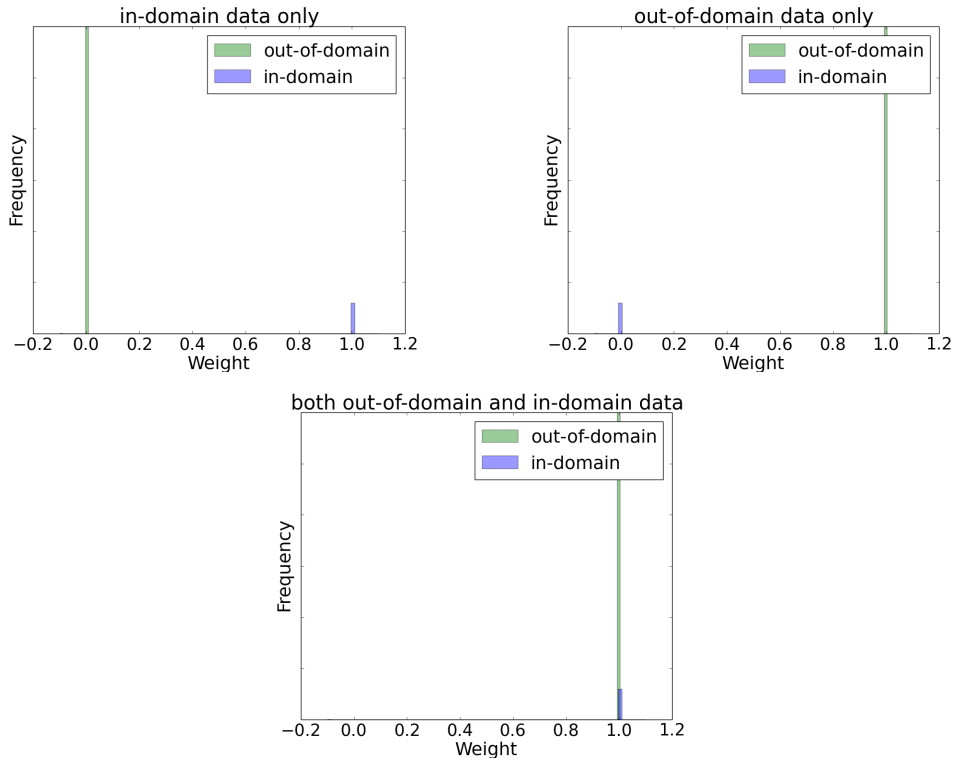

Figure 3.1: Illustration of equivalent weight distribution when a system is trained on in-domain data (blue), out-of-domain data (green) and both.

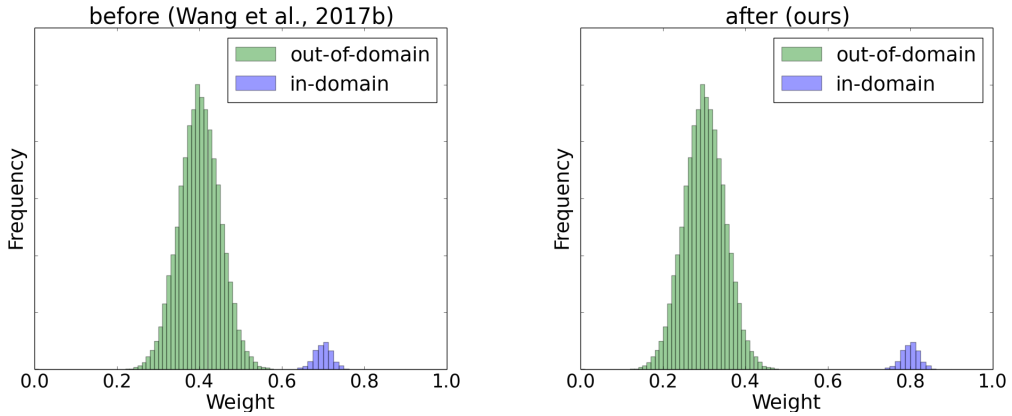

Figure 3.2: Increasing the difference between weight distributions for out-of-domain (green) and in-domain (blue) data.

We implement the above algorithm using `sklearn.mixture.GaussianMixture()` with EM algorithm, in scikit-learn[1], a popular machine learning package in Python (Pedregosa et al., 2011). First we create a `GaussianMixture()` object, with parameter `n_components=2` to indicate that there are two Gaussian distributions in the mixture. Next, all weights are fit into the model using `GaussianMixture.fit` method and predicted a label (distribution) each using `GaussianMixture.predict` method. Finally 0.1 is subtracted from weights in the out-of-domain distribution, and added to weights in the in-domain distribution, with 0 and 1 as thresholds.

3.2 Tag-and-weight

In this section, we propose a novel domain adaptation technique inspired by both data and model perspective approaches, called **tag-and-weight**. The new approach performs sentence weighting (Wang et al., 2017b) on sentences with domain tags based on cross-entropy difference.

We add a domain tag to each source sentence, in all training, validation and test sets. The detailed procedure of our tagging algorithm is as follows:

1. train two language models (LMs) on in-domain and out-of-domain source sentences in training data.
2. use two LMs to compute cross entropy values of each sentence in all data.
3. calculate a score for each sentence by subtracting cross entropy of in-domain LM from that of out-of-domain LM.
4. mini-max normalise all scores into [0, 1] range.
5. convert scores to tags, that do not already exist in the vocabulary.
    5.1. *Either* fit scores to a Gaussian mixture of two components, and parameterise using expectation-maximisation. Then, convert scores with smaller mean to "<out>" and scores with larger mean to "<in>".
    5.2. *Or* convert scores to "<domain_X >", where X is the tenth digit of a score. E.g. if a sentence is scored 0.6789, it will be tagged "<domain_6>".
6. concatenate the tag at the beginning of its corresponding source sentence.

Steps 1-4 are similar to Wang et al. (2017b)'s sentence weighting introduced in Section 2.7.4, except that we do not take target sentences into consideration. This is be-

[1] scikit-learn: https://scikit-learn.org/stable/

cause we assume that source and target sentences are in the same domain if they are in a pair. Using cross-entropy, we expect to identify pseudo-in-domain and pseudo-out-of-domain data. Step 5.1 results in 2 different tags while 5.2 results in 10 different tags (X ranging from 0 to 9, from most out-of-domain to most in-domain). The two methods will create a 2-domain system and a 10-domain system respectively.

Our method can be more applicable than Sennrich et al. (2016a)'s. It derives tags from source sentences without accessing target (reference) sentences. This is a more generalisable way of tagging. One potential limitation of our approach is that it requires a huge data size if we choose to model various domains (e.g. in Step 5.2 we have 10 tags/domains). The more domains we have, the more data we need in order to represent the sentences and domains well.

Below we feature comparison of a tagged sentence pair with an original sentence pair. Only the source sentence is changed and the target sentence remains unchanged. This applies to all training, validation and test data, after preprocessing which will be introduced in a later chapter.

	Source	Target
original	<s> A source sentence . </s>	<s> A target sentence . </s>
tagged	<s> <tag> A source sentence . </s>	<s> A target sentence . </s>

Table 3.1: An example comparing tagged and original sentence pairs

Apart from tagging, we adopt the exact same sentence weighting scheme as Wang et al. (2017b), describe in Section 2.7.4. Theoretically, a multi-domain system is able to deal with different domains. However, if we already know that a multi-domain system only needs to perform translation (evaluation) in a specific domain, we wish to adapt it to that domain. Having sentence weighting is like the icing on the cake, where it helps to adapt the model to the domain that it is more likely to see during the evaluation.

This system combines the use of domain tags and sentence weighting, but the two techniques are from different perspectives. We argue that the two techniques can improve domain adaptation performance orthogonally without affecting each other. To verify our claim, we will compare this system with a multi-domain system and a system with only sentence weighting in a later chapter.

4 | Proposing Word Weighting

To the best of our knowledge, there is no published literature on word-level weighting for neural machine translation domain adaptation purpose. In this chapter, we first state the feasibility of word-level weighting and our motivation to carry out such work. Then we propose three word weighting schemes based on word probability and language model scores. Finally, we discuss the possibility of summing up word weights to form a new sentence weighting method.

4.1 Motivation

Researchers have worked on tackling domain adaptation in NMT. We have introduced some methods involving cost function such as fine-tuning and sentence weighting, as well as proposed some improvement to sentence weighting. These sentence weighting schemes covey the idea of treating data differently such that the model can put more emphasis on in-domain data during training. In this chapter, we move one step further to investigate on word-level weighting.

When proposing sentence level weighting, Wang et al. (2017b) also experimented on how training solely on in-domain or out-of-domain data will affect performance. This is actually a trivial case of sentence weighting, as we stated in the previous chapter. Moreover, we argue that weighting sentences is a trivial case of weighting words. Under sentence weighting, words in different sentences will receive different weights, but words in the same sentence will have the weights. Weighting each word equally in a sentence is equivalent to weighting a sentence with the same magnitude. This can be easily verified in Equation 4.1 where w_i denotes the weight for the i^{th} sentence and *word_loss* denotes the loss calculated on word predictions against the reference:

$$\text{total\_loss} = \underbrace{\sum \overbrace{w_i \times \sum \text{word\_loss}}^{\text{sentence\_loss}}}_{\text{sentence weighting}} = \underbrace{\sum \overbrace{\sum w_i \times \text{word\_loss}}^{\text{sentence\_loss}}}_{\text{word weighting}} \tag{4.1}$$

Furthermore, NMT operates on word (or sub-word) level, where the whole system can be regarded as a large language model on both source and target words. In addition, during backpropagation, the objective function is computed as the sum of losses on words. While our project is being carried out, Gong et al. (2019) proposed to adjust weight and gradient for each token (word) in a sentence to regularise the bias caused by maximum likelihood estimation.

These points justify the feasibility of modifying word gradient directly in a neural network model to achieve desired effects. Hence we are motivated to propose word-level weighting in NMT for domain adaptation, which weights each word in an output sentence separately based on certain schemes. When the loss of a word is calculated, its weight acts as a coefficient, which amplifies or shrinks the loss. Under such schemes, words that are more important to in-domain data should be assigned larger weights, leading to larger gradients. Moreover, if we break down weights to word level, we might be able to identify pseudo-in-domain words or phrases (e.g. named entities) more accurately.

4.2 Word Weighting

In this section, we design and propose completely new word weighting methods for neural machine translation domain adaptation. The methods include using smoothed word frequency (probability) ratio and distributing sentence weights.

4.2.1 Ratio of Word Probability

In Information Retrieval, an important metric to measure the importance of a word to a document in a collection of documents is term frequency-inverse document frequency (TF-IDF) (Salton and McGill, 1986). The TF-IDF score of a word term t to a document d is calculated as Equation 4.2 where D denotes the whole collection.

$$\text{tfidf}(t,d,D) = \frac{\text{term\_frequency}(t,d)}{\text{document\_frequency}(t,D)} \tag{4.2}$$

It has been widely used as a weighting factor in text mining and searches, while the reason behind is simple. Term frequency indicates the importance of a word divided by document frequency to offset it, because naturally some words appear more often than other words in a language.

We propose to use a ratio of word probabilities (frequencies), which resembles TF-IDF, to measure the importance of a word to a domain. We change the TF-IDF ratio to be that of in-domain word probability to out-of-domain word probability. The in-domain probability reflects the importance of a word to the specific domain, which is then normalised by the word's out-of-domain probability.

Also, it is possible that a word does not occur in one of out-of-domain or in-domain corpora (but not both). Counting results in zero probability in this case, so we use a language model to compute word probabilities, where unseen words will get a smoothed probability. The weighting method is presented in Equation 4.3, where "unk" refers to an unseen word.

$$\text{weight}(w_i) = \begin{cases} \dfrac{P_{in}(w_i)}{P_{out}(unk)} & \text{if } w_i \text{ is not in out-of-domain data} \\ \dfrac{P_{in}(unk)}{P_{out}(w_i)} & \text{if } w_i \text{ is not in in-domain data} \\ \dfrac{P_{in}(w_i)}{P_{out}(w_i)} & \text{otherwise} \end{cases} \tag{4.3}$$

This weighting scheme favours words that occur frequently in in-domain data and infrequently in out-of-domain data. The intention is to make objective function penalise more for mistakes on producing in-domain words. As such, the model should have a tendency to produce a vocabulary more similar to in-domain data.

4.2.2 Ratio of Logarithmic Word Probability

According to Zipf's Law, which states that the frequency of a word is inversely proportional to its rank, the probabilities of words in a corpus can vary largely, ranging from 10^{-7} to 10^{-3}. This results in a problem of using the ratio of probability, that the computed weights can be exponentially large or small. One improvement for our previous method is to take logarithm (base 10 in our case) of probabilities to reduce the variance of ratio. Since probabilities are always smaller than 1, taking logarithm results in negative values. Thus, we add a constant c to both numerator and denominator to bring the smallest log probability to 1. The small constant c also acts as a regularisation term to prevent the ratio from being too large or small. This leads to Equation 4.4:

$$\begin{aligned} \text{weight}(w_i) &= \frac{\log P_{in}(w_i) + c}{\log P_{out}(w) + c}, \\ \text{where } c &= max\Big(1 - \log P_{in}(w), 1 - \log P_{out}(w_i)\Big), \\ &\text{and we use base 10 in logarithm} \end{aligned} \tag{4.4}$$

Similar to the ratio of word probabilities, the ratio of log probabilities still favours words occur frequently in in-domain data and infrequently in out-of-domain data, but to a less extreme degree.

4.2.3 Distributing Sentence Weight

As we have discussed earlier, domain mismatch is not only about lexical difference or word frequency, but also complicated language features like sentence length, writing style, etc. From this point of view, both methods introduced in previous sections have some drawbacks.

An obvious one is that the approaches consider target words when computing weight, but source language is not utilised. Moreover, a word weight is calculated from the (smoothed) frequencies of that single word, so word context is not considered. The same word always receives the same weight regardless of which sentence it is in. However, in a natural language, there are long-range dependencies like subject-verb agreement and idioms, and even worse, discourse relationships between sentences.

To address the above two limitations, we can make use of sentence weights in Wang et al. (2017b)'s sentence weighting. The sentence weighting treats source and target sentence equally and considers word context using language model cross entropy. Building from this work, we design another word weighting scheme which distributes the sentence weights to each word in the target sentence, such that the average of word weights in a sentence is equal to the sentence weight. The idea is demonstrated in Equation 4.5, where weight_s denotes sentence weight and weight_i denotes the word weight for the i^{th} word in the sentence. The resulted total loss (cost) from word weighting will be different from that from the original sentence weighting scheme, and we expect this to be more sophisticated than the loss from sentence weighting.

$$\begin{gathered}\text{sentence weighting: total\_loss\_sent} = \sum \text{weight}_s \times \sum \text{word\_loss} \\ \Downarrow \\ \text{word weighting: total\_loss\_word} = \sum\sum \text{weight}_i \times \text{word\_loss} \\ \text{where weight}_s = \frac{1}{n}\sum_i^n \text{weight}_i\end{gathered} \tag{4.5}$$

Until now, we have an idea of distributing sentence weights, so we need a way to distribute a sentence weight to words. In general, we decide to assign a score to each word, and the word weight should be proportional to that score, multiplied (normalised) by the length of the whole sentence. For a sentence of length n made up with words $x_1,...,x_i,...,x_n$, the weight of word x_i is calculated as Equation 4.6, where score() denotes a function to score a word (explained later), w_s denotes sentence weight and w_x_i denotes the weight of word x_i.

$$\begin{gathered}w\_x_i = \frac{n \times w_s \times \text{score}(x_i)}{\text{score}(x_1) + ... + \text{score}(x_i) + ... + \text{score}(x_n)} \\ \text{which satisfies } w_s = \frac{1}{n}\sum_i^n w\_x_i\end{gathered} \tag{4.6}$$

Multiplying each word weight with sentence length n is necessary because it sets off the influence of sentence length on word weight. Without this coefficient, a sentence weight for a long sentence is dilated too much, while a word in shorter sentences will receive too much weight.

A good score() function should be designed so that a score is large if the word is more likely to occur in in-domain data and less likely to occur in out-of-domain data. We use language models to calculate word probabilities for in-domain and out-of-domain data. We compute probabilities for all words in a sentence in one go so that word context is considered[1]. Then we use logarithmic probability ratio shown in Equation 4.7 as our score() function:

$$\text{score}(w_i) = \frac{\log P_{in}(w_i)}{\log P_{out}(w_i)} \tag{4.7}$$

[1]Corresponds to `full_scores()` method in KenLM's Python module: https://github.com/kpu/kenlm/blob/master/python/kenlm.pyx

The use of ratio of logarithmic word probability is similar to our previous scheme. However, this scheme distributes sentence weights based on the ratio, whereas the previous scheme can be considered as distributing a constant 1 to each word. This scheme assigns different weights in different context for the same word.

4.3 Summing Word Weights

Apart from the sentence weighting introduced in earlier chapters, we can also obtain sentence weights by summing up word weights in a sentence, normalised by sentence length. The idea behind is that a sentence should be weighted more if it contains more important in-domain words. While word-level weighting may be too aggressive to incorporate word context features, summing up word weights alleviates such problem. We will experiment on the three word weighting schemes first, and further experiment on summing weights if a word weighting achieves promising results.

5 | Experiments

In this chapter, we design and perform two experiments to evaluate our proposed sentence and word weighting schemes. We first present our datasets and experiment design. Then we describe steps for preprocessing datasets and training language models. Finally, we will report the results of our experiments on both datasets and identify the approaches that result in improvement.

5.1 Data and Domains

To evaluate our proposed approaches, we prepare two tasks of varying degree of domain specificity and proportion of in-domain data:

1. adapting news to a biomedical domain for Romanian to English translation.
2. adapting news to TED talks for English to German translation.

In this section, we describe our data sources and partition for each task. All the data we use can be found and directly downloaded from the open parallel corpus[1] (OPUS) (Tiedemann, 2012). To better understand the domain mismatch problem in each task, besides data size, we report two extra linguistic features, namely average sentence length and logarithmic type-token ratio.

Average sentence length is computed as the number of words divided by the number of sentences as Equation 5.1. It reflects text structure and readability (Stymne et al., 2013), and has been shown to vary across different domains (Kwon et al., 2009), writing styles and topics (Koeva et al., 2012).

$$\text{average sentence length} = \frac{\text{count}(\text{words})}{\text{count}(\text{sentences})} \tag{5.1}$$

[1] OPUS: http://opus.nlpl.eu/index.php

Type-token ratio is computed as the number of unique words (size of vocabulary) divided by the number of total words as Equation 5.2. It indicates the variety of lexical choice (François and Fairon, 2012; Bentivogli et al., 2016), as well as density of subject matters or thoughts (Stymne et al., 2013).

$$\text{type-token ratio} = \frac{\text{count}(\text{unique words})}{\text{count}(\text{all words})} \tag{5.2}$$

Notwithstanding, the type-token ratio is dramatically affected by corpus size (Kettunen, 2014). From Zipf's law, we can deduce successively that the number of new words increases less than proportionately to an increase in number of total words. Therefore we report **logarithmic type-token ratio** instead, which is reasonably invariant to corpus size (Weitzman, 1971). It is computed as Equation 5.3, and we use a logarithm of base 10.

$$\text{logarithmic type-token ratio} = \frac{\log\big(\text{count}(\text{unique words})\big)}{\log\big(\text{count}(\text{all words})\big)} \tag{5.3}$$

The number of sentences, number of words and number of unique words required to compute the above two measurements, can be obtained by executing the following three commands.

```
$ wc -l input_file # count sentences

$ wc -w input_file # count words

$ awk -v RS=" " '{a[$0]++} END{for(k in a) sum++; \
> print sum}' input_file # count unique words
```

5.1.1 Romanian-English News-Biomedicine

For Romanian to English translation (RO-EN), we adapt news data to a biomedical domain. The news data mainly comes from news translation task in Workshop on Machine Translation (WMT) 2016[2]. It is made up with proceedings of the European Parliament (Europarl) (Koehn, 2005), Southeast European Times (SETimes) and back-translated monolingual data from News Crawl[3] (Sennrich et al., 2016c). We refer to it as "WMT news" from now on. Our biomedical data is a corpus made out of documents from the European Medicines Agency (EMEA) and we refer to it as "EMEA".

[2]WMT16: http://www.statmt.org/wmt16/translation-task.html

[3]Back-translations from monolingual News Crawl data: http://data.statmt.org/rsennrich/wmt16_backtranslations

The WMT news corpus contains 2.4 million sentence pairs as our out-of-domain training data. Regarding in-domain data, EMEA corpus contains nearly 1 million sentence pairs. We randomly draw 2,000 sentence pairs to our validation set, and the same applies to our test set. The rest 990,499 sentence pairs form in-domain training data. Both in-domain and out-of-domain training data are combined to form training data of 3.4 million sentences and shuffled (by the system) during training. We present the statistics of our RO-EN data in Table 5.1.

		number of sentence	**average length**		**log type-token ratio**	
			source	target	source	target
Train	Combined	3,389,297	19.33	17.72	0.827	0.822
	WMT news	2,398,798	22.34	20.36	0.831	0.826
	EMEA	990,499	12.06	11.33	0.775	0.771
Valid	EMEA	2,000	12.21	11.53	0.885	0.877
Test	EMEA	2,000	12.51	11.77	0.886	0.877

Table 5.1: Data for Romanian-English translation

In-domain EMEA data makes up 29.2% of the training data and out-of-domain WMT news makes up the rest 70.8%. Regardless of language, WMT news has an extremely long, nearly doubled sentence length comparing to EMEA, and a higher lexical diversity. Interestingly, EMEA valid and test sets have the highest lexical diversity. Language-wise, Romanian (source) sentences are normally 1-2 words longer, and have a slightly higher lexical diversity than English (target) sentences. In general, although one-third of the training data are in-domain, but the domain difference between WMT news and EMEA is significant.

5.1.2 English-German News-TED Talks

For English to German translation (EN-DE), we stick to WMT news as out-of-domain data. It is made up of Europarl, Common Crawl and News Commentary, and contain 4.7 million sentence pairs. Our in-domain data is TED talks from speech translation task in the International Workshop on Spoken Language Translation (IWSLT) each year[4]. We refer to it as "IWSLT". The same data combination was used in Luong and Manning (2015)'s fine tuning and Wang et al. (2017b)'s sentence weighting experiments.

we argue that the training data for EN-DE is of higher quality than that for RO-EN, because of two facts. First is that its out-of-domain data does not contain back-translated

[4]IWSLT17: http://workshop2017.iwslt.org/59.php

monolingual texts, and second is that the in-domain data has been used for an established speech translation task for many years.

Since the datasets are used by mainstream translation tasks, training, validation and test sets are readily available. We mix WMT news training data and IWSLT training data to form our overall training data. We then use IWSLT 2012's test set as our validation set, and IWSLT 2013 and 2014's test sets as our two test sets, named Test1 and Test2. This train/valid/test partition follows Wang et al. (2017b)'s work. Similarly, we present EN-DE data statistics in Table 5.2.

		number of sentence	**average length**		**log type-token ratio**	
			source	target	source	target
Train	Combined	4,714,797	22.70	21.06	0.817	0.837
	WMT news	4,520,620	22.93	21.27	0.817	0.837
	IWSLT	194,177	17.28	16.09	0.816	0.843
Valid	IWSLT12 test	1700	15.58	14.64	0.851	0.877
Test1	IWSLT13 test	993	17.94	16.80	0.859	0.881
Test2	IWSLT14 test	1305	16.24	15.38	0.863	0.883

Table 5.2: Data for English to German translation

For this task, in-domain data only makes up 4.1% of total training data, in clear contrast to the previous Romanian-English task. We observe that in-domain WMT news sentences are on average 5.5 words shorter than out-of-domain IWSLT sentences, for both languages. Regarding logarithmic type-token ratio, vocabularies of WMT news and IWSLT have similar complexity for both languages. Language-wise, English (source) sentences are slightly longer, but its vocabulary diversity is always lower than German (target) sentences, regardless of domain or partition. Overall the task is difficult due to its low resource in-domain data.

5.2 Experiment Settings

In this chapter, we design and describe our experiments on the proposed sentence and word weighting schemes. Since we aim for improved adaptation with our methods, we choose Wang et al. (2017b)'s sentence weighting to be our baseline. Also, to verify their claim, we train a vanilla system without any domain adaptation technique. In order for resulted systems to be comparable, we keep all configurations unchanged, except for weighting method and/or weights.

We experiment on all proposed approaches together with baseline and vanilla systems using Romanian to English (RO-EN) task, except tag-and-weight. This is because tag-and-weight approach needs a fairly large amount of data in order to represent multiple domains whereas RO-EN has a relatively small data size.

For English to German (EN-DE) task, we first run vanilla and baseline experiments. Then we will evaluate our novel tag-and-weight approach, with different configurations of tag amount and weighting. Finally, we identify all the techniques that achieve promising results in RO-EN task and experiment them on EN-DE task.

RO-EN data is small in scale comparing to EN-DE, which allows for quick experimentation given that machine translation experiments take long time[5] and computational resources are expensive and limited.

Table 5.3 and Table 5.4 summarise our planned experiments for RO-EN and EN-DE tasks respectively. The word "vanilla" refers to a system without any domain adaptation technique, i.e. a standard NMT system.

Weighting	**Approach**
none	vanilla
sentence	*baseline*
	increasing domain difference
word	word frequency ratio
	log word frequency ratio
	distributing sentence weight
summing word weighting that beats baseline	

Table 5.3: Experiment planned for RO-EN

Weighting	**Approach**	**Description**
none	vanilla	
sentence	*baseline*	sentence weighting
	tag-and-weight	2 tags
	tag-and-weight	2 tags + sentence weighting
	tag-and-weight	10 tags
	tag-and-weight	10 tags + sentence weighting
and all systems that beat baseline in RO-EN		

Table 5.4: Experiment planned for EN-DE

[5]Training one system on 1 NVIDIA GTX 1080 GPU takes 1-2 days using our configurations.

We choose to run experiments with Marian[6] (Junczys-Dowmunt et al., 2018), which is an efficient neural machine translation framework written purely in C++. It implements state-of-the-art architectures and techniques. One feature that is crucial to the success of our project is that it supports cost weighting at sentence-level or word-level.

Since the goal of our project is to explore sentence and word weighting schemes, we *do not* perform hyperparameter tuning. We use the same model configurations as Marian's reproduction of the University of Edinburgh's submission to WMT2016 news translation task for RO-EN (Sennrich et al., 2016b). One exception is early stopping, which is changed to happen once there is no improvement in five consecutive validation BLEU scores. The reason is that in a pilot experiment we find that it is hard for cross-entropy to stall under sentence weighting.

The model configurations can be found in the `run-me.sh` script in Marian's example of Edinburgh's WMT16 RO-EN[7]. We outline the important configurations in Table 5.5.

Paramter	Value
encoder type	bidirectional RNN
encoder layer	1
decoder type	unidirectional RNN
decoder layer	1
cell type	GRU
embedding size	512
hidden unit size	1024
layer normalisation	True
RNN dropout	0.2
source word dropout	0.1
target word dropout	0.1
mini-batch size	dynamic
optimiser	Adam
learning rate	0.0001
validation criteria	translation, cross entropy
early stopping	5
beam size	12

Table 5.5: Important configuration for experiments

[6]Marian: https://marian-nmt.github.io/

[7]Marian's reproduction of Edinburgh's WMT16 RO-EN: https://github.com/marian-nmt/marian-examples/tree/master/training-basics

5.3 Preprocessing

Before data are fed to a system for training, they are preprocessed for better performance. Preprocessing is done in mainstream NMT frameworks like OpenNMT, Marian and Nematus. In this section, we introduce our preprocessing steps for RO-EN and En-DE. Following preprocessing, we describe how statistical language models are trained for calculating cross entropy and probabilities used in weighting.

5.3.1 Preprocessing Corpora

First, all punctuation marks are normalised. Then, particularly for Romanian, we normalise Romanian letters and remove diacritics.

Following character normalisation, sentences in all datasets are tokenised, so words and punctuation marks are split by space. Next, empty and long sentences over 80 tokens are cleaned from the training corpus. Then a truecaser is trained on training data to inform the model of which words (characters) should be capitalised (e.g. first letter for named entities). Afterwards, it is applied to all data to convert words to lowercase.

We then segment all words to subwords. After word segmentation, the system can deal better with rare words as well as keep a reasonable vocabulary size (imagine an extreme case, where English words are segmented to characters. This results in only 26 letter plus symbols). The widely used technique now is the unsupervised segmentation based on byte pair encoding (BPE) by Sennrich et al. (2016d). We set the final vocabulary size to be 85,000.

Word segmentation is important to our project. All of our approaches are only applied after words are segmented. This is simply because vocabulary changes after the words are segmented, and costs are calculated at the subword level.

The whole preprocessing procedure, same as model configurations, follows Edinburgh's WMT2016 news task submission. For German and English, we do not apply Romanian-specific steps. The code for preprocessing can be found in the self-contained `script/` folder[8] in Marian's example of Edinburgh's WMT16 RO-EN.

[8]Preprocessing scripts folder: https://github.com/marian-nmt/marian-examples/tree/master/training-basics/scripts

5.3.2 Training Language Models

Statistical language models are needed throughout our project to compute cross entropy and word probabilities. We pick KenLM[9] (Heafield et al., 2013) due to its ease of use. It incorporates modified Kneser-Ney smoothing and provides fast querying with low memory cost. KenLM also allows users to easily specify the order n for n-grams in the command line when building a language model. It also has a Python interface for querying, which can be integrated with Python scripts for sentence or word weighting methods easily.

Compilation of KenLM is done using `cmake`, by following the command listed in `README.md` file in its GitHub repository[9]. After that, a language model can be built with the command listed below. Parameter `-S` is the amount of memory to be allocated, `-o` is the desired order of language model, `corpus` is the input text file and `output.arpa` is the output language model file.

```
$ build/bin/lmplz -S 1G -o 1 <corpus >output.arpa
```

For our first two word weighting schemes, namely the ratio of probabilities and the ratio of logarithmic probabilities, we need a smoothed probability for every single word, including unseen words. We obtain this from a unigram language model trained on the corpus. The language model tool generates a file in ARPA format containing word and logarithmic probability pairs, for which we write a script to parse the logarithmic probabilities of each word easily.

To compute cross entropy, we need to pick a more sophisticated order n for our n-gram language model to represent our data well. This is because we need to consider longer word context, rather than single word frequencies. Choosing small or large n is a trade-off between bias and variance. The reason is that small n leads to underfitting (high bias) while large n leads to overfitting (high variance). Besides, using a larger n means that more memory and storage is needed for generating language model files. The order of language model is not explicitly stated in Wang et al. (2017b)'s research on NMT, but Axelrod et al. (2011) used 4-gram for their experiment on SMT and achieved improvement for domain adaptation. Moreover, the popular BLEU metric usually measures up to 4-gram accuracy for evaluation of fluency and accuracy. Thus we choose to use 4-gram in our project too. We use KenLM's Python module for loading language model file and querying for probabilties and cross-entropy values.

[9] KenLM: https://github.com/kpu/kenlm

5.4 Experiment Results

5.4.1 Romanian-English News-Biomedicine

In this section, we report our experiment results on adapting Romanian-English news to biomedicine. All models are trained using the same configurations except the weighting scheme. Hence the results are comparable. After we notice that using logarithmic word frequency ratio as word weighting (No. 3) has achieved promising improvement, we summed the weights to create a new sentence weighting (No. 7).

We report the BLEU scores on both valid and test sets in Table 5.6. Since we do not perform hyperparameter tuning using the valid set, valid BLEU scores are indicative. However, we only compare performance and make conclusions based on test performance. Vanilla refers to the system without any domain adaptation technique. The baseline is in italic and the systems that beat the baseline are in bold.

Weighting	System No.	System	Valid	Test
none	1	vanilla	22.82	21.85
word	2	word frequency ratio	22.23	21.61
	3	**log word frequency ratio**	**24.11**	**22.70**
	4	distributing sentence	23.17	21.90
sentence	5	*baseline (Wang et al., 2017b)*	*22.62*	*22.13*
	6	increasing domain difference	22.65	22.11
	7	**summing log word frequency ratios**	**23.76**	**22.97**

Table 5.6: Model performance on adapting Romanian-English news to biomedicine

From the results, we can see that sentence weighting (No. 5) does outperform original model without weighting (No. 1), by 0.28 BLEU, but creating a larger difference between in-domain and out-of-domain weights (No. 6) does not improve performance. Hence we stick to the original sentence weights when distributing them over words.

Simply weighting words based on frequency ratio (No. 2) leads to an inferior result, comparing to both vanilla model without weighting (No. 1) and baseline (No. 5). This is expected because the ratio can be exponentially large.

We find that weighting words using the ratio of logarithmic frequency (No. 3) improves 0.57 BLEU comparing to baseline (No. 5). This could justify our choice of using a logarithmic scale on word frequency ratio.

When we sum up the word weights from the ratio of logarithmic frequency, normalised by sentence length, to obtain sentence weights (No. 7), we have the largest improvement of 0.84 BLEU over baseline (No. 5), and 1.12 BLEU over the vanilla system (No. 1). This improvement is three times larger than the 0.28 BLEU improvement of using baseline sentence weighting over the vanilla system.

We have identified two weighting schemes (No. 3 and 7) that achieve better BLEU over baseline sentence weighting, so we will carry on to evaluate the two schemes on English to German task in the next section.

5.4.2 English-German News-TED Talks

In this section, we report our experiment results on EN-DE experiments. It includes a vanilla system, a baseline sentence weighting, our proposed tag-and-weight with different configurations as well as the two best schemes from the previous section.

Weighting	System No.	System	Valid	Test1	Test2
none	1	vanilla	27.57	29.83	**26.50**
word	2	log word frequency ratio	27.76	29.88	**26.37**
sentence	3	*baseline (Wang et al., 2017b)*	*27.76*	*29.88*	*26.19*
	4	**summing log freq ratios**	27.88	**30.04**	**26.36**
tag-and-weight	5	2 tags	26.63	29.78	25.76
	6	**2 tags + sentence weighting**	27.78	**30.50**	**26.88**
	7	10 tags	19.41	25.45	20.56
	8	10 tags + sentence weighting	20.08	24.84	20.66

Table 5.7: Model performance on adapting English-German news to TED talks

From the results, we notice that our baseline (No. 3), sentence weighting based on cross-entropy difference, only results in a slight improvement over the vanilla system (No. 1) on Test1, and even degraded performance on Test2.

Whilst word weighting with logarithmic word frequency ratio (No. 2) does not lead to significant improvement over baseline, our new sentence weighting of summing up such ratio (No. 4) still achieves improvement over baseline, but with a smaller magnitude of around 0.2 BLEU for on Test1 and Test2. This could be explained by our reasoning in previous chapter that summing up word weights can reduce word weighting's aggressiveness. Interestingly, the vanilla system still performs better than this new sentence weighting on Test2.

Regarding our tag-and-weight method, we observe that systems with 10 tags (No. 7 and 8) produce an inferior performance on both test sets, regardless of applying weighting or not. This is probably caused by using insufficient data to model too many domains. On the other hand, whilst 2 tags (No. 5) does not improve translation performance, 2 tags with sentence weighting (No. 6) reaches the best BLEU score among all systems for both tasks. It is on average 0.6 BLEU better than baseline system, and 0.5 BLEU better than the best sentence weighting we proposed (No. 4). It is also the only approach that beat the vanilla system on both test sets.

Also, using 2 tags with sentence weighting beat simply using 2 tags and baseline sentence weighting. Thus, we are able to conclude that our proposed tag-and-weight approach achieves a better domain adaptation over the two state-of-the-art techniques, namely multi-domain NMT and sentence weighting. In the next chapter, we will try to analyse where the improvement in BLEU comes from.

6 | Analysis and Discussion

From Section 5.4, we find that for English to German translation, systems seem to produce inconsistent performances on Test1 and Test2. Hence in the first half of this chapter, we look into the test data and try to find out the reason behind.

Also, three of our proposed techniques generally produce better BLEU score than our baseline, Wang et al. (2017b)'s sentence weighting based on cross-entropy difference. The three techniques are word weighting using logarithmic word frequency ratio, sentence weighting by summing up logarithmic word frequency ratio, and tag-and-weight using 2 tags and sentence weighting. We will analyse the translation outputs to look for the source of improvement in the second half of this chapter.

6.1 Comparing English-German Test Data

We have two test sets for English to German translation, Test1 from IWSLT 2013 test data and Test2 from IWSLT 2014 test data. We observe that systems have inconsistent performances on the two test sets. The vanilla system without any domain adaptation technique has very strong performance on Test2 but not Test1, as shown in Section 5.4. We hypothesise that Test2 is not of a good in-domain quality comparing to Test1.

It is easy to test our hypothesis. In our tag-and-weight approach, we have assigned a tag to each source sentence in all data including test sets, using cross-entropy difference of in-domain and out-of-domain language models. In specific, we simply need to examine the data used for the approach with 2 tags. Then we can compare the percentage of sentences tagged as out-of-domain and in-domain in each test set. This will indicate the in-domain quality of each dataset. We write a script to read test sets, and report the number and percentage of tags for each domain in Table 6.1.

As seen from the table, only 8.8% of the sentences in Test1 are tagged as out-of-domain, while half of Test2 are tagged as to out-of-domain. This clearly implies that Test2 is not good enough with regard to being in-domain. Our hypothesis holds true.

	Test1	Test2
Number (%) of in-domain tags	906 (91.2%)	652 (50.0%)
Number (%) of out-of-domain tags	87 (8.8%)	653 (50.0%)
Number of all tags	993	1305

Table 6.1: Number of in-domain and out-of-domain tags in two test sets for 2-tag tag-and-weight experiment on EN-DE. We used language models trained on training data for tagging.

It is thus sensible to see that the vanilla system results in better performance than many domain-adapted systems on Test2. Such a vanilla system is inclined to out-of-domain data due to its overwhelming proportion (95.9%) in training data, as presented in Section 5.1.2. We also conclude that it is also not a good choice to use Test2 to measure domain adaptation effort because it contains an equal amount of in-domain and out-of-domain data.

A notable observation is that our tag-and-weight system, using 2 tags and sentence weighting, reaches the best BLEU for both Test1 and Test2. Despite being domain adapted, its performance on Test2, which is regarded as mixed-domain, is still better than vanilla. This implies that our system has the capability of differentiating domains, rather than simply being adapted to one domain only. Since simply using 2 tags did not achieve such a good outcome, we regard our attempt to combine multi-domain NMT and sentence weighting as a remarkable success.

6.2 Comparing Machine Translations with Reference

In Section 5.4, some of our proposed approaches have resulted in higher BLEU than baseline. We are curious to know what characteristics or measurements of the translations have improved.

6.2.1 Recall of Named Entities

Translation of named entities is hard but important for domain adaptation, for the reason that named entities are representative of a domain, but the words are infrequent (Li et al., 2018). NMT is poor at dealing with less common words because maximum likelihood estimation puts more than proportionate prediction probabilities on common words (Gong et al., 2019). We hypothesise that through weighting in-domain words/sentences more, the rare named entities can get a higher chance to be recalled.

Hence, we are interested in knowing each system's recall percentage of named entities. We use the method described in Currey et al. (2017)'s work to calculate such percentage[1]. First, they record words that are identical in source and reference sentence pairs in test data, neglecting one-character tokens (e.g. punctuation marks) and case of text. They then compute the percentage of these words occurring in the corresponding output sentences.

There is a limitation to this method that not all named entities stay unchanged in source and target sentences. For example, English names are phonetically converted to Chinese characters (Wan and Verspoor, 1998) during translation. However, this problem is mitigated to some extent in our data settings. All involved languages (Romanian, English, German) are Latin/Roman characters based, so we assume more names and acronyms can be preserved across languages. Thus we stick to their method and report recall in Table 6.2. Tag-and-weight is not evaluated using RO-EN task, so it does not have a recall percentage. Also, We use Test1 as the only reference for EN-DE because we have found earlier that Test2 is not representative of our desired domain.

	RO-EN	EN-DE
vanilla	66.1%	85.8%
baseline (Wang et al., 2017b)	*65.8%*	*87.2%*
word weighting log frequency ratio	**66.9%**	85.8%
sentence weighting sum of log frequency ratio	**67.2%**	**87.6%**
tag-and-weight 2 tags and weighting	N/A	**88.2%**

Table 6.2: Recall of named entities for top performing systems

From the results, we find that all novel weighting schemes proposed by us have better named entities recall, except for word weighting in EN-DE. The result looks promising, so we are able to confirm that through weighting in-domain sentences or words more, the domain-specific named entities can be more easily produced in translation.

6.2.2 Average Length and Logarithmic Type-Token Ratio

Similar to Section 5.1, we compare average sentence length and logarithmic type-token ratio of output translations with those of baseline and reference translations.

[1] Currey et al. (2017) named it "pass-through accuracy", but it is recall in Information Retrieval.

The output translations are the output from three techniques, word weighting using logarithmic word frequency ratio, sentence weighting by summing up logarithmic word frequency ratio, and tag-and-weight with 2 tags. We have EN-DE output translation for all three schemes and RO-EN for the first two, because the last scheme was not evaluated using RO-EN task. Also, as we have discussed in the previous section that EN-DE's Test2 is not of high quality, so we only use Test1 as the reference for EN-DE. The statistics of translations is presented in Table 6.3. Note that it is not meaningful to compare across the two tasks.

	RO-EN		EN-DE	
	avg length	log TTR	avg length	log TTR
reference (test, target)	11.77	0.877	16.80	0.881
vanilla	12.45	0.873	17.01	0.875
baseline	*12.53*	*0.872*	*17.02*	*0.875*
word weighting: log frequency ratio	12.34	0.871	17.04	0.875
sentence weighting: sum of log frequency ratio	12.36	0.871	16.99	0.875
tag-and-weight: 2 tags and weighting	N/A	N/A	17.07	0.874

Table 6.3: Statistics of references and output translations from top-performing systems. Log TTR denotes logarithmic type-token ratio.

From the table, we observe that all systems tend to produce longer translations than desired, which is expected because out-of-domain data are much longer than in-domain data in both tasks. Also, the logarithmic type-token ratio drops for all systems. This means that NMT systems, in general, produce a smaller vocabulary than reference. This observation agrees with Gong et al. (2019)'s claim on NMT's bias towards common words.

Our novel word and sentence weighting schemes have some effect on regularising the output sentence length for RO-EN, but not too obvious. In contrary, our proposed tag-and-weight causes a slightly worsened sentence length for EN-DE pair. Besides, there is no improvement in logarithmic type-token ratio from any of the approaches, comparing to the vanilla system.

Hence we arrive at the conclusion that the two metrics are not improved by our proposed methods, even though they result in higher BLEU. A possible explanation is that our methods are centred around weighting probabilities of words, which helps little on sentence length and lexical diversity.

7 | Conclusion

7.1 Summary

In this project, we first introduce neural machine translation, corpora, evaluation and the problem of domain mismatch. Then we review state-of-the-art domain adaptation approaches, with a focus on sentence weighting based on cross-entropy difference. With a thorough understanding of the advantages and limitations of current approaches, we contribute to the research on sentence and word level cost weighting.

We improve sentence weighting by enlarging the difference between in-domain and out-of-domain weights. We also design a new tag-and-weight method that combines state-of-the-art multi-domain NMT and sentence weighting. Furthermore, we propose novel word level weighting schemes, leveraging word frequencies and language model scores.

Our approaches are evaluated using two tasks. The first task is translating from Romanian to English and adapting news data to biomedical text, and the second task is translating from English to German and adapting news data to TED talks. BLEU scores from both tasks indicate that three of our approaches produce promising results. The top-performing system, tag-and-weight using 2 tags and sentence weighting, achieves significant improvement of 0.6 BLEU and 0.7 over both sentence weighting and multi-domain system.

Following our in-depth quantitative and qualitative analysis, we find that the two test sets for English to German task are not of consistent quality. The one taken from IWSLT 2014 is not representative of in-domain data. Moreover, regarding the source of improvement in BLEU, average sentence length and logarithmic token-type ratio do not seem to improve (to get closer to reference data) for any system. Nonetheless, our proposed approaches are significantly better at recalling named entities comparing to vanilla and baseline systems.

7.2 Future Work

Our experiments show that word-level weighting has promising performance comparing to sentence weighting. However, due to constrained time and resource, we could do neither hyperparameter tuning nor trying out more word-level weighting schemes. Our word weighting schemes are inspired by TF-IDF and language models. Future work can explore other metrics like distributing Latent Dirichlet allocation (Blei et al., 2003) scores at sentence-level.

Also, Behnke (2018)'s dynamic word weighting can be used for domain adaptation. It can be incorporated in a way similar to fine-tuning and mixed fine-tuning introduced in Section 2.7, where in-domain weights gradually increase until model converges. It is also reasonable to resemble learning rate scheduling techniques, like cosine annealing with warm restarts (Loshchilov and Hutter, 2016). Dynamic weighting can have two potential advantages, to regularise the attention paid to in-domain words, as well as to help a model to recover from local minimum during training.

Furthermore, we find that both tag-and-weight and sentence weighting by summing up logarithmic word frequency ratios achieve a better recall for rare named entities. This can be seen as a regularisation effect on the bias on common words caused by maximum likelihood estimation (Gong et al., 2019). Thus it is sensible to apply our techniques on not only domain adaptation problems, but also general neural machine translation. It is possible that weighting, especially direct word weighting can prevent vocabulary from shrinking too much through translation, by putting more than proportionate weights on rare words.

Finally, we are also interested in comparing word-level weighting to how modern NMT architectures like the attention mechanism model individual words. Behnke (2018) provided an insightful starting point that weighting source side words for each target word could have a similar effect as attention mechanism. We suggest that, if an attention-equivalent word weighting can be invented, the whole neural machine translation community will benefit from it. The reason is indisputable, that the attention mechanism adds complexity to a model and consumes extra training resources, whereas word-level weighting is interpretable and computationally cheap.

Bibliography

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2018). Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 355–362.

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*.

Behnke, M. (2018). *Improving Neural Machine Translation Training via Weighted Objectives*. Master's thesis, School of Informatics, The University of Edinburgh, Edinburgh, UK.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*, 3:1137–1155.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas. Association for Computational Linguistics.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017). Findings of the 2017 conference on machine

translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 conference on machine translation (wmt16). In *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 131–198. Association for Computational Linguistics.

Brown, P. F., Pietra, V. J. D., Mercer, R. L., Pietra, S. A. D., and Lai, J. C. (1992). An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40.

Chen, B., Cherry, C., Foster, G., and Larkin, S. (2017). Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 40–46, Vancouver. Association for Computational Linguistics.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Chu, C., Dabre, R., and Kurohashi, S. (2017). An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391. Association for Computational Linguistics.

Chu, C. and Wang, R. (2018). A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1304–1319. Association for Computational Linguistics.

Currey, A., Miceli Barone, A. V., and Heafield, K. (2017). Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark. Association for Computational Linguistics.

Domhan, T. and Hieber, F. (2017). Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505. Association for Computational Linguistics.

Dorr, B., Olive, J., McCary, J., and Christianson, C. (2011). Machine translation evaluation and optimization. In Olive, J., Christianson, C., and McCary, J., editors, *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*, chapter 5, pages 745–843. Springer, New York, NY.

Etchegoyhen, T., Fernández Torné, A., Azpeitia, A., Martínez Garcia, E., and Matamala, A. (2018). Evaluating domain adaptation for machine translation across scenarios. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

François, T. and Fairon, C. (2012). An "AI readability" formula for french as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 466–477, Jeju Island, Korea. Association for Computational Linguistics.

Freitag, M. and Al-Onaizan, Y. (2016). Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*.

Gamon, M., Aue, A., and Smets, M. (2005). Sentence-level mt evaluation without reference translations: Beyond language modeling. In *European Association for Machine Translation (EAMT)*.

Gong, C., Tan, X., He, D., and Qin, T. (2019). Sentence-wise smooth regularization for sequence to sequence learning. *AAAI*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *CoRR*, abs/1410.5401.

Gulcehre, C., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H.-C., Bougares, F., Schwenk, H., and Bengio, Y. (2015). On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified

Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Hieber, F. and Domhan, T. (2017). Train neural machine translation models with sockeye. Retrieved 06 March 2019 from https://aws.amazon.com/blogs/machine-learning/train-neural-machine-translation-models-with-sockeye.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jelinek, F., Mercer, R. L., Bahl, L. R., and Baker, J. K. (1977). Perplexity–a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A. F. T., and Birch, A. (2018). Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

Kettunen, K. (2014). Can type-token ratio be used to show morphological complexity of languages? *Journal of Quantitative Linguistics*, 21(3):223–245.

Khayrallah, H., Kumar, G., Duh, K., Post, M., and Koehn, P. (2017). Neural lattice search for domain adaptation in machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 20–25. Asian Federation of Natural Language Processing.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT.

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 transla-

tion. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Koeva, S., Rizov, B., Tarpomanova, E., Dimitrova, T., Dekova, R., Stoyanova, I., Leseva, S., Kukova, H., and Genov, A. (2012). Bulgarian-english sentence-and clause-aligned corpus. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2), Lisbon*, pages 51–62.

Kwon, O.-W., Choi, S.-K., Lee, K.-Y., Roh, Y.-H., and Kim, Y.-G. (2009). Customizing an english-korean machine translation system for patent/technical documents translation. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 2*, volume 2.

Lample, G., Denoyer, L., and Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *CoRR*, abs/1711.00043.

LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Li, Z., Wang, X., Aw, A., Chng, E. S., and Li, H. (2018). Named-entity tagging and domain adaptation for better customized translation. In *Proceedings of the Seventh Named Entities Workshop*, pages 41–46, Melbourne, Australia. Association for Computational Linguistics.

Loshchilov, I. and Hutter, F. (2016). SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983.

Luong, M.-T. and Manning, C. D. (2015). Neural machine translation systems for spoken language domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation*, pages 76–79, Da Nang, Vietnam.

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

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recur-

rent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Mozer, M. (1995). A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3.

Nagao, M. (1984). A framework of a mechanical translation between japanese and english by analogy principle. *Artificial and human intelligence*, pages 351–354.

Nirenburg, S. (1989). Knowledge-based machine translation. *Machine Translation*, 4(1):5–24.

Olah, C. and Carter, S. (2016). Attention and augmented recurrent neural networks. *Distill*.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Park, J., Song, J., and Yoon, S. (2017). Building a neural machine translation system using only synthetic parallel data. *arXiv preprint arXiv:1704.00253*.

Pawar, T. (2017). Language modeling using recurrent neural networks. Retrieved 03 March 2019 from https://medium.com/praemineo/language-modeling-using-recurrent-neural-networks-part-1-427b165576c2.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. Association for Computational Linguistics.

Priddy, K. L. and Keller, P. E. (2005). *Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering, Vol. TT68)*. SPIE- International Society for Optical Engineering.

Rafalovitch, A. and Dale, R. (2009). United nations general assembly resolutions: a

six-language parallel corpus. In *MT Summit XII proceedings*. International Association of Machine Translation.

Resnik, P. and Smith, N. A. (2003). The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press, Cambridge, MA, USA.

Salton, G. and McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Sennrich, R., Haddow, B., and Birch, A. (2016a). Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016b). Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*, pages 371–376.

Sennrich, R., Haddow, B., and Birch, A. (2016c). 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. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016d). 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. Association for Computational Linguistics.

Skadina, I. and Pinnis, M. (2017). Nmt or smt: Case study of a narrow-domain english-latvian post-editing project. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 373–383. Asian Federation of Natural Language Processing.

Stymne, S., Tiedemann, J., Hardmeier, C., and Nivre, J. (2013). Statistical machine translation with readability constraints. In *Proceedings of the 19th Nordic Conference of Computational Linguistics, NODALIDA 2013, May 22-24, 2013, Oslo University, Norway*, pages 375–386.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

van der Wees, M., Bisazza, A., and Monz, C. (2017). Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Wan, S. and Verspoor, C. M. (1998). Automatic english-chinese name transliteration for development of multilingual resources. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 1352–1356, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wang, R., Finch, A., Utiyama, M., and Sumita, E. (2017a). Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 560–566. Association for Computational Linguistics.

Wang, R., Utiyama, M., Liu, L., Chen, K., and Sumita, E. (2017b). Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017 September 11, 2017*, pages 1482–1488.

Weitzman, M. (1971). How useful is the logarithmic type/token ratio? *Journal of Linguistics*, 7(2):237–243.

Williams, P., Sennrich, R., Post, M., and Koehn, P. (2016). Syntax-based statistical machine translation. *Synthesis Lectures on Human Language Technologies*, 9(4):1–208.

Yu, D., Deng, L., and Dahl, G. (2010). Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition. In *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2019). *Dive into Deep Learning*, page 319. Unpublished draft. Retrieved 03 March 2019 from https://en.d2l.ai/d2l-en.pdf.

Zhang, S. and Xiong, D. (2018). Sentence weighting for neural machine translation domain adaptation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3181–3190.

Ziemski, M., Junczys-Dowmunt, M., and Pouliquen, B. (2016). The united nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA).