TÉCNICO LISBOA

Improving the Quality of Neural Machine Translation

Pedro Miguel Lobato Ferreira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Dr. André Filipe Torres Martins

Examination Committee

Chairperson: Prof. Dr. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Dr. André Filipe Torres Martins
Member of the Committee: Prof. Dr. Bruno Emanuel Da Graça Martins

November 2018

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First of all I would like to thank my supervisor, Prof. André Martins, for all the support and meaningful discussions throughout this thesis. Having the chance of working with you allowed me to learn much more about the field of NLP than it would have been possible in any other circumstances. Furthermore, I would like to thank Unbabel, for letting me develop this thesis in a challenging, and intellectually rewarding environment. More in particular, I would like to thank both the Applied AI and the AI Research team, for all the shared knowledge throughout this time.

I would also like to thank my family for all the support, and my close group of friends at Técnico, who made this experience much more enjoyable. In particular, I would like to thank Miguel Gameiro, with whom I had the chance of working in several projects throughout this journey, in what proved to be a very gratifying experience.

Last, but not least, I want to thank Isabel, for being such a positive influence in my life and a motivation to outdo myself everyday.

This work was supported by the Fundação para a Ciência e Tecnologia through contract CMU-PERI/TIC/0046/2014 (GoLocal).

Resumo

Avanços recentes levaram a que a tradução automática neural se tornasse a principal abordagem ao problema de tradução automática. Ainda assim, apesar dos modelos actuais produzirem traduções fluentes, estas nem sempre são adequadas, prejudicando a sua qualidade. Nesta dissertação abordamos duas possíveis causas de traduções não adequadas: má cobertura das palavras durante a tradução, o que pode levar a repetições e a que palavras não sejam traduzidas; existência de palavras raras e frases fora de um certo domínio.

De modo a reduzir os problemas de cobertura propomos um modelo baseado na fertilidade das palavras, juntando este conceito a funções de transformação esparsas e restritas do mecanismo de atenção. São ainda apresentadas duas métricas automáticas com o intuito de avaliar os problemas de sobre e sub-tradução de palavras da frase original.

Quanto aos problemas de palavras raras e desajuste de domínio, seguimos uma abordagem existente na literatura que usa o conceito de unidade de tradução para guiar a parte de descodificação dos modelos de tradução automática neural. Este modelo é melhorado, introduzindo correções para problemas identificados, e aplicado ao problema de adaptação de domínio, algo que não tinha sido tentando até agora.

Por fim, avaliamos empiricamente os métodos propostos em três pares de linguagens e apresentamos uma análise extensa de erros, tornando possível compreender os pontos fortes e fracos de cada método e como melhorá-los no futuro.

Palavras-chave: Aprendizagem profunda, processamento de língua natural, tradução automática neural, funções de transformação do mecanismo de atenção, adaptação de domínio

Abstract

Over the last few years, neural machine translation has become the major approach to the problem of automatic translation. Nonetheless, even though current models are able to output fluent translations, they often lack adequacy. In this thesis we target adequacy issues with two different causes: poor coverage of source words during translation, which lead to unnecessary repetitions and erroneously untranslated words; rare words and out-of-domain sentences.

In order to mitigate coverage issues we propose a fertility-based approach to neural machine translation, which couples the concept of fertility with sparse and constrained attention transformations. Furthermore, we present two machine translation metrics that allow us to understand how much the problems of over and under-translations are affecting the model.

To deal with rare words and out-of-domain sentences, we implement an existing approach in the literature, that makes use of the concept of translation pieces to guide the decoding step of NMT models. We further extend this method by solving some identified issues, and by applying it to the problem of domain adaptation, something which had not been done in the past.

Finally, we provide an empirical evaluation in three language pairs, presenting an extensive error analysis. This makes it possible to understand the strengths and weaknesses of each of the models, and how they may be improved in the future.

Keywords: Deep learning, natural language processing, neural machine translation, attention transformations, domain adaptation

Contents

List of Tables

List of Figures

Acronyms

BPE Byte Pair Encoding.

GRU Gated Recurrent Unit.

LSTM Long Short-Term Memory.

MLP Multi-Layer Perceptron.

NMT Neural Machine Translation.

RNN Recurrent Neural Network.

SMT Statistical Machine Translation.

Chapter 1

Introduction

1.1 Motivation and Related Work

Living in an ever more global society has increased the necessity of communicating with people from all over the world. Even though some global languages have been established, the comfort of one with its own native(s) language(s) is something hard to surpass. This leads to the necessity of having people be understood in several languages. Human translators are one way of achieving a better understanding between people, but depending on a third party is something that might not always be possible, both due to time and availability constraints, nor it is always possible to disclose a conversation to a third party. With this in mind, automatic machine translation would be able to close this gap.

One of the first impactful approaches to the problem of machine translation was Statistical Machine Translation (SMT), which, as the name implies, approaches the problem from the statistical point of view. One of the pioneer works on the field (Brown et al. [1]), was based on the premise that it is possible to automatically create translation models between two languages. Broadly speaking, translation models would behave like a dictionary, learning the mappings between the different words of two languages by making use of a corpus of parallel translated data. Throughout time, this was made possible by five IBM Models, each one introducing different intricate features. While being helpful, a simple word-based approach had several weaknesses, namely, being unable to distinguish homonyms and allowing only many-to-one alignments. A stronger approach to SMT to appear later was *phrase-based translation* (Koehn et al. [2], Marcu and Wong [3]). This approach introduced the concept of phrase tables, enabling many-to-many alignments, by learning to map longer structures between languages, and not only words. As a much stronger take on the problem of machine translation, it became the most widely used approach for several years.

Increasing computational power, and the possibility of dealing with massive amounts of data, allowed neural networks (McCulloch and Pitts [4], Goodfellow et al. [5]) to become relevant once again leading to the so called Deep Learning paradigm (LeCun et al. [6]). The versatility of this class of algorithms allowed to approach MT from a different perspective, leading to Neural Machine Translation (NMT) (Kalchbrenner and Blunsom [7], Cho et al. [8], Sutskever et al. [9]). This approach introduced some

highly sought characteristics: NMT models allow long-term dependencies on sentences to be captured; easier decoding procedures and a simplified sequence of steps; models can be conveniently trained end-to-end (Luong [10]). Even though Recurrent Neural Networks (RNN) (Elman [11]) using Long Short-Term Memory (LSTM) cells (Hochreiter and Schmidhuber [12]), or Gated Recurrent Unit (GRU) (Cho et al. [8]), are the most common approaches, the utilization of convolutional neural networks (CNNs) has also been proposed (Gehring et al. [13]). Neural machine translation can also include an attention mechanism (Bahdanau et al. [14], Luong et al. [15]), that allows models to learn which source sentence words should be attended in order to generate a given target language word. In particular, some models built on top of this attention mechanism have also been proposed, achieving strong performances (Vaswani et al. [16]). Finally, another staple of nowadays NMT are subword units (Sennrich et al. [17]), which help overcome the issue of rare words.

Despite the recent improvements in the overall quality of NMT there are still some areas of concern. Namely, current NMT systems are able to output *fluent* sentences that often lack *adequacy*, i.e., are not able to convey the idea of the source sentence (Snover et al. [18]). Adequacy errors may be split into three different major types of mistakes (Tu et al. [19]):

- *Over-translations*, meaning that some source words are translated more often than they should.
- *Under-translations*, when some source words are erroneously untranslated.
- *Mistranslations*, when a given source word is attended, but the output word is not correct.

The last two points are the ones more obviously related with adequacy mistakes. Nonetheless, it is also possible that repetitions cause no harm to fluency, while decreasing the adequacy of the sentence. Both over and under-translation are related with poor coverage of the source words during translation. Furthermore, the problem of mistranslations may be related with the inability of NMT systems to perform well when translating rare words and out-of-domain sentences (Arthur et al. [20], Koehn and Knowles [21]).

To further illustrate the mentioned mistakes, we present two examples in Table 1.1, produced by current NMT approaches. In the first example, the source word *"online"* was not translated, making the obtained translation miss the original idea of the sentence, while remaining fluent. In Example 2, the output is fluent, but the translation fails to convey the intended meaning. In this case the differences are even more striking.

Table 1.1: NMT Examples with Poor Adequacy. The golden color indicates the portion of the reference that the example targets, the **red color** indicates the mistranslations/repetitions and finally, the *blue color* highlights correct translations of the parts of the sentence being focused. The <**MISS**> symbol refers to portions of the sentence that were missed.

Example 1	
Reference	And this is one of the main three sources of online problems today.
Translation	And this is one of the three main sources of existing <**MISS**> problems today.
Example 2	
Reference	Do not let the tablet dissolve in your mouth.
Translation	Do **you respect the fact that the taboo is not already** in the mouth.

In the current literature it is possible to find different approaches to the problem of improving adequacy. Tu et al. [19] and Mi et al. [22] target the problem of adequacy by adapting the concept of coverage to NMT. Namely, both introduce a coverage vector, with the goal of keeping track of which parts of the source sentence have already been attended. In a different approach, Tu et al. [23] proposed using a gating mechanism to decide how much information flows from either the encoder or the decoder, based on the premise that both affect fluency and adequacy differently. Another possible way of ensuring adequacy, is by training the model to be able to reconstruct the original sentence from the obtained translations (Tu et al. [24]). All the previous approaches have one thing in common, namely, they introduce some kind of artifact that is used during training. On the contrary, Wu et al. [25] introduces two terms which re-score the candidate translations during beam search: coverage penalty, which penalizes hypotheses that have source words unattended; and length normalization, that ensures the model does not produce sentences which are too short.

Other works in the literature, that try to include external knowledge into NMT models, are more concerned with rare words, or making sure that the translations are closer to the desired. These approaches range from lexical constraints and biases (Arthur et al. [20], Hokamp and Liu [26], Chatterjee et al. [27]), to strategies that try to find examples similar to the input sentence to re-train the model (Farajian et al. [28], Li et al. [29]), that leverage information about the hidden states and the output distribution of the closest examples (Gu et al. [30]), or that make use of a dictionary-like memory (Feng et al. [31]). For a more comprehensive view on the mentioned state-of-the-art work, refer to Section 2.5.

Despite the current approaches to the problem of adequacy in NMT, some questions still remain as open problems. Namely,

- How can we avoid the problem of unnecessary repetitions in NMT?
- How can we make sure that every source word is attended during translation?
- How can we make NMT models more robust to domain changes and rare words?

1.2 Contributions

The main contributions of this thesis are the following:

- We propose two new metrics that aim to measure how much of a problem over-translations and under-translations are in the output sentences. Namely, these metrics are able to capture information that the commonly used automatic metrics miss. They are described in Chapter 3 and available online.[1]
- We introduce a new sparse constrained transformation, *Constrained Sparsemax*, that is able to produce upper bounded sparse probability distributions, while being end-to-end differentiable (Section 4.1).

[1] Code available online at: `https://github.com/PedroMLF/MT-Adequacy-Metrics`

- We introduce a new approach to NMT, *fertility-based NMT* (Chapter 4). It pairs sparse and constrained attention transformations with the fertility concept to promote better coverage of the source words, both during training and inference. By enhancing the coverage of the model, fertility-based NMT should decrease the adequacy issues related with over and under-translations.[2]

- We implement a model based on the one proposed by Zhang et al. [32], with the respective changes introduced in Chapter 5. We later use this model in the same context of the original paper (in-domain translation), and further evaluate its performance in a domain adaptation scenario. This approach should help mitigate the identified problem of mistranslations.[3]

- We perform a comprehensive empirical analysis in two language pairs for each of the experiments. Furthermore, we conduct an extensive error analysis, providing a better intuition on the strengths and weaknesses of each of the proposed approaches (Chapter 6).

In the course of this thesis, part of the presented work was also published in the *2018 Annual Meeting of the Association for Computational Linguistics (ACL)*, under the name *"Sparse and Constrained Attention for Neural Machine Translation"* (Malaviya et al. [33]), available online at: `http://aclweb.org/anthology/P18-2059`.

1.3 Document Outline

This dissertation starts by addressing the theoretical foundations that serve as background to the proposed work. In Chapter 2, a brief introduction to neural networks is first presented (Section 2.1). Then, the concept of language models is introduced in Section 2.2, leading to neural machine translation (Section 2.3). This is followed by an introduction to two relevant activation functions for this work (Section 2.4). Afterwards, a comprehensive overview of the state-of-the-art is presented, with regard to adequacy issues and solutions in NMT (Section 2.5). Finally, the last two sections present the machine translation automatic metrics to be used (Section 2.6), and the concept of fertility (Section 2.7).

Having established the necessary background, the three main contributions of this dissertation are discussed. First, Chapter 3 introduces two automatic metrics concerned with the problem of adequacy in NMT, with regard to over and under-translations. Then, Chapter 4 introduces fertility-based NMT as an alternative approach to the existing ones. The main characteristic of this approach is its ability to leverage sparse and constrained attention transformations, producing translations where the adequacy issues are mitigated. Finally, in Chapter 5 it is used as starting point the work from Zhang et al. [32]. This model, which we call guided-NMT, is improved based on some identified weaknesses and it is further extended to be used for the problem of domain adaptation. Furthermore, both fertility-based NMT and guided-NMT are then evaluated over three sets of experiments in Chapter 6. Finally, Chapter 7 introduces some future considerations for further improvements upon the presented work, as well as the major takeaways from this dissertation.

[2]Code available online at: `https://github.com/Unbabel/sparse_constrained_attention`
[3]Code available online at: `https://github.com/PedroMLF/guided-nmt`

Chapter 2

Background

This chapter introduces the key theoretical concepts that will be used throughout this work. First, a brief introduction to neural networks will be presented. Then, language models will be used as a means of introducing the problem of NMT, and several of its main components. In addition, the existing activation functions in the literature, as well as the main approaches to the problem of adequacy in NMT, are presented. Finally, two commonly used metrics in MT are discussed, as well as the concept of fertility.

2.1 Neural Networks

In this section neural networks will be presented as a means of defining a mapping between a given input $\boldsymbol{x} \in \mathbb{R}^N$ and an output y, which may take many forms. In this work the focus will be on simpler neural networks, that have no feedback connections from the output to the input (feedforward neural networks), and on neural networks where such feedback loops occur (recurrent neural networks).

2.1.1 Feedforward Neural Networks

The natural precursor to feedforward neural networks are linear models. A linear model makes it possible to obtain some output y for a given input $\boldsymbol{x} = (x_1, ..., x_N)$, by means of an affine transformation, i.e., the intern product between $\boldsymbol{x}$ and a vector of weights $\boldsymbol{w}$, followed by the addition of a bias term. Each individual weight represents the importance of the corresponding entry for the yielded output. In summary,

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{i=1}^{N} w_i x_i + b. \tag{2.1}$$

A flaw of such a model is the fact that not all problems have linearly separable points in a given feature space. A classic example that illustrates such a limitation is the impossibility to devise a linear model able to solve the exclusive-or boolean operator (XOR), depicted in Figure 2.1.

This highlights the necessity of introducing a different approach to the problem, capable of handling non-linear cases. A possible way of doing so is through the Multi-Layer Perceptron (MLP), a class of

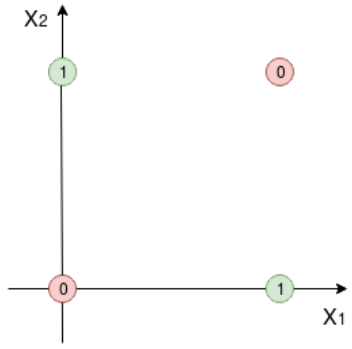


Figure 2.1: XOR representation. Label y is represented by the numbers 0 and 1.

feedforward neural networks. The main difference between the multi-layer perceptron, and the aforementioned linear models, is the inclusion of an intermediate hidden layer. This hidden layer is simply an affine transformation followed by a non-linear function, $g(.)$. Figure 2.2 shows a single-layer MLP, where the described behavior may be observed. In fact, MLPs may have as many hidden layers as necessary. More formally, assuming a MLP with M hidden layers, it is possible to write

$$\begin{aligned} \boldsymbol{h_1} &= g\left(\boldsymbol{W}^{(1)^T}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) \\ &\dots \\ \boldsymbol{h}_m &= g\left(\boldsymbol{W}^{(m)^T}\boldsymbol{h}_{m-1} + \boldsymbol{b}^{(m)}\right) \\ &\dots \\ y &= g\left(\boldsymbol{W}^{(M+1)^T}\boldsymbol{h}_M + \boldsymbol{b}^{(M+1)}\right), \end{aligned} \tag{2.2}$$

where $\boldsymbol{x} \in \mathbb{R}^N$, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{N \times D_1}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^{D_1}$, $\boldsymbol{W}^{(m)} \in \mathbb{R}^{D_{m-1} \times D_m}$, $\boldsymbol{b}^{(m)} \in \mathbb{R}^{D_m}$, and the functions $g(.)$ are the aforementioned non-linear functions (sometimes also called activation functions). The choice of non-linear function depends on whether it yields yet another hidden state $\boldsymbol{h}$, or the output y. Between hidden layers, the most common activation functions are the rectified linear unit (Glorot et al. [34]), and the hyperbolic tangent. The non-linearity that leads to the output varies with the problem at hands. For a binary classification problem, it is common to use the sigmoid function, defined as

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{2.3}$$

Since the sigmoid functions yields a value between 0 and 1, it is straightforward to define a threshold at 0.5 to separate both possible classes. On the other hand, if the problem requires multi-class classification, i.e., it requires obtaining a probability distribution over a set of N classes, softmax is the usual choice. Formally, it is defined as

$$\text{softmax}(\boldsymbol{z})_n = \frac{\exp(z_n)}{\sum_{j=1}^{N} \exp(z_j)}, \tag{2.4}$$

where the input vector $\boldsymbol{z}$, corresponds to the logits, i.e., a vector of non-normalized scores.

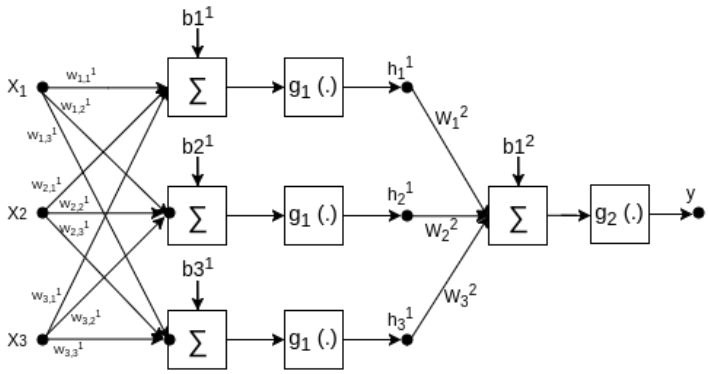


Figure 2.2: Representation of a 1-layer MLP.

Looking at the formulation presented in Equation 2.2, one may clearly see that feedforward neural networks may be interpreted as a chain of functions (not necessarily the ones presented for the MLP) that for a given input are able to calculate some output (Neubig [35]). Moreover, the information flows from the input layer, throughout the hidden layers, until the output layer is reached.

Training

Having described feedforward neural networks, it is now time to clarify the method of training such a model, i.e., adjusting the matrices of weights that multiply the multiple inputs throughout time so that the output of the neural network is the correct one. The first step of training is to define a loss function, L. This choice is highly dependent on the type of problem being solved. A common choice for classification problems is the *cross-entropy error*, defined by

$$L(f(\boldsymbol{x};\boldsymbol{\theta}),\boldsymbol{y}) = H(f(\boldsymbol{x};\boldsymbol{\theta}),\boldsymbol{y}) = -\sum_{i=0}^{n} y_i \log(f_i(\boldsymbol{x};\boldsymbol{\theta})), \tag{2.5}$$

where $\boldsymbol{\theta}$ are the parameters of the model, $H(.)$ represents the cross entropy, $f_i(\boldsymbol{x};\boldsymbol{\theta})$ is the model prediction for the *i*-th of the n classes and y_i is the corresponding reference value. Then, it is necessary to update the gradients of the network. The standard method to do so is to use the back-propagation algorithm (Rumelhart et al. [36]). The general idea of this algorithm is to back-propagate the error throughout the network, from the output to the initial layer, adjusting the weights so that the gradient of the error is reduced, and the predictions of the network improve over training time. This may be seen as a process with two phases: one first phase, in which the gradients of the loss with respect to the several parameters of the network are obtained; a second and final phase, where the weights are updated using the chosen optimization technique. These optimization techniques range from the simpler stochastic gradient descent (LeCun et al. [37]), to the more intricate Adam optimizer (Kingma and Ba [38]). A simple algorithm to train a neural network is summarized in Algorithm 1(Goldberg [39]).

Algorithm 1 Simple neural network training algorithm

1: **Input:** Training set with inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_d$ and outputs $\boldsymbol{y}_1, ..., \boldsymbol{y}_d$
2: **Input:** Feedforward neural network model, $f(\boldsymbol{x_i}; \boldsymbol{\theta})$
3: **Input:** Loss function, L
4: **while** Stopping criteria is not met **do**
5: Sample training pair, $\boldsymbol{x}_i, \boldsymbol{y}_i$
6: Compute the loss, $L(f(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{y}_i)$
7: Obtain the gradients of the loss w.r.t to $\boldsymbol{\theta}$, $\Delta_\theta L$
8: Use the obtained gradients and the chosen optimization technique to update the parameters, $\boldsymbol{\theta}$
9: **Output:** $\boldsymbol{\theta}$

2.1.2 Recurrent Neural Networks

In this section a specific architecture of neural networks will be presented. Recurrent neural networks (RNNs) (Elman [11]) are a class of neural networks of particular interest in neural machine translation, as well as in natural language processing problems in general, due to their recurrent nature. In particular, RNNs are able to handle inputs of arbitrary size (such as sentences with different lengths) while capturing relations between the different components of the input (such as relations between different words of a sentence). The main difference between the aforementioned feedforward networks and RNNs is the introduction of the previous hidden state, in addition to the regular input, at each time step's calculation. Therefore, hidden states are now defined as

$$\boldsymbol{h}_t = \begin{cases} g(\boldsymbol{W}^x \boldsymbol{x}_t + \boldsymbol{W}^h \boldsymbol{h}_{t-1} + \boldsymbol{b}^h) & \text{t} \geq 1, \\ \text{Some initialization} & \text{otherwise.} \end{cases} \tag{2.6}$$

where $\boldsymbol{W}^x \in \mathbb{R}^{d_x \times d_h}$, $\boldsymbol{W}^h \in \mathbb{R}^{d_h \times d_h}$, $\boldsymbol{b}_h \in \mathbb{R}^{d_h}$, $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ and $\boldsymbol{h}_t, \boldsymbol{h}_{t-1} \in \mathbb{R}^{d_h}$. The fact that at each time step, both the current input and the previous hidden state are fed into the calculation of the current hidden state and consequently, of the output, allows information from previous inputs to be incorporated and passed to next states. This leads to the conclusion that RNNs are indeed able to capture long term dependencies in the data. It is also possible to yield predictions at each RNN step, by applying an affine transformation followed by a non-linearity. For natural language processing problems it is common to have a discrete output space, thus, a convenient choice for the non-linearity is the softmax (Equation 2.4). So, at each time step, it is possible to predict an output as

$$\boldsymbol{y}_t = \text{softmax}(\boldsymbol{W}^y \boldsymbol{h}_t + \boldsymbol{b}^y). \tag{2.7}$$

where, $\boldsymbol{W}^y \in \mathbb{R}^{d_h \times d_y}$, $\boldsymbol{b}^y \in \mathbb{R}^{d_y}$, $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$, $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$, and the softmax activation function is used to project the logits into a probability space. The described process may be visualized in Figure 2.3.

Training

As with feedforward neural networks, it is necessary to have a mechanism that adjusts the weights of recurrent neural networks, so that the model learns to correctly map inputs to outputs. The process is similar to the one described for the feedforward neural networks, with the exception that now, due to the

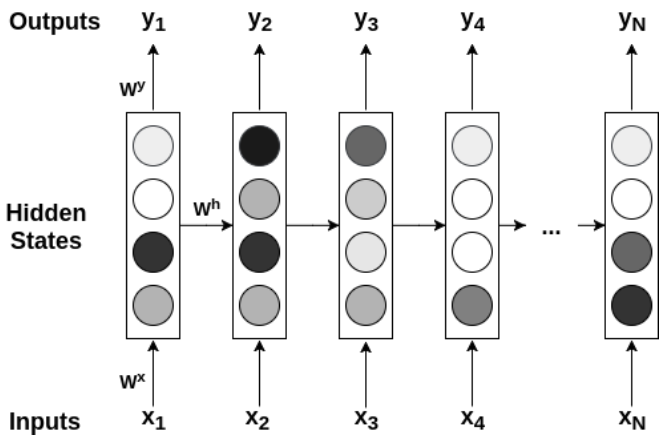


Figure 2.3: Simplified representation of a recurrent neural network.

recurrent nature of the network, it is necessary to backpropagate the error throughout the different time steps. This is done with *backpropagation through time* (Werbos [40]).

During the back-propagation of the error throughout the network, the values of the gradient become smaller at each time step. Therefore, for long sentences, it will be difficult for the later words to effectively capture the meaning of the earlier ones. This phenomenon causes RNNs to be hard to train and goes by the name of *vanishing gradient problem* (Hochreiter et al. [41]).

Long Short-Term Memory

The Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber [12]) has been widely used as a basic unit of RNNs in order to overcome the issue of the vanishing gradient. It does so by introducing a *memory cell* $\boldsymbol{c}_t$, capable of maintaining constant the gradient throughout the time steps of backpropagation. Furthermore, the LSTM architecture introduces three gates: the input gate $\boldsymbol{i}_t$; the forget gate $\boldsymbol{f}_t$; and the output gate $\boldsymbol{o}_t$. Each of these gates is calculated based upon the current input and the previous hidden state.

The cell starts by making an update $\boldsymbol{g}_t$, similarly to the vanilla RNN. This update to the state is weighted by the input gate and, summing to it the previous value of the memory cell weighted by the forget gate, yields the current memory cell. This cell is then passed through an hyperbolic tangent non-linearity and finally, it is weighed by the output gate, leading to the current hidden state $\boldsymbol{h}_t$. This process is formally defined by the following set of equations

$$\boldsymbol{g}_t = \tanh(\boldsymbol{W}^{xg}\boldsymbol{x}_t + \boldsymbol{W}^{hg}\boldsymbol{h}_{t-1} + \boldsymbol{b}_g), \tag{2.8}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^{xi}\boldsymbol{x}_t + \boldsymbol{W}^{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i), \tag{2.9}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^{xo}\boldsymbol{x}_t + \boldsymbol{W}^{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o), \tag{2.10}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^{xf}\boldsymbol{x}_t + \boldsymbol{W}^{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f), \tag{2.11}$$

$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \boldsymbol{g}_t + \boldsymbol{c}_{t-1} \odot \boldsymbol{f}_t, \tag{2.12}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{2.13}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$, $\boldsymbol{g}_t, \boldsymbol{i}_t, \boldsymbol{o}_t, \boldsymbol{f}_t, \boldsymbol{c}_t, \boldsymbol{h}_t \in \mathbb{R}^{d_h}$, $\boldsymbol{W}^{xi}, \boldsymbol{W}^{xf}, \boldsymbol{W}^{xo}, \boldsymbol{W}^{xg} \in \mathbb{R}^{d_x \times d_h}$, $\boldsymbol{W}^{hi}, \boldsymbol{W}^{hf}, \boldsymbol{W}^{ho}, \boldsymbol{W}^{hg} \in \mathbb{R}^{d_h \times d_h}$ and $\odot$ represents the *Hadamard product*, i.e., the element-wise product. When using an LSTM, it is still possible to make predictions at each time step, as mentioned for RNNs, using Equation 2.7.

2.2 Language Models

This section presents a brief overview on language models. This class of models makes it possible to assign a probability to a certain sequence of words or, more concretely, the probability of a given word considering that a certain sequence of words appeared before (Goldberg [42]). First, *n*-gram language models are presented. Then, the concept of word representation is introduced, due to its importance for the next two classes of language models: feedforward language models; and recurrent language models. Then, the concept of *perplexity* is briefly discussed, as it is the most common way of measuring the performance of language models.

2.2.1 N-Gram Language Models

Assigning a probability to a given sequence of words may be described formally as

$$P(w_1, w_2, w_3, ..., w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)P(w_4|w_1, w_2, w_3)...P(w_n|w_1, w_2, ..., w_{n-1}), \tag{2.14}$$

by using the chain rule of probability. From Equation 2.14 it is possible to observe that, the probability of a given sentence is the product of probabilities of words, conditioned on all words that precede it. This approach to the problem has two major shortcomings. The first is that finding whole sequences (sometimes long) of words in the training corpus might not be reasonable. The second is that, in the case of the last word, obtaining this probability might be a problem as hard as modeling an entire sequence (Goldberg [42]).

In order to overcome the latter weakness, it is possible to use the *n*-th order Markovian assumption, which states that the probability of a given word depends only on the last *n* words, hence

$$P(w_{i+1}|w_1, ..., w_i) \approx P(w_{i+1}|w_{i-n}, ..., w_i), \tag{2.15}$$

making it possible to rewrite 2.14 as

$$P(w_1, w_2, ...w_n) \approx \prod_{i=1}^{n} P(w_i|w_{i-n}, ..., w_{i-1}). \tag{2.16}$$

Keeping the Markovian assumption in mind, it is now possible to follow the maximum likelihood estimation (MLE) approach to obtain values for the probabilities of words given a certain context. These calculations make use of the counts of the respective *n*-grams in a training corpus, resulting in

$$P_{MLE}(w_{i+1}|w_{i-n},...,w_i) = \frac{c(w_{i-n},...,w_i,w_{i+1})}{c(w_{i-n},...,w_i)} \tag{2.17}$$

where $c(.)$ is the count of occurrences of a given word, or sequence of words, in the training corpus. While more effective than the previous approach, the solution denoted in Equation 2.17 still does not fully address the aforementioned problem of not finding certain *n*-grams in the training corpus. When that happens, the probability for that given *n*-gram is zero, and thus, the probability of the whole sequence is zero (see Equation 2.16). This means that the model is limited to what it has seen on the training set and it is not capable to generalize to unseen expressions, even if they are very similar. There are plenty of techniques to overcome this issue, being *modified Kneser-Ney smoothing* (Kneser and Ney [43]; Chen and Goodman [44]) the most widely used currently.

Even though *n*-gram language models have obvious benefits, such as being easy to train on large corpora, there are drawbacks, such as expensive memory requirements when scaled up, smoothing techniques that back-off to lower order *n*-grams, making the model unable to learn long term context, and also the lack of generalization capability (Goldberg [42]).

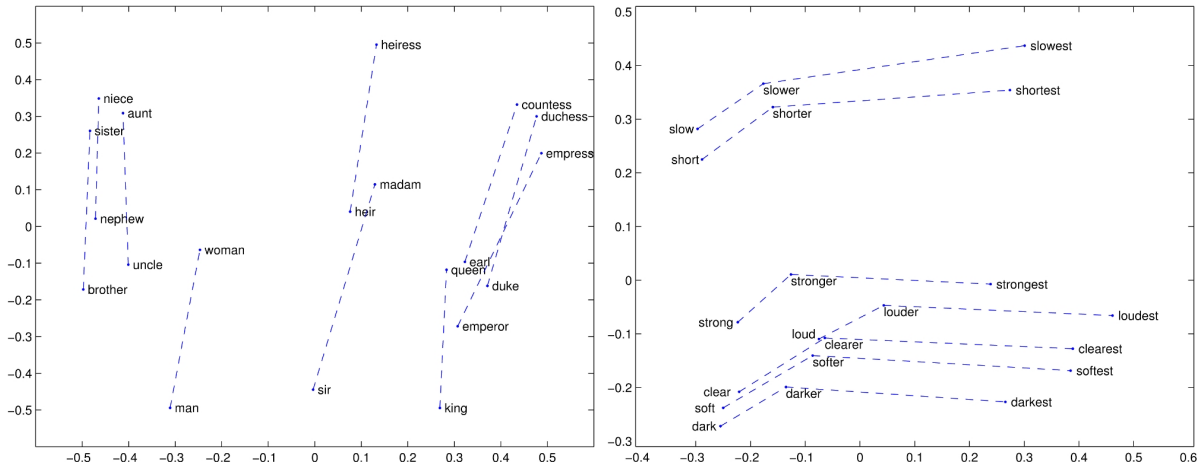
2.2.2 Word Representations

In order to make words something intelligible for neural networks, it is necessary to represent them as vectors. There are two common approaches to do so, *one hot encoding* and *dense feature embedding*.

One hot encoding produces one hot vectors, with size equal to the number of different words in the vocabulary. In these vectors, each entry corresponds to one feature/word, and it will have value one if it corresponds to the word it is encoding, and zero otherwise. One of the major weaknesses of this approach is that, due to the way these vectors are defined, it is not possible to find similarities between the represented words.

The dense feature embedding approach produces word embeddings, in which each feature is projected into a *d*-dimensional space, with *d* being much smaller than the number of words in the vocabulary. Thus, it will exist a small dimension space in which every word/feature is represented as a dense vector. There are several different strategies to obtain dense embeddings. One of the most widely known approaches is the *Word2vec* (Mikolov et al. [45]), where two possible methods are used: *continuous bag of words* or *continuous skip-gram*. The main idea behind the bag of words approach is to predict a word given its context, whereas for the skip-gram approach the goal is the opposite, i.e., predicting the context of a given word. Another widely used approach is *GloVe* (Pennington et al. [46]). This approach tries to get the best from neural and count-based approaches. In particular, it trains a weighted least squares regression model that is able to leverage word-word co-occurrence counts over the training corpus. In particular, this work was able to show linear relations between vectors, as seen in the examples the authors provide (Figure 2.4). Namely, it is possible to see that the differences between vectors are able to correctly portray the semantic difference between words.

More recently two approaches have gained traction, namely *fastText* (Bojanowski et al. [47]), and *ELMo* (Peters et al. [48]). *fastText* is an approach based on the previously mentioned skip-gram, with



(a) man-woman (b) comparative-superlative

Figure 2.4: Linear relation between word embeddings. Retrieved from GloVe's webpage.[1]

the peculiarity that instead of learning word-level vectors, it learns character level *n*-gram vectors, being a word vector the sum of these. By using character level *n*-grams, *fastText* makes it possible to, not only incorporate word morphology in the obtained representations, but also to compute word representations for unseen words at training time, something highly beneficial and that was not possible in previous approaches. Finally, ELMo (Peters et al. [48]) has achieved state-of-the-art results in several NLP tasks. In this approach, the yielded word embeddings are the hidden states of a language model based on a specific class of RNNs to be presented later in this work, which is able to encompass not only left to right context, but also right to left context. ELMo's authors highlight three characteristics: the representation of each word is based upon the whole context; it combines all layers of a deep neural network model; and its representations are character-based, meaning that it is able to leverage morphological cues from different words (Peters et al. [48]). This last characteristic means that, similarly to *fastText*, ELMo's embeddings are also able to deal with rare words.

The advantage of using dense feature embeddings is twofold: first, it provides a better representation in computational terms, when compared with the sparsity, and the size, of the one hot encoding representations; second, it makes it possible for semantically similar words to be projected close to each other in the space where they are embedded. The latter is particularly useful because, even if there are no specific training examples that relate two concepts, models should be able to leverage the semantic similarity captured by the dense embeddings. For example, words like *"train"* and *"bus"* will be frequently paired with verbs related to motion and traveling. Imagine now that it is necessary to relate the word *"airplane"* with that class of verbs, but there were no such instances in the training data. The fact that the word *"airplane"* should be close to *"train"* and *"bus"* in terms of word embeddings, will make models able to use it with the class of mentioned verbs, even if there were no examples of that behaviour in the training corpus. As previously mentioned, this kind of generalization would not be possible using one hot encoding due to the independence amongst features.

Also interesting to mention is that, as it may have become clear during this subsection, word rep-

[1] https://nlp.stanford.edu/projects/glove/

resentations may sometimes be a side product of training language models, such as the case of the mentioned ELMo embeddings Peters et al. [48].

2.2.3 Feedforward Neural Language Model

As an alternative to the standard *n*-gram language models, Bengio et al. [49] introduced the concept of feedforward neural language models. This class of language models follows closely the process described in Section 2.1, namely, regarding its training procedure.

With a feedforward neural language model, in order to obtain the probability of a word given its context, i.e., $P(w_{i+1}|w_{i-n}, ..., w_i)$, the first step is to create the input vector. This input vector is obtained by concatenating the embeddings of the context words. In particular, it could be seen as

$$\boldsymbol{x} = [\boldsymbol{E}_{[w_{i-n}]}; ...; \boldsymbol{E}_{[w_i]}], \tag{2.18}$$

where $\boldsymbol{E}_{[w_i]} \in \mathbb{R}^{D_e}$ corresponds to the embedding vector of word i. Then, as in Section 2.1, this vector is passed through a series of affine transformations plus non-linear functions, with the final one being a softmax. This creates a probability distribution over a vocabulary of size $|V|$. Having the probabilities for each word in the vocabulary, the common behaviour is to select the one with the highest probability. This would be our w_{i+1}, given the context window of size n.

Even though feedforward networks are a strong approach to language modelling, in particular since they can leverage dense word embeddings, and high volumes of data while being trained with the backpropagation algorithm, there are some shortcomings to consider. The first shortcoming is that, increasing the number of context words increases the number of trainable parameters. The second is that, it does not seem likely that a feedforward approach is able to completely take advantage of the context, due to the recurrent nature of a sentence.

2.2.4 Recurrent Neural Language Model

Even though feedforward neural language models are more capable than *n*-gram language models, they are still unable to capture recurrent dependencies in the provided context, and scale poorly with the increase in the number of context words. It is possible to overcome this two weaknesses by introducing language models based on recurrent neural networks (Subsection 2.1.2), more concretely, *recurrent neural language models* (Mikolov et al. [50]). In particular, RNNs make it possible to use the whole context to produce the next word, and not only the words in a given window of size n. Also, the number of parameters of the model does not depend on the size of the context, since only a single word embedding is provided at each time step, and not the concatenation of several, as described for feedforward language models. The training process would be similar to the one described in Subsection 2.1.2, namely, making use of the *backpropagation through time* algorithm and using cross-entropy as the loss function.

In practice, if we were to obtain the word w_i given its context, the process would be very similar to the standard behavior for a RNN. The first hidden state would be randomly initialized and then, it would be

updated following $\boldsymbol{h}_i = g\big(\boldsymbol{W}^e \boldsymbol{E}_{[w_i]} + \boldsymbol{W}^h \boldsymbol{h}_{i-1} + \boldsymbol{b}^h\big)$. At step time i, applying a softmax transformation to the hidden state $\boldsymbol{h}_i$, yields a probability distribution over the vocabulary $|V|$, from which it is possible to choose the most probable word. During training this probability distribution is created at each time step, from where it is possible to obtain the log probability of the true prediction.

2.2.5 Perplexity

Having defined three approaches to building language models, it is necessary to define a way of measuring their perfomance. A common way to do so, is to use *perplexity*. It is defined as

$$\mathrm{ppl}(\mathrm{LM}, N) = 2^{-\frac{1}{N}\sum_{i=1}^{N} log_2 P_{LM}(w_i|w_1,...,w_{i-1})} \tag{2.19}$$

where N is the number of words in the given training corpus, LM is the trained language model, and P_{LM} is the probability yielded by the aforementioned language model for the word w_i, given its context.

The idea behind this concept is that it answers the question "if random words are chosen from the probability distribution calculated from the language model at each time step, how many words, on average, would have to be picked in order to get the correct one?" (Neubig [35]). The lower the value of perplexity, the better the language model.

2.3 Neural Machine Translation

In this section the main ideas of neural machine translation (NMT) will be presented. Neural machine translation may be seen as an extension of the recurrent neural language model. The goal of NMT is to translate a sequence of words from a source language, $\boldsymbol{X} = x_1, x_2, ..., x_J$ to a sequence of words in a target language, $\boldsymbol{Y} = y_1, y_2, ..., y_T$, by learning to model a conditional probability, $P(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{t=1}^{T} P(y_t|y_1, ..., y_{t-1}, \boldsymbol{X})$. The first step will be to define the encoder-decoder approach to model this probability. Then, the attention mechanism will be discussed, as it is important for the scope of this work. Finally, the section provides a brief overview on how the best translations are obtained, and also on how the choice of vocabulary size affects the model performance.

2.3.1 Encoder-Decoder Approach

The most common approach to tackle the problem of neural machine translation is the encoder-decoder architecture (Kalchbrenner and Blunsom [7], Cho et al. [8], Sutskever et al. [9]). Even though it is possible to follow different strategies (Gehring et al. [13], Vaswani et al. [16]), the most usual approach is to use RNNs with LSTMs as unit cells.

The main idea of this approach may be defined in two steps: first, an encoder is used to output a hidden state, i.e., a vector that is supposed to encode the source language sentence; then, the decoder, which may be seen as a recurrent neural language model, will use that intermediate representation as

its initial hidden state and, conditioned on the source sentence, generate target language words. A simplified representation of the process may be seen in Figure 2.5.

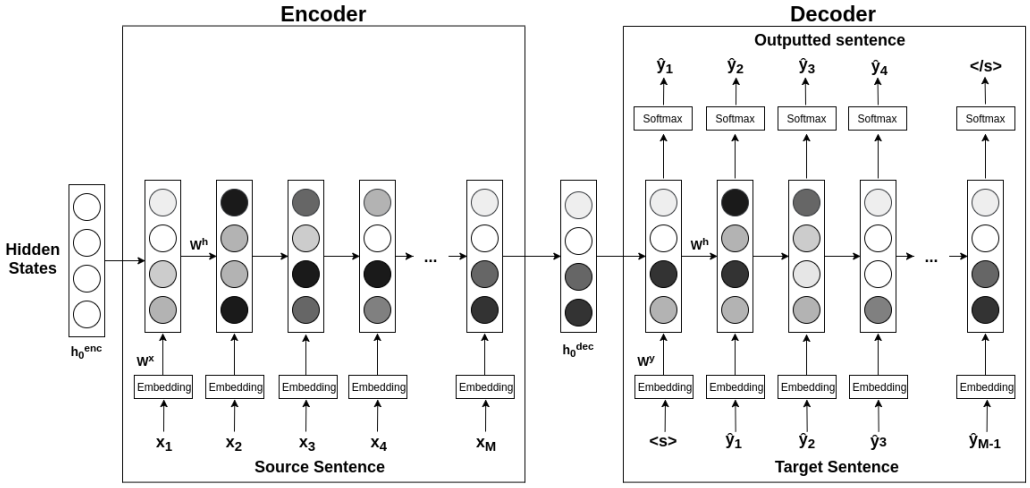


Figure 2.5: Simplified representation of the NMT encoder-decoder architecture.

The previously described encoder-decoder approach, as it is, has three major shortcomings. The first one is that hidden states will have to hold information about the whole source language sentence, both at the end of the encoding step and during decoding. Also, the hidden states are expected to have information on whether a word is yet to be translated, or if it has already been translated. Moreover, since hidden states have a pre-defined size, and sentences come in various lengths, there might be occasions in which the size of the hidden vectors is too small to encode all the necessary particularities of sentences, and others in which it might be too large, ending up overfitting the data. Finally, if the sentences are too long, and the words to be translated into each other are at opposing extremities, then this long dependency might not be captured (Neubig [35]).

To overcome the aforementioned issues previous studies have introduced a soft alignment model, called attention (Bahdanau et al. [14], Luong et al. [15]). This alignment model allows the decoder to partially attend the source words at each decoding step. In this case, the source sentence is represented by a matrix where every column j is the hidden state of the encoder for source word x_j. Using the attention mechanism alleviates the burden of the hidden states at the decoder side, while also providing a better context when producing the target language sentence.

Encoder

The goal of the encoder portion of the model is to capture the meaning of the source sentence and represent it as a vector. This is achieved by running a recurrent neural network, in which hidden vectors will encode all the words to its left. Since the encoder has access to the full source sentence, it seems reasonable to also leverage the context to the right of the source sentence being processed. This is achievable by using a second encoder that deals with the source language sentence from right to left. By concatenating the hidden states of both encoders at each time step, it is possible to produce encoder states which have information about the whole context of each source word. This approach is known

as the bi-directional encoder (Bahdanau et al. [14], Schuster and Paliwal [51]), usually referred to as bi-LSTM. More formally, the aforementioned description may be defined as

$$\overrightarrow{\boldsymbol{h}_j^e} = \mathrm{RNN}(\overrightarrow{\boldsymbol{h}_{j-1}^e}, \boldsymbol{E}_{[x_j]}) \tag{2.20}$$

$$\overleftarrow{\boldsymbol{h}_j^e} = \mathrm{RNN}(\overleftarrow{\boldsymbol{h}_{j+1}^e}, \boldsymbol{E}_{[x_j]}), \tag{2.21}$$

being $\boldsymbol{E}_{[x_j]}$ the embedded vector of word x_j, and $\boldsymbol{h}_j^e$ the hidden state of the encoder at position j, resulting then in the bi-directional representation

$$\boldsymbol{h}_j^e = \left[\overleftarrow{\boldsymbol{h}_j^e}; \overrightarrow{\boldsymbol{h}_j^e}\right]. \tag{2.22}$$

The final hidden state of the encoder is used to initialize the decoder.

Decoder

The goal of the decoder is to produce target language words, until a end of sentence token is output. The common choice for the decoder is to use a recurrent neural network. Each new hidden state $\boldsymbol{h}_t^d$, depends on the previous hidden state $\boldsymbol{h}_{t-1}^d$, the embedding of the previous output word $\boldsymbol{E}_{[y_{t-1}]}$, and also on a context vector $\boldsymbol{c}_t$. Each hidden state of the decoder is therefore given by

$$\boldsymbol{h}_t^d = \mathrm{RNN}(\boldsymbol{h}_{t-1}^d, \boldsymbol{c}_t, \boldsymbol{E}_{[y_{t-1}]}). \tag{2.23}$$

At each decoding time step t, it is then possible to obtain a probability distribution over the vocabulary by making use of the softmax. This probability distribution allows us to choose the output word at each time step during the translation procedure, as it will be further explored in Subsection 2.3.4. It is worth noting that, at training time, it might be used as input to the Recurrent Neural Network (RNN) the embedding of the previous reference output (the translation *gold-standard*) y_{t-1}, whereas at inference time the actual previous output $\hat{y}_{t-1}$, is always used.

2.3.2 Attention Mechanism

As previously stated, the attention mechanism provides a way of enriching the source side information the decoder has available at each time step. This is achieved by the context vector $\boldsymbol{c}_t$, that is used as input to the recurrent neural network. The context vector is obtained as a weighted sum of the hidden states produced during the encoding step, formally defined as

$$\boldsymbol{c}_t = \sum_{j=1}^{J} \alpha_{tj} \boldsymbol{h}_j^e, \tag{2.24}$$

being J the number of words of the source sentence, and $\boldsymbol{\alpha}$ the normalized vector of attention scores. This vector is in fact a probability distribution over the source words, and thus, each of its values may be interpreted as the importance of the corresponding source word for the output at decoding step t. The

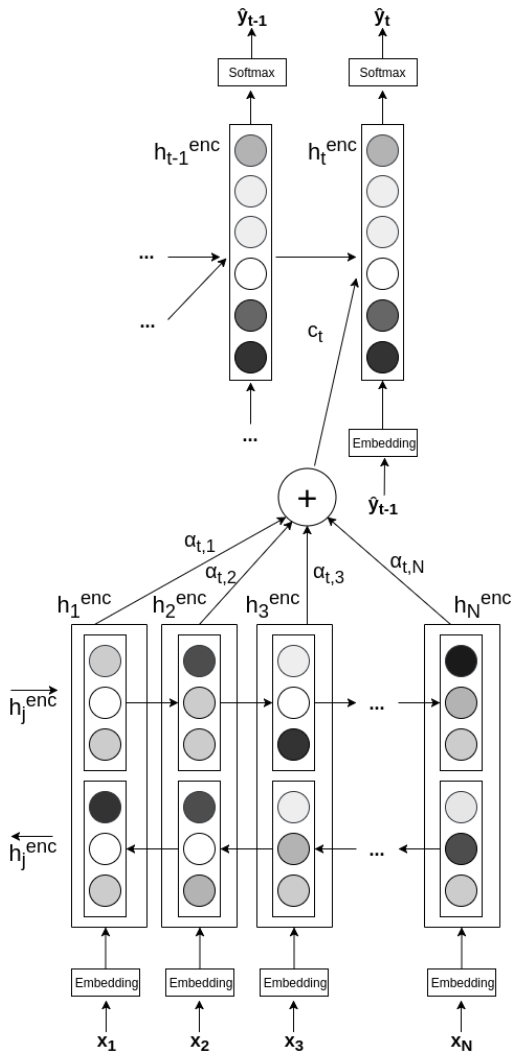


Figure 2.6: Simplified representation of the NMT encoder-decoder architecture with Bahdanau et al. [14]'s attention.

probability α_{tj}, relative to each encoder hidden state $\boldsymbol{h}_j^e$, is given by

$$\alpha_{tj} = \frac{\exp(z_{tj})}{\sum_{k=1}^{J} \exp(z_{tk})}, \tag{2.25}$$

where z_{tj} is an alignment model that is able to capture how important the source words close to position j are for the current decoding step t. There are multiple ways of defining this alignment model.

The most widely known is the approach by Bahdanau et al. [14]. In particular, it proposes training a feedforward neural network simultaneously with the rest of the NMT model. The alignment model in this case is defined as $z_{tj} = a(\boldsymbol{h}_{t-1}^d, \boldsymbol{h}_j^e)$. The way this mechanism is incorporated in the NMT model may be seen in Figure 2.6.

Posterior work by Luong et al. [15] presented a slightly different approach to the attention mechanism, as well as novel alignment models. Before introducing those alignment models, it is necessary to go through some of the subtleties of Luong et al. [15]'s attention. Namely, in Equation 2.23, the context vector is no longer provided, i.e., it is re-written as $\boldsymbol{h}_t^d = \mathrm{RNN}(\boldsymbol{h}_{t-1}^d, \boldsymbol{E}_{[y_{t-1}]})$. Instead, the context is

used to obtain an intermediate representation, $\boldsymbol{h'} = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t^d])$. Notice that Equations 2.24 and 2.25 are not affected by this change and still hold true. Finally, the probability distribution at a given decoding step is given by $p(y_t|y_1, ..., y_{t-1}, \boldsymbol{X}) = \mathrm{softmax}(\boldsymbol{W}_s\boldsymbol{h'})$. $\boldsymbol{W}_c$ and $\boldsymbol{W}_s$ are both matrices of weights learned during training. In the case of Luong et al. [15], three different alignment models z_{tj}, are introduced:

- *Dot product*: $\boldsymbol{h}_t^{d^T}\boldsymbol{h}_j^e$.
- *General*: $\boldsymbol{h}_t^{d^T}\boldsymbol{W}_a\boldsymbol{h}_j^e$.
- *Concat*: $\boldsymbol{v}_a^T \tanh\Big(\boldsymbol{W}_a\big[\boldsymbol{h}_t^d; \boldsymbol{h}_j^e\big]\Big)$.

2.3.3 Training

Training a neural machine translation model is similar to what is done for a recurrent neural language model, with two differences. The first, is that the training now has the goal of maximizing the conditional log-likelihood,

$$C(\theta) = \frac{1}{D}\sum_{d=1}^{D} \log p_\theta(\boldsymbol{Y}^d|\boldsymbol{X}^d) = \frac{1}{D}\sum_{d=1}^{D}\sum_{t=1}^{M_d} \log p_\theta(y_t^d|y_1^d, ..., y_{t-1}^d, \boldsymbol{X}^d), \tag{2.26}$$

via a set of parameters $\boldsymbol{\theta}$, being D the number of sentence pairs in the data set and M_d the length of a target sentence. This will penalize sets of parameters that are not able to output probability distributions at each decoding time step in which the correct words are probable to happen. The second, is that the decoder RNN is initialized with the last hidden state of the encoder, whereas the encoder RNN is initialized by following some particular initialization technique.

2.3.4 Translation Generation

After modelling the conditional probability $p(\boldsymbol{Y}|\boldsymbol{X})$, it is necessary to define how to choose the output words at each decoding time step. As discussed previously, for each decoding step it is obtained an output vector with a probability distribution over all possible words. Finding the best translation corresponds to finding the sequence of output words that maximizes $p(\boldsymbol{Y}|\boldsymbol{X})$, i.e.,

$$\hat{\boldsymbol{Y}} = \arg\max_{Y} p(\boldsymbol{Y}|\boldsymbol{X}). \tag{2.27}$$

Solving this problem would require computing the log probability of every possible translation. Since that is not feasible, an approximate search has to be used as an alternative. An intuitive procedure would be to select the most probable word at each time step, performing what is called a *greedy 1-best search*. A shortcoming of this strategy is the possibility of falling into the *garden-path problem* (Koehn [52]). This problem concerns sentences known for their ambiguity, since their first words are not enough to understand which is the structure of the full sentence. One example of these type of sentences is *"The complex houses married and single soldiers and their families."* (Petrie et al. [53]). In this case, as

the sentence is created, the most common assumption would be that *houses* refers to a noun, when actually it is a verb. In terms of the greedy search, this means that less probable words may be the best choice early on the translation, since they may lead to more probable ones later. With greedy search, once there is a commitment to a given ”path” there is no turning back, which may render impossible to deal with some particular sentences. For instance, in the aforementioned example it is very unlikely that the most probable word after *”house”* would be *”married”*, compromising the corresponding translation.

A more reasonable approach is to keep a *beam* of the *n* most probable sequences of words at each time step, the so-called *beam search* approach. This means that, at each decoding time step there are temporarily $n \times |V|$ hypotheses (being $|V|$ the number of words in the vocabulary), which are then reduced to the top *n*. When the end sentence token $< /s >$, is predicted by the decoder, a given hypothesis is assumed to be terminated. Whenever this happens, that hypothesis is stored, the width *n* of the beam is reduced by one, and the procedure continues, until that value is reduced to zero. The most probable hypothesis, among the *n* candidates, is chosen as the translation of the source sentence.

The probability of a given sequence is the product of the individual probability of each of the words that composes it. This means that, every time a word is added to a sequence, since a probability is a positive number greater than or equal to 1, the overall probability of the sequence will decrease or remain the same. Consequently, shorter sentences will be preferred, producing the so called *length bias* toward shorter sentences. There are some strategies to deal with this situation, such as normalizing the probability of the candidate translations by its length (Cho et al. [54]), or by introducing a scoring function that depends on a length normalization term (Wu et al. [25]), which will be further explored in Subsection 2.5.4.

2.3.5 Vocabulary Size

One of the main aspects of a neural machine translation model is its vocabulary. Regardless of using a joint vocabulary, or a specific vocabulary for each of the languages, its size has a clear impact in the performance of the model. The two portions of the model affected by the length of the vocabulary are: the embedding matrix that maps the source/target words of the vocabulary into the respective embedding space; and the output layer, which calculates a probability over the vocabulary. Therefore, it is necessary to find a balance between a large vocabulary size, that leads to the least amount possible of out-of-vocabulary words (usually replaced by a *unknown* symbol, denoted as $< \mathrm{unk} >$), and a small vocabulary, which reduces the complexity of the model.

A common way of overcoming this problem is to use the Byte Pair Encoding (BPE) algorithm to create *subwords* (Sennrich et al. [17]). This approach is based on the idea that different words share common smaller units, the so called subwords, whose translation may be concatenated and lead to the correct translation of full words. The categories of words that Sennrich et al. [17] mentions as plausible to be split into subwords are: named entities; cognates and loanwords; and morphologically complex words. The created subwords are easily recognizable by the appended termination, ”@@”. The benefits from using this approach, as opposed to a fixed-vocabulary, are twofold: first, the models improve their ability

of translating rare and out-of-vocabulary words; second, Sennrich et al. [17] reports improvements in BLEU while reducing the vocabulary size, and thus, the complexity of the whole process of NMT.

Another work concerned with vocabulary issues, more concretely the impact of unknown words, is Luong et al. [55]. Namely, it proposes keeping track of the attended source words when unknown symbols are produced at decoding time. This information is later used to replace those unknown words by a translation found in an external dictionary, or by the source word itself. This strategy may be paired with BPE to completely remove the scarce unknowns that may be nonetheless produced.

2.4 Activation Functions

In the attention mechanism, the most common activation function used to obtain the attention vector $\boldsymbol{\alpha}$, is the standard softmax, as seen in Equation 2.25. In this section, two alternatives to obtain a probability distribution are presented, namely, sparsemax (Martins and Astudillo [56]), and constrained softmax (Martins and Kreutzer [57]).

2.4.1 Sparsemax

Sparsemax is an activation function that distinguishes itself from the commonly used softmax due to its capability of producing sparse probability distributions. This means that, contrary to softmax, where the yielded probabilities are strictly positive, sparsemax is able to produce probability distributions where some of the individual values are zero. The sparse probability distribution $\boldsymbol{p}$, is obtained by projecting the vector of logits $\boldsymbol{z} \in \mathbb{R}^K$, onto the probability simplex $\Delta^{K-1} := \{\boldsymbol{p} \in \mathbb{R}^K \mid \sum_k p_k = 1, p_k \geq 0 \; \forall_k\}$, as

$$\mathrm{sparsemax}(\boldsymbol{z}) := \underset{\boldsymbol{p} \in \Delta^{K-1}}{\arg\min} \|\boldsymbol{p} - \boldsymbol{z}\|^2, \tag{2.28}$$

and it has a closed form solution given by

$$\mathrm{sparsemax}_i(\boldsymbol{z}) = [z_i - \tau(\boldsymbol{z})]_+, \tag{2.29}$$

where $\tau : \mathbb{R}^K \to \mathbb{R}$ is the only function that satisfies $\sum_i [z_i - \tau(\boldsymbol{z})]_+ = 1$ for every z_i, i.e., all the individual probabilities sum to one. Notice as well the use of the function $[t]_+ = \max\{0, t\}$, ensuring that each individual value is non-negative. Both the aforementioned conditions are necessary to obtain a valid probability distribution. The necessary steps to obtain the sparsemax of a vector are represented in Algorithm 2 (Martins and Astudillo [56]).

Algorithm 2 Sparsemax Evaluation

1: **Input:** $\boldsymbol{z}$
2: Sort $\boldsymbol{z}$ as $z_1 \geq z_2 \geq ... \geq z_K$
3: Find $k(\boldsymbol{z}) := \max\{k \in [K] \mid 1 + kz_k > \sum_{j \leq k} z_j\}$
4: Define $\tau(\boldsymbol{z}) = \frac{\left(\sum_{j \leq k} z_j\right) - 1}{k(\boldsymbol{z})}$
5: **Output:** $\boldsymbol{p}$ $s.t.$ $p_i = [z_i - \tau(\boldsymbol{z})]_+$

2.4.2 Constrained Softmax

Constrained softmax is an activation function that differs from softmax by making it possible to define a hard upper bound on the probability that is assigned to each element. The constrained softmax of a vector is obtained by solving the following optimization problem

$$\begin{aligned} &\mathrm{csoftmax}(\boldsymbol{z};\boldsymbol{u}) := \underset{\boldsymbol{p}\in\Delta^{K-1}}{\arg\min} -\mathbf{H}(\boldsymbol{p}) - \boldsymbol{z}^T\boldsymbol{p} \\ &\text{subject to} \quad \boldsymbol{p} \leq \boldsymbol{u}, \end{aligned} \tag{2.30}$$

in which $\boldsymbol{p} \in \mathbb{R}^K$ is the probability distribution, $\boldsymbol{u} \in \mathbb{R}^K$ is a vector of upper bounds to the individual probabilities, and $\mathbf{H}$ is the entropy. The formulation of Equation 2.30 comes naturally from rewriting the softmax expression in a form that takes into account the Kullback-Leibler (KL) divergence, a measure of how different a given probability distribution is from a reference probability distribution (Kullback and Leibler [58]),

$$\mathrm{softmax}(\boldsymbol{z}) = \underset{\boldsymbol{p}\in\Delta^{K-1}}{\arg\min}\, \mathrm{KL}(\boldsymbol{p};\mathrm{softmax}(\boldsymbol{z})) = \underset{\boldsymbol{p}\in\Delta^{K-1}}{\arg\min} -\mathbf{H}(\boldsymbol{p}) - \boldsymbol{z}^T\boldsymbol{p}. \tag{2.31}$$

In words, the optimization problem that leads to the constrained softmax allows us to find the closest probability distribution to the one yielded by softmax, with the constraint of having the probability values capped by a vector of upper bounds.

2.5 State-of-the-Art of Adequacy in NMT

In this section, the most relevant state-of-the-art approaches to the problem of adequacy are going to be presented. Namely, we start by going through three different works by the same research group, each one proposing a different approach to the problem of adequacy in neural machine translation. These three approaches are: introduction of a coverage vector (Tu et al. [19]); addition of a new gating mechanism related with context (Tu et al. [23]); and training the model to be able to reconstruct the original input sentence from the obtained translation (Tu et al. [24]). Then, the section presents a strategy where beam hypotheses are re-scored according to a new scoring function that includes a coverage penalty, and a length normalization factor(Wu et al. [25]). Finally, it is presented a brief overview of the strategies existing in the literature with regard to the translation of rare words, and the robustness to domain mismatches.

2.5.1 Coverage

In phrase-based SMT, the decoder keeps a vector, namely a *coverage vector*, that indicates if a given source word has already been translated. This coverage vector is used to ensure that each word of the source language sentence has already been translated, or ”covered”, during the procedure. Such an equivalent artifact does not exist in the NMT models discussed until here. Intuitively, it is possible to understand that such a vector would make it possible to address two of the adequacy issues mentioned

in Subsection 1.1: the problem of translating some words multiple times; and the problem of leaving source words untranslated during the process. In order to overcome this issue, Tu et al. [19] proposed the utilization of a coverage vector that summarizes the attention history for a given source hidden state $\mathbf{h}_j^e$. The intuition is that this coverage vector should be able to capture if a given source word x_j, has already been covered, thus preventing the model from spending more attention on that given position, and forcing it to be spread among the remaining words. This coverage vector is introduced on the NMT model by rewriting the alignment model of attention as

$$z_{t,j} = a(\boldsymbol{h}_{t-1}^d, \boldsymbol{h}_j^e, \mathcal{C}_{t-1,j}), \tag{2.32}$$

where $\boldsymbol{h}_{t-1}^d$ is the previous decoder hidden state, $\boldsymbol{h}_j^e$ is the encoder hidden state produced by the input word x_j, and $\mathcal{C}_{t-1,j}$ is the coverage of source word j before the decoding time step t,

$$\mathcal{C}_{t,j} = f(\mathcal{C}_{t-1,j}, \alpha_{t,j}, \boldsymbol{h}_j^e, \boldsymbol{h}_{t-1}^d), \tag{2.33}$$

where the function f is defined as a RNN using the GRU gating function (Cho et al. [8]).

A very similar approach was also proposed in Mi et al. [22]. There are two main differences with regard to the work of Tu et al. [19]: each source word is represented by a coverage embedding which may be updated using a GRU, or a weighted subtraction of the embedding of the output word; and it is introduced a regularization term in the training objective. In particular, this regularization term penalizes the absolute value of the coverage vector, i.e., penalizes having unattended source words during the translation.

2.5.2 Context Gates

In Tu et al. [23] it is proposed the use of *context gates*, non-linear gating units which are able to define the ratio of information that should flow from both the source and target hidden states, during decoding. This follows the authors' findings that source context influences *adequacy*, whereas target context influences *fluency*. The context gate r_t, is defined as

$$r_t = \sigma(\boldsymbol{W}^y \boldsymbol{E}_{[y_{t-1}]} + \boldsymbol{W}^{h_d} \boldsymbol{h}_{t-1}^d + \boldsymbol{W}^{h_e} \boldsymbol{h}_t^e). \tag{2.34}$$

The architecture that yielded the best results when expanding NMT models with this context gate, corresponds to a linear interpolation between the source and target hidden states, defined by

$$\boldsymbol{h}_t^d = f\Big((1 - r_t) \circ (\boldsymbol{W}^y \boldsymbol{E}_{[y_{t-1}]} + \boldsymbol{W}^{h_d} \boldsymbol{h}_{t-1}^d) + r_t \circ \boldsymbol{W}^{h_e} \boldsymbol{h}_t^e\Big), \tag{2.35}$$

where $\boldsymbol{h}^d$ corresponds to the hidden state of the decoder, $\boldsymbol{h}^e$ corresponds to the hidden state of the encoder, and $\boldsymbol{W}^y$, $\boldsymbol{W}^{h_d}$ and $\boldsymbol{W}^{h_e}$ are weight matrices.

2.5.3 Neural Machine Translation with Reconstruction

Tu et al. [24] proposed tackling the issue of adequacy by introducing a *reconstructor* module into the encoder-decoder approach. This reconstructor works by translating the obtained target language sentence back to the source language while calculating a reconstruction score. Training this model is possible by introducing a new training objective that includes both the *vanilla* likelihood and the reconstruction score, interpolated by a hyper-parameter, λ. Also, the reconstructor receives as input the output sentence, and has access to the corresponding hidden states, allowing it to make use of the attention mechanism.

The intuition behind this approach is that, by forcing the model to be able to reconstruct as closely as possible the original sentence, based on the output sentence, the model will be induced into learning to incorporate the necessary information to prevent poor adequacy.

2.5.4 Scoring Function with Coverage Penalty and Length Normalization

Wu et al. [25] introduced a global scoring function, that re-scores the candidate translations in beam search according to a *coverage penalty*, and a *length normalization* term. The goal of beam search decoding is then to find the target language sequence $\boldsymbol{Y}$, that maximizes this scoring function. The aforementioned scoring function, as well as the coverage penalty and the length normalization, are defined as

$$s(\boldsymbol{Y}, \boldsymbol{X}) = \frac{\log(P(\boldsymbol{Y}|\boldsymbol{X}))}{lp(\boldsymbol{Y})} + cp(\boldsymbol{X}; \boldsymbol{Y}) \tag{2.36}$$

$$lp(\boldsymbol{Y}) = \frac{(5 + |\boldsymbol{Y}|)^{\alpha}}{(5+1)^{\alpha}} \tag{2.37}$$

$$cp(\boldsymbol{X}; \boldsymbol{Y}) = \beta \times \sum_{j=1}^{|\boldsymbol{X}|} \log\Bigg(\min\Bigg(\sum_{t=1}^{|\boldsymbol{Y}|} p_{jt}, 1.0\Bigg)\Bigg), \tag{2.38}$$

where $\boldsymbol{Y}$ and $\boldsymbol{X}$ are sequences of words in the target and source languages, respectively, $lp(.)$ is the length normalization term, $cp(.)$ is the coverage penalty term, p_{jt} is the attention probability between the *t*-th target word and the *j*-the source word, and α and β are constants which control the impact of both terms. When both of them are defined as zero, the scoring function performs as the normal beam search.

Notice as well that both introduced terms impact the beam search as expected. The length normalization term divides a numerator which is the logarithm of a probability, yielding a value between $-\infty$ and 0. Furthermore, the value of lp is always higher or equal than one, as $|\boldsymbol{Y}| \geq 1$. Therefore, the higher lp is, the less negative $\frac{\log P(\boldsymbol{Y}|\boldsymbol{X})}{lp(\boldsymbol{Y})}$ is, and thus, the less penalized the corresponding hypothesis is. The goal of this term is to counteract the beam tendency to choose shorter sentences. Remember that each added word sums a negative log probability to the current value, leading to worse scores for longer sentences. Looking at the definition of lp, larger translations in a given beam lead to higher length normalization terms. Thus, it will act as intended and penalize smaller translation candidates.

In the case of the coverage penalty, hypotheses where the source words have received less attention overall will be more penalized than hypotheses where source words have been more attended. Taking the logarithm further emphasizes cases where some source words have low values of p_{jt}. Finally, Wu et al. [25] suggests defining both α and β as 0.2, as these are the values that yield the best scores.

2.5.5 Translating Rare Words

As discussed in Subsection 2.3.5, one of the benefits of using BPE is the improved robustness of the model with regard to rare words. Nonetheless, one of the reasons for poor adequacy in translations is still the subpar performance of regular NMT models when translating words that are not frequently seen in the training data (Arthur et al. [20], Koehn and Knowles [21]).

Using external knowledge to bias the translation is something that has been used in the past with some degree of success. Arthur et al. [20] proposed using traditional SMT techniques to learn lexical translation probabilities, that could then be either added or interpolated with the NMT output layer. A more rigid approach is followed by Hokamp and Liu [26] and Chatterjee et al. [27], in which the choice of the output word at each time step may be restricted by a set of external guidelines.

Another possible way of incorporating external knowledge is through retrieval-based approaches, in which models can search and leverage information from similar sentence pairs, part of an extra corpus. Such models are inspired by *translation memories* (Sadler and Vendelmans [59]) and *example based translation* models, first proposed in Nagao [60]. More concretely, Farajian et al. [28] proposed an unsupervised method in which sentence pairs from extra data, whose source is similar to the input sentence, are used to re-train a NMT model, specific for that sentence. Li et al. [29] proposes a similar approach but re-trains the NMT model for one epoch with the retrieved set of similar sentence pairs, whereas Farajian et al. [28] dynamically decides the learning rate and the number of epochs based in the similarity between the input and the retrieved sentences. In a different approach, Gu et al. [30] uses a translation model to obtain information about the hidden states and output layers of retrieved translation pairs. These are then used during the training of the regular NMT model, either by merging the actual decoder hidden state with the retrieved ones (*deep fusion*) or by biasing the output layer (*shallow fusion*). Finally, Feng et al. [31] uses one-to-one alignments between words, automatically created from the training data, to create a dictionary which may be attended by the vanilla NMT model during translation. Any of the mentioned approaches requires either training the model again using a restrict set of sentences, or to severely increase the computationally cost of the training procedure.

A particularly interesting approach is the recent work by Zhang et al. [32], which offers a computationally lighter approach to the problem. Namely, it proposes using *translation pieces*, i.e., *n*-grams of target language subwords, automatically extracted using similarity and alignment information between several retrieved sentences and an input sentence. This translation pieces are then used to bias the NMT output layer during decoding and thus, avoiding the need of re-training the model.

2.5.6 Domain Adaptation

Current NMT models are not able to generalize for out-of-domain sentences, clearly impacting the adequacy of the output translations (Koehn and Knowles [21]). Domain Adaptation is a possible way of overcoming the problem by having some way of leveraging extra in-domain data, in order to obtain better results for that particular domain. Being a powerful and thus, much used method in NMT, there is an extensive number of strategies proposed in the literature. A comprehensive review of the different approaches may be seen in Chu and Wang [61]. The most common approach to domain adaptation is *fine-tuning* (Luong and Manning [62]). In this approach, first a neural machine translation model is trained using a generic, large corpus. Then, starting from this strong baseline, the model is further trained using a set of in-domain data. Furthermore, Chu et al. [63] has shown that straightforward fine-tuning is able to yield scores comparable with other slightly more intricate approaches.

2.5.7 Other Approaches

Besides the aforementioned strategies to deal with adequacy and coverage issues in NMT, there are also some other relevant works. Namely, Feng et al. [64] suggests tackling the issue of adequacy by introducing a *condition vector*, able to keep track of the parts of the sentence already translated, akin to the previously mentioned coverage vectors. The authors argue that this vector is able to portray the behavior of *distortion* and *fertility* models, improving the alignments obtained with the attention mechanism. Distortion models act by controlling how words are re-ordered, while fertility is concerned with the expected number of target words obtained from translating a given source word.

In Zheng et al. [65], instead of keeping a coverage vector, information regarding the source sentence is separated into two recurrent layers, one concerning translated past content and other concerning untranslated future content. The combination of both holds full information on the source state, meaning that it is still possible to maintain the full context of the source side, while improving the knowledge about the parts of the sentence that have already been attended.

2.6 Automatic Metrics

In machine translation, one of the major areas of ongoing research is on how to obtain automatic metrics that correlate well with human judgment on the quality of a given translation. Several metrics have been proposed throughout time, approaching the problem in several different ways. Nonetheless, only some have achieved a good enough level of correlation with human judgment while being practical enough to be used by researchers of the field of machine translation. In this thesis two automatic metrics are going to be used, BLEU (Papineni et al. [66]) and METEOR (Banerjee and Lavie [67]).

2.6.1 BLEU

Bilingual evaluation understudy (BLEU) (Papineni et al. [66]) is a precision based metric and one of the most widely used, due to its reported high correlation with human judgment. Other BLEU's favourable characteristics are that it is both very simple to calculate and inexpensive to run. The basic premise of the metric is that "the closer a machine translation is to a professional human translation, the better it is" (Papineni et al. [66]). The output value ranges from 0 to 1, although it is usually reported in a scale from 0 to 100. The higher the value, the closer the candidate translation is to the reference and therefore, the better it is. BLEU scores are reported at a corpus-level. Following Papineni et al. [66], BLEU may be defined as

$$\mathrm{BLEU} = BP \times \exp\left(\sum_{n=1}^{N} w_n log\left(p_n\right)\right), \tag{2.39}$$

where the BP is the brevity penalty used to penalize candidate translation shorter than the respective reference, and p_n are the *n*-grams modified precision values. The default implementation uses $N = 4$ and $w_n = \frac{1}{N}$. The brevity penalty is defined as

$$BP = \begin{cases} 1 & \text{if } |C| > |R| \\ e^{(1-\frac{|R|}{|C|})} & \text{if } |C| \leq |R|, \end{cases}$$

where $|C|$ and $|R|$ are the lengths of the candidate and reference translations (total number of words). Furthermore, even though BLEU was just presented as a precision based metric, it actually uses a modified precision. This modified precision differs from the normal precision by clipping the counts of each translated *n*-gram to the maximum number of times they appear in a reference translation.

The implementation used to calculate BLEU scores throughout this thesis is available online at `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`.

2.6.2 METEOR

METEOR is an automatic evaluation metric for machine translation, first proposed in Banerjee and Lavie [67], and whose current version was proposed in Denkowski and Lavie [68]. The metric is fairly more complex than BLEU, but it is also able to leverage much more information. Namely, METEOR works by aligning candidate translations with one or more reference translations. These alignments are based on several different types of matches: *exact*, if the words are identical; *stem*, if the words share the same stem; *synonym*, if the words have the same meaning; and *paraphrase*, to match phrases according to a language dependent paraphrase table (Denkowski and Lavie [68]). Furthermore, some of the parameters used to calculate METEOR are tuned to maximize the correlation with human judgment scores. The corresponding implementation, as well as further details, are available at `http://www.cs.cmu.edu/~alavie/METEOR/index.html`.

2.7 Fertility

Fertility, first presented in the context of statistical machine translation (Brown et al. [1]), represents a measure of how many target words are created for a given source word. More formally, it is defined as a random variable $\boldsymbol{\Phi_j}$, where the fertility value depends on the fertilities of the previous source sentence words, i.e. $\boldsymbol{\Phi_j} = p(\boldsymbol{\Phi_0}, ..., \boldsymbol{\Phi_{j-1}}; x)$, in which j refers to a index of a source word. The distribution of this random variable results from the parameters of an hard alignment model.

The concept has been adapted to NMT by Tu et al. [19]. In this case the fertility value does not depend on the previous values of fertilities. In particular, the fertility of a source word, Φ_j, is predicted by a feedforward neural network, given by

$$\Phi_j = N\sigma(\boldsymbol{W_j}\boldsymbol{h}_j^e), \tag{2.40}$$

in which $N \in \mathbb{R}$ is a predefined scale factor that dictates the maximum number of words a source sentence may produce, $\sigma(.)$ is the sigmoid activation function, $\boldsymbol{W}_j$ is a weight matrix, and $\boldsymbol{h}_j^e$ is the hidden state for a given source word x_j. The encoded representation $\boldsymbol{h}_j^e$, is obtained with a bi-directional RNN, and thus, encompasses information about the whole sentence.

A posterior work by Gu et al. [69] also uses the concept of fertility, namely by introducing a *fertility predictor* module into their Non-Autoregressive Transformer model for NMT. The fertility values for each source word are obtained through a single-layer neural network, followed by a softmax classifier. Despite being used in a different NMT model architecture, the hidden vectors from the encoder are still the input used to obtain the fertilities. This is to be expected due to the very definition of fertility.

2.8 Summary

This chapter introduced the necessary background for the proposed work. First, both feedforward and recurrent neural networks (RNN) were presented. In particular, due to the recurrent nature of natural language, RNNs are particularly suitable to problems in natural language processing. Then, the most important concepts related to language models were discussed. Namely, it was followed an intuitive progression, from simpler n-gram language models, to the more widely used recurrent neural language models. Moreover, a brief overview on the several ways of representing words was presented, as well as the concept of perplexity as a way of evaluating the performance of language models.

The introduced recurrent language models are a natural precursor to neural machine translation (NMT). Thus, the problem of NMT was then defined and further explained, using the encoder-decoder approach with the attention mechanism, while making use of RNNs. Apart from mentioning how these models are trained, other aspects, such as the generation of translations with the trained model by making use of beam search, were approached. One of the key aspects of this work is the chosen attention transformation. Therefore, two alternatives to softmax were introduced: sparsemax (Martins and Astudillo [56]), and constrained softmax (Martins and Kreutzer [57]). Sparsemax has the advantage of producing sparse probability distribution, whereas constrained softmax makes it possible to define a

vector of upper bounds.

Next, several existing approaches to the problem of adequacy in NMT were discussed. In particular, the described approaches are mainly related to mitigating the three highlighted issues in Section 1.1: over-translations, under-translations, and poor performance with rare words and out-of-domain data. Then, two of the most widely used metrics in machine translations tasks, BLEU (Papineni et al. [66]) and METEOR (Denkowski and Lavie [68]), were presented, as they will be used throughout the work as a way of assessing the overall performance of the proposed approaches. Finally, the concept of fertility (Brown et al. [1]), i.e., how many words in the target language word are created for a given source word, was introduced.

Chapter 3

Machine Translation Adequacy Metrics

This chapter presents two automatic metrics for machine translation evaluation, first introduced in our work, Malaviya et al. [33], and further extended in this thesis. The goal of both metrics is to provide the user a better intuition on how the model is behaving in terms of over and under-translations, which render themselves as repetitions and dropped words. As mentioned in Section 1.1, these two issues are symptoms of poor adequacy. In particular, $\mathrm{REP}_{\mathrm{Score}}$ accounts for over-translations and $\mathrm{DROP}_{\mathrm{Score}}$ accounts for under-translations. Implementations for both metrics are available online.[1]

First, the overall description of both metrics is presented. Next, a simple example is provided to show how both metrics are calculated. Then, the evaluation metrics are applied to concrete translations tasks, and it is reported how they correlate with human judgment. This is achieved by calculating the Pearson correlation between the proposed metrics, and both BLEU scores and human annotated scores, for over and under-translations.[2] The yielded coefficient of correlation ranges from -1 to 1, with both extreme values implying a strong linear correlation between the variables being tested, and 0 implying their independence. The Pearson correlation coefficient has a set of properties which make it appealing: it is independent to changes of origin and changes of scale; and it is symmetric. Finally, the proposed metrics are compared with two metrics posterior to our work, OTEM and UTEM (Yang et al. [70]), that target the same issue of over and under-translations, respectively, and whose implementation is available online.[3] The idea behind these metrics is to "count the lexical differences between gold reference and system translation, and provide quantitative measurements according to the mismatched *n*-grams" (Yang et al. [70]). With regard to over-translations, the idea is somewhat close to the proposed $\mathrm{REP}_{\mathrm{Score}}$, but does not further penalize known NMT *hallucinations*, such as repeating the same word consecutively. On the other hand, the UTEM under-translation formulation is very different from the proposed $\mathrm{DROP}_{\mathrm{Score}}$. It is possible to argue that the same sentence may be written correctly in different ways and therefore, differences between *n*-gram counts could lead erroneously to the conclusion that a candidate translation

[1] Code available online at: `https://github.com/PedroMLF/MT-Adequacy-Metrics`
[2] Calculated with: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html`
[3] Code available online at: `https://github.com/DeepLearnXMU/Otem-Utem`

is dropping words. In this case, an align-based approach, as used in $\mathrm{DROP}_{\mathrm{Score}}$, should be more robust and capable of measuring dropped words, in particular when using only one reference.

The human evaluation data was provided by the authors of Yang et al. [70]. As mentioned in their description, two annotators were asked to score four sets of translations, corresponding to a test corpus of 222 randomly sampled sentences from the NIST 2002 dataset[4], according to the presence of over and under-translated words, and with respect to four reference sentences. These translations were obtained by four different NMT systems, using the Chinese to English language pair. The scores go from 1 (worst score) to 5 (best score), and the final score for each output sentence is the average of the value given by both annotators. The values will be referred to as OTHJ (Over-Translation Human Judgment) and UTHJ (Under-Translation Human Judgment). Similarly to Yang et al. [70], in order to obtain a corpus level value for these scores, the corresponding sentence level values are averaged for each system, resulting in a single value for each of the four systems. Furthermore, all the values reported in this chapter will refer to evaluations made on the aforementioned Chinese to English candidate translations.

3.1 REP-score

$\mathrm{REP}_{\mathrm{Score}}$ (Repetition Score) is an automatic metric whose value indicates how problematic are over-translations for a given model. The intuition for $\mathrm{REP}_{\mathrm{Score}}$ is that, if the candidate translation's *n*-grams counts are much larger than the respective counts in the reference translation, then it is very likely that the model is over-translating. Another aspect that is widely seen in machine translation outputs are the cases in which words are generated repeatedly and consecutively. $\mathrm{REP}_{\mathrm{Score}}$ tries to penalize both of these cases according to the following formula,

$$\sigma(c, r) = \lambda_1 \sum_{s \in c^n, c(s) \geq 2} \max\{0, c(s) - r(s)\} + \lambda_2 \sum_{w \in c} \max\{0, c(ww) - r(ww)\}, \tag{3.1}$$

where c and r are the candidate and corresponding reference sentences, respectively, $s \in c^n$ represents a *n*-gram in the candidate translation, $c(s)$ and $r(s)$ are its counts in the given sentences, $w \in c$ is a word in the candidate sentence, and $c(ww)$ and $r(ww)$ are counts of consecutive words appearances in the given pair of sentences. $\mathrm{REP}_{\mathrm{Score}}$ is obtained by summing $\sigma(c, r)$ over all candidate sentences, normalized over the number of words in the reference corpus and multiplied by 100. The normalization by the number of words in the reference corpus makes the metric less sensitive to the reference length. The λ weights are defined as $\lambda_1 = 1$ and $\lambda_2 = 2$, so that consecutive repeated words are more penalized, since they are a better indicative of a problem with over-translations and a common NMT output *hallucination*. The default *n*-grams used are bi-grams.

Posterior work by Yang et al. [70], suggested that such metrics could exploit shorter candidate sentences in order to have lower over-translation scores, since the counts of words directly affect the final scores. Hence, we introduce a brevity penalty, defined as

[4]https://catalog.ldc.upenn.edu/LDC2010T10

$$BP = \begin{cases} 1 & \text{if } |C| > |R| \\ e^{-(1-\frac{|R|}{|C|})} & \text{if } |C| \leq |R| \end{cases}, \tag{3.2}$$

where $|C|$ and $|R|$ are the lengths of the candidate and reference translations, respectively. Therefore, $\mathrm{REP}_{\mathrm{Score}}$'s final expression is given by

$$\mathrm{REP}_{\mathrm{Score}} = 100 \times BP \times \sum_{c \in C} \sigma(c, r) \times \frac{1}{|R|}, \tag{3.3}$$

where C is the set of all candidate translations for a corpus, c is a candidate translation with r being its corresponding reference, and $|R|$ is the number of words in the set of all reference translations. The final values for this metric are non-negative and unbounded, and the lower the better. A value of 0 would be interpreted as not having any *n*-gram (with a specific value for n) occurring more than twice in the candidate translation while being less represented in the reference translation, together with not having consecutive equal words appearing in the candidate translation more often than in the reference translation.

$\mathrm{REP}_{\mathrm{Score}}$ is also capable of handling multiple references by introducing two changes. Firstly, in Equation 3.1 it is taken the minimum of the maximum difference count between the candidate and every reference sentence, both for the *n*-gram counts and the consecutive words. Secondly, both the normalization factor and the brevity penalty reference length are the total number of words of the set of reference sentences closest in that regard to the candidate translations.

3.1.1 Examples

To clarify how $\mathrm{REP}_{\mathrm{Score}}$ is calculated, a very simple example is introduced. It is assumed that all sentences are tokenized and lowercased. The corpus of reference sentences is,

- *this is a test sentence , this is short and simple .*
- *but sometimes sentences have some repetitions .*
- *sometimes the mistakes appear several times in a row .*

and the corresponding set of candidate translations is,

- *this is a sentence , this is small and simple .*
- *nonetheless sentences have repetitions have repetitions .*
- *often the error appears in a row row row .*

The first step is to calculate the $\sigma(c, r)$ for each of the candidate translations. Remembering Equation 3.1, it is necessary to find the bi-grams that appear more than once, while keeping track of the ones whose words are both the same. For the first sentence the *n*-gram *('this', 'is')* appears twice. But, since it appears twice as well in the corresponding reference, the first term of the equation will be zero. The

second term is also null so $\sigma(c_1, r_1) = 0$. For the second sentence, *('have', 'repetitions')* appears twice in the candidate translation and only once in the reference. Therefore $\sigma(c_2, r_2) = 1$, since $\lambda_1 = 1$. Finally for the third sentence, the repeated bi-gram is *('row', 'row')* and it appears twice. This bi-gram will be both penalized with the first term, since that bi-gram never appears in the corresponding reference, and in the second term. Thus, $\sigma(c_3, r_3) = 1 \times 2 + 2 \times 2 = 6$.

The only thing left to obtain $\mathrm{REP}_{\mathrm{Score}}$ is the normalization factor and the brevity penalty. The length of the candidate translation corpus is 28 words, whereas the length of the reference corpus is 29 words. This means that, since $|C| \leq |R|$, the brevity penalty will be 1.0364, calculated accordingly to Equation 3.2. The final $\mathrm{REP}_{\mathrm{Score}}$ for this simplified example is then $\mathrm{REP}_{\mathrm{Score}} = 100 \times 1.036 \times (0+1+6) \times \frac{1}{29} = 25.02$

3.1.2 Correlation with Human Judgment and BLEU

In order to further validate $\mathrm{REP}_{\mathrm{Score}}$ both the correlation with human judgment scores for over-translations (OTHJ), and with BLEU, are going to be presented. Since the Pearson correlation yields a value between -1 and 1, obtaining a high correlation implies having an absolute value close to 1. The Pearson correlation between $\mathrm{REP}_{\mathrm{Score}}$ and OTHJ is -0.929, which indicates that there is a strong linear correlation between the value of $\mathrm{REP}_{\mathrm{Score}}$ and the human assessment of the problem of over-translations, at a corpus level. The negative signal comes from the fact that for $\mathrm{REP}_{\mathrm{Score}}$, the lower, the better, whereas for OTHJ it is the other way around. The value is comparable with the one yielded by the OTEM metric (Yang et al. [70]), which results in a similar Pearson correlation of -0.948 for the same 4 NMT systems. Despite resulting in a good correlation coefficient, a test conducted with only 4 points is far from ideal[5], which means that it is not possible to assess the statistical significance of the results obtained. Nonetheless, this experiment is reported, as it seems to be the standard way of correlating metric scores with human judgment (see Papineni et al. [66], Banerjee and Lavie [67], and Yang et al. [70]). The correlation between $\mathrm{REP}_{\mathrm{Score}}$ and BLEU is -0.042, which encourages the use of $\mathrm{REP}_{\mathrm{Score}}$, since it is providing different information than BLEU. The linear regression for both of the previous correlations may be seen in Figure 3.1. Finally, the Pearson correlation between BLEU and OTHJ is 0.249. This indicates that BLEU is not able to capture the problem of over-translation in the candidate translation, making way for other metrics that are able to do so.

3.2 DROP-score

$\mathrm{DROP}_{\mathrm{Score}}$ (Dropped Words Score) is an automatic metric that measures the opposite of the aforementioned $\mathrm{REP}_{\mathrm{Score}}$. The goal of $\mathrm{DROP}_{\mathrm{Score}}$ is to measure how present the problem of under-translation is in a given candidate translation. Broadly speaking, the metric finds the amount of source words that aligned with words from the reference translation, but that did not align with any word from the candidate translation. The reasoning behind this is that, if a given source word aligned with some word of the reference, but did not align with any word from the candidate translation, then it is likely that that source

[5] Scipy stats documentation indicates a minimum of 500 samples to obtain valid p-values.

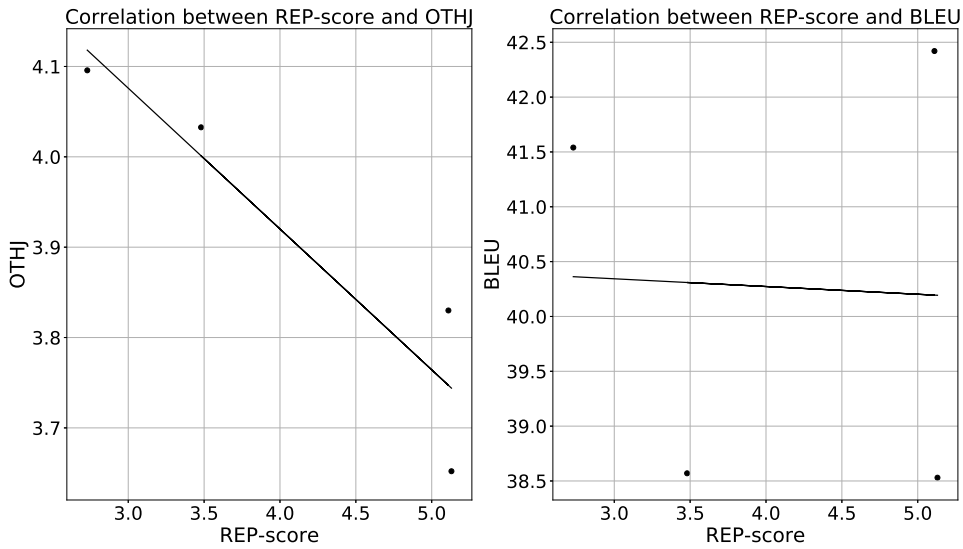


Figure 3.1: $\mathrm{REP}_{\mathrm{Score}}$ correlation with OTHJ and BLEU.

word was dropped during the process.

In order to do this, $\mathrm{fast\_align}$ (Dyer et al. [71]) is used to obtain both a set of alignments between the source and the reference corpus, and a set of alignments between the source and the candidate translation. From these set of alignments, the indexes of the aligned source words are stored, for both cases. Obtaining the difference between these two sets, and dividing that number by the size of the set of source words that aligned with reference translation words, yields the exact percentage of source words that aligned with some word of the reference but with none from the candidate translation. Although this value is interpretable, longer candidate translations will have more words for the source side to align with, and therefore, may be benefited during the calculation of this metric. This leads to the necessity of including a penalty for these cases (taking note from Yang et al. [70] regarding their under-translation metric, UTEM). $\mathrm{DROP}_{\mathrm{Score}}$'s final value is given by

$$\mathrm{DROP}_{\mathrm{Score}} = 100 \times LP \times \sum_{c \in C} \frac{|S_r \setminus S_c|}{|S_r|}, \tag{3.4}$$

where c is the candidate translation, C is the set of all candidate translation for a given corpus, S_r is the set of source word indexes that aligned with some word of the reference translation, S_c is the set of source word indexes that aligned with some word of the candidate translation, and LP is a length penalty, defined as

$$LP = \begin{cases} e^{-(1-\frac{|C|}{|R|})} & \text{if } |C| > |R| \\ 1 & \text{if } |C| \leq |R| \end{cases}, \tag{3.5}$$

where $|C|$ is the length of the candidate translation and $|R|$ is the length of the reference translation. This length penalty penalizes the score of candidate translations which are longer than the reference, and therefore, are more likely to have alignments between source and target language words. The final

values for the metric are non-negative and unbounded (due to the length penalty), and the lower the value, the better. A value of 0 would be interpreted as having every source word that aligned with some word of the reference also having been aligned with some word in the output candidate translation.

To use more than one reference in the calculation of $\mathrm{DROP}_{\mathrm{Score}}$ it is only necessary to change how S_r is obtained. To accommodate multiple references, this set of source words now contains the source words that aligned with a word from any of the references. To calculate the length penalty, the reference with the length closest to the candidate translation is chosen.

3.2.1 Examples

In this subsection a simple worked out example is presented, showing how $\mathrm{DROP}_{\mathrm{Score}}$ is computed. It is assumed, once again, that there is a tokenized and lowercased corpus of source sentences,

- *dies ist ein testsatz , es ist kurz und einfach .*
- *sätze sind nicht immer das , was sie zu sein scheinen .*
- *manchmal wird hier und da ein wort fallen gelassen .*

the corresponding set of reference sentences,

- *this is a test sentence , it is short and simple .*
- *sentences are not always what they seem .*
- *sometimes a word is dropped here and there .*

and finally the obtained translations,

- *this is a test sentence , it is small and simple .*
- *sentences are always what they seem .*
- *sometimes a word is dropped there .*

To use $\mathrm{DROP}_{\mathrm{Score}}$ it is also necessary to have the alignments between the source and the reference sentences, and between the source and the candidate translations. The goal is to obtain the source word indices that were aligned with some word from the reference but not from the candidate translation. This step is done sentence by sentence in order to obtain the sum term of Equation 3.4. To keep the example simple, only source words that were not aligned with any word from the candidate translation are mentioned, and it is assumed that all source words aligned with some reference word.

For the first sentence, all the source words were aligned with, and therefore the first term of the sum is zero. For the second sentence, no candidate sentence word aligned with the source words *nicht* and *zu*. Therefore the sum term is $\frac{12-10}{12} = 0.167$. In the third sentence, no word aligned with *hier* and *und*, thus resulting in sum term of 0.2. The set of candidate sentences has 26 words, whereas the reference has 29 words. This results in a pre-defined length penalty of 1. Therefore, the final value is given by $\mathrm{DROP}_{\mathrm{Score}} = 100 \times 1 \times (0 + 0.167 + 0.2) = 36.7$.

3.2.2 Correlation with Human Judgment and BLEU

Similarly to what was reported for $\mathrm{REP}_{\mathrm{Score}}$, the correlation for corpus-level scores between $\mathrm{DROP}_{\mathrm{Score}}$ and both BLEU and human judgment scores for under-translations (UTHJ), are presented. The Pearson correlation between $\mathrm{DROP}_{\mathrm{Score}}$ and UTHJ is -0.882. Nonetheless, it is important to mention one aspect that may cause this value to be lower than the actual correlation. $\mathrm{DROP}_{\mathrm{Score}}$ requires training an aligner to obtain the alignments. For the provided human annotated data, only a small number of source-target language sentence pairs are available to train it and therefore, the aligner might not be performing at its best. Nonetheless, it is still able to outperform UTEM (Yang et al. [70]), which yields a correlation of -0.790.

The correlation between $\mathrm{DROP}_{\mathrm{Score}}$ and BLEU is -0.856, meaning that there is a significant correlation between both. Moreover, the correlation between BLEU and UTHJ is 0.913. A possible way of interpreting the last value, namely when compared with the lower value reported for the correlation between BLEU and OTHJ, is that for this set of translations, BLEU is more affected by words being dropped than repeated. The correlation values also seem to indicate the bigger necessity of metrics concerned with the problem of over-translation than with the problem of under-translation. Once again, the number of samples used to calculate the correlation is small and therefore, it is not possible to produce p-values to ascertain the statistical significance of the reported values.

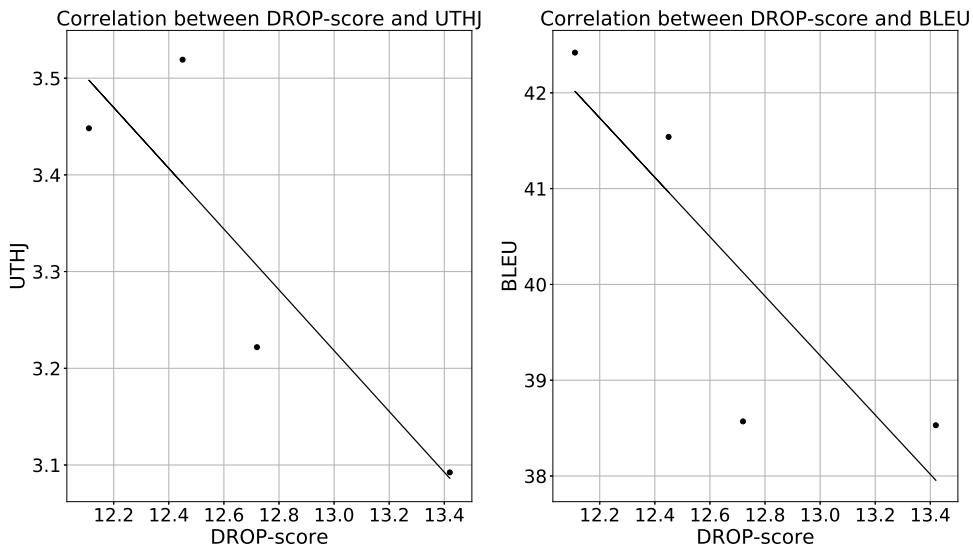


Figure 3.2: $\mathrm{DROP}_{\mathrm{Score}}$ correlation with UTHJ and BLEU.

3.3 Summary

This chapter presented two metrics, each targeting one of the possible causes of adequacy issues in translations. Namely, the issues with repetitions and dropped source words are respectively measured by $\mathrm{REP}_{\mathrm{Score}}$ and $\mathrm{DROP}_{\mathrm{Score}}$. Both metrics result in a non-negative, unbounded score, where the lower

the score, the better the translation is, with regard to the corresponding over or under-translation problem. Both metrics, besides being formally defined, were also found to be strongly correlated with human judgment data measuring the aforementioned issues. This positive result encourages their use throughout this thesis, as an automatic metric that provides us more information about the quality of the output translations being analyzed.

Chapter 4

Fertility-Based Neural Machine Translation

One of the goals of this work is to propose methods that help neural machine translation (NMT) models output translations more robust to the problems of over and under-translation of the source sentence. These two issues are symptoms of poor adequacy, and have been identified as so in Section 1.1. To overcome these, it is suggested a fertility based approach to neural machine translation, that also takes advantage of different attention mechanisms.

The chapter is structured by first describing Constrained Sparsemax, a constrained approach to the already existing sparsemax transformation (Martins and Astudillo [56]). Then, the fertility-based neural machine translation approach is introduced, describing how the concept of fertility (Brown et al. [1]) and different attention mechanisms may be used to obtain translations with better adequacy.

The work presented throughout the chapter is heavily based on our work, Malaviya et al. [33], published in the *2018 Annual Meeting of the Association for Computational Linguistics* (ACL), and also one of the contributions of this thesis.

4.1 Constrained Sparsemax

Constrained sparsemax, proposed in Malaviya et al. [33], is an activation function based on sparsemax and that differs from it by introducing upper bounds in the attention scores it returns. It is calculated by solving the problem

$$\begin{aligned} &\mathrm{csparsemax}(\boldsymbol{z};\boldsymbol{u}) := \underset{\boldsymbol{\alpha}\in\Delta^{J-1}}{\arg\min}\, ||\boldsymbol{\alpha}-\boldsymbol{z}||^2, \\ &\text{subject to} \quad \boldsymbol{\alpha} \leq \boldsymbol{u} \end{aligned} \tag{4.1}$$

where $\boldsymbol{z} \in \mathbb{R}^J$ is a vector of scores, $\boldsymbol{u} \in \mathbb{R}^J$ is the upper bounds vector, and $\boldsymbol{\alpha}$ are the resulting attention probabilities. Therefore, obtaining the attention scores at a given decoding step t, may be simply put as

$$\boldsymbol{\alpha_t} = \text{csparsemax}(\boldsymbol{z_t}; \boldsymbol{u_t}). \tag{4.2}$$

Further considerations on this attention transformation may be found in Appendix A.

4.2 Overall Description

This section defines the main points of the fertility-based NMT approach. The first step is to explain how the concept of fertility may be paired with sparse and constrained transformations to help mitigate the issue of both over and under-translations. Then different strategies to define the fertility values are discussed, followed by a brief discussion about a token that makes the proposed model possible, as well as an exhausting strategy that can be used to further improve the performance of the model.

4.2.1 Pairing Fertility with Sparse and Constrained Transformations

Having defined a series of alternative attention transformations, such as the sparsemax (Martins and Astudillo [56]; see Subsection 2.4.1), constrained softmax (Martins and Kreutzer [57]; see Subsection 2.4.2), and the proposed constrained sparsemax (Section 4.1), it is now necessary to define how they may be incorporated into neural machine translation models.

The most straightforward, and plausible way to do so, is to change Equation 2.25, by replacing the commonly used softmax by one of the aforementioned transformations. As previously mentioned in Subsection 2.3.2, the attention transformation outputs a probability distribution over the source words. Thus, it yields an output with dimension J, where J is the length of the corresponding source sentence. To further generalize the notation, Equation 2.25 is replaced by

$$\boldsymbol{\alpha_t} = \rho(\boldsymbol{z_t}, \boldsymbol{u_t}), \tag{4.3}$$

where ρ corresponds to any given attention transformation, $\boldsymbol{z_t} \in \mathbb{R}^J$ are the scores provided by the attention's chosen alignment model, at a given decoding step t, and $\boldsymbol{u_t} \in \mathbb{R}^J$ is the upper bounds vector. The latter is only necessary when ρ is defined as one of the constrained attention transformations, and it defines an upper bound in the attention each source word can receive.

Each of the proposed attention transformations affects the attention scores differently and thus, the obtained translations. Before addressing these differences it is necessary to establish the behavior of softmax. The most important aspect to consider is that the probability distribution yielded by softmax is strictly positive. This means that every word of a given input sentence receives some probability mass when the attention scores are calculated. Therefore, some attention is always "lost" due to the residual probability mass given to the less probable source words at each decoding step. If the source sentences are long, this may lead to a non-negligible portion of the attention being unnecessarily spent, leaving less attention to distinguish between other source words. Keep in mind that at each decoding step there is "one unit" of attention to spend, corresponding to the probability distribution over the source words.

Sparsemax, previously introduced in Subsection 2.4.1, differs from softmax by producing sparse probability distributions, i.e., some of the words may receive zero probability. Since the attention over the source words sums up to one, having probabilities of zero leads to the accumulation of attention in certain words, and therefore, more peaked distributions. This is beneficial since it allows the attention scores to focus on a few source words at each decoding step, thus, giving a better context when producing the target words.

Notice that none of the previous activation functions takes advantage of the proposed upper bounds, introduced in the new definition of the attention scores (Equation 4.3). One way of defining the upper bounds is through the concept of fertility. Namely, it is possible to create a very intuitive abstraction by defining the vector of upper bounds $\boldsymbol{u_t}$, at a given decoding step t as

$$\boldsymbol{u_t} = \boldsymbol{f} - \boldsymbol{\beta_{t-1}}, \tag{4.4}$$

where $\boldsymbol{f} \in \mathbb{R}^J$ is a vector where each element corresponds to the fertility of a source word, and $\boldsymbol{\beta_{t-1}} \in \mathbb{R}^J$ corresponds to the cumulative attention over source words, more formally, $\sum_{\tau=1}^{t-1} \boldsymbol{\alpha_\tau}$. The element-wise difference between both vectors may be interpreted as the amount of attention a given source word still has available to receive, given that it started with a "credit" of f_j. Consequently, when a source word exhausts its attention's "credit", it will stop receiving any probability mass. For this reason, constrained attention transformations are able to introduce sparsity over time. This opposes sparsemax, which is able to produce sparse probability distributions by itself. Also, by exhausting the "credit" of attention of some source words, the model will be forced to attend different words. This should help the model mitigate some coverage issues, which deem themselves as repetitions and dropped words. To further clarify how, the two following points are highlighted:

- *Over-translations* are tackled by defining a "credit" of attention each source word has available to receive. Once that "credit" is exhausted, the source word will no longer be attended. This should help minimize instance of repetitions, where the same source word is attended several times.
- *Under-translations* should be mitigated by forcing the model to spread the attention over the words which have not had their "credit" of attention exhausted or reduced.

There are two constrained transformations available: constrained softmax and constrained sparsemax. Both can leverage the introduced upper bound definition. Following the previous paragraph reasoning, it is possible to conclude that constrained softmax is able to introduce sparsity over decoding time steps. This has clear advantages for the problem of adequacy in NMT. Yet, more interesting is constrained sparsemax, which is based on a transformation able itself to generate sparse probability distributions. Furthermore, by making use of the vector of upper bounds, it is also able to generate sparsity over decoding time steps. By taking advantage of two different sources of sparsity, constrained sparsemax should provide a better control of coverage related issues.

Figure 4.1 highlights the behavior of the different attention transformations. In particular, it is possible to observe that for both sparsemax and constrained sparsemax, the projection onto the probability simplex in fact yield sparse probability distributions. Also, for the constrained transformations it is noticeable

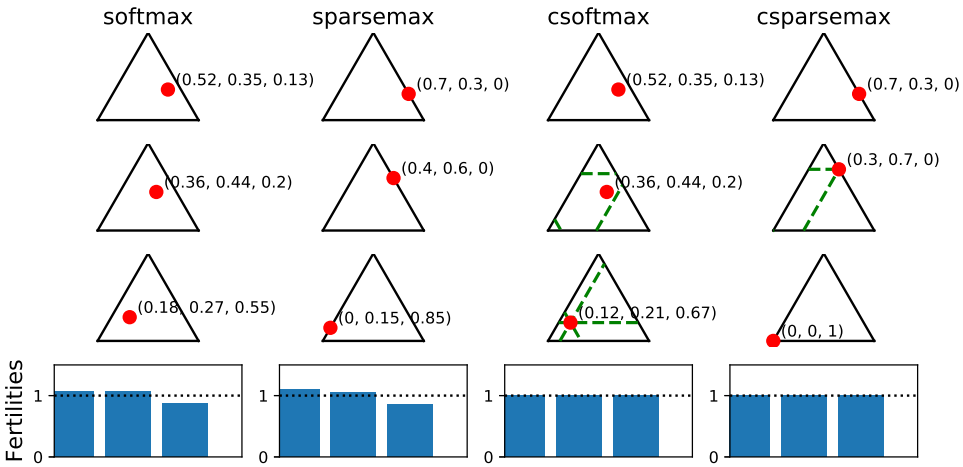


Figure 4.1: Behavior of the different attention transformations in a toy problem, with three source words. Each row corresponds to a decoding step, being the first row the first decoding step. The logit values, per row are: $\boldsymbol{z} = (1.2, 0.8, -0.2)$; $\boldsymbol{z} = (0.7, 0.9, 0.1)$; and $\boldsymbol{z} = (-0.2, 0.2, 0.9)$. The red dots show the corresponding attention values on the probability simplex. For the constrained transformations, $\boldsymbol{f} = 1$. Also for those, the green dashed lines represent the upper bounds $\boldsymbol{u}$. The last row shows the cumulative attention for each of the source words. Retrieved from Malaviya et al. [33].

the impact of the upper bound values (green dotted lines), in fact limiting the projection onto the simplex. Finally, it is possible to observe that the introduced constraints also limit the attention received by each word, according to the value of $\boldsymbol{f}$. As outcome, the attention each source word receives throughout the three decoding steps is the same for the constrained approaches, whereas in the case of softmax and sparsemax the third word is less attended than the others.

To further illustrate the differences between attention transformations, Figure 4.2 shows the attention maps obtained by the four aforementioned strategies in a concrete translation example. Attention maps are an easy to read representation of the attention probabilities yielded during translation. Furthermore, they make it easy to identify coverage issues, by finding overly-attended words, or other who have not received attention. In particular, it is possible to observe that the cases of repetitions are indeed situations in which the same source word, in this case *"todesstrafe"*, receives an high probability several times. It is noticeable as well the aforementioned sparsity over time, introduced by constrained approaches. Namely, in constrained softmax, the source word being repeatedly attended has its attention "credit" exhausted after five time steps, and thus, stops receiving any probability mass afterwards. Finally, it is possible to observe the sparsity of the probability distributions produced by sparsemax and constrained sparsemax. Even though constrained sparsemax produces a mistake, it is noteworthy that both sparse approaches are able to obtain much better soft alignments than the softmax-based approaches.

4.2.2 Fertility Bounds Initialization Strategies

One critical aspect in the proposed fertility-based neural machine translation model is how to define the vector of fertilities $\boldsymbol{f}$, used in Equation 4.4. Three possible approaches are defined:

- *Constant*, by defining the fertility upper bound for each source word as a pre-defined constant.
- *MaxAlign*, where the fertility of each source word is defined as the maximum number of target words that aligned with it, using a word aligner like $\mathrm{fast\_align}$ (Dyer et al. [71]).
- *Predicted*, which uses a bi-LSTM tagger to predict the fertility value for each source word. The fertility values used to train the model are the actual number of target words that align with a given source word, according to $\mathrm{fast\_align}$.

Each of the three proposed strategies has a different balance in terms of computational complexity versus potential quality of the defined values. The *Constant* approach is the simplest to use, but also the only which is unable to provide different values for each source word. The *MaxAlign* approach overcomes this possible issue, but requires using an external aligner. Finally, *Predicted*, albeit being the most expensive strategy to use, should also be the one providing the best individual fertility values, as a tailored value is obtained for each source word. The main advantage of *Predicted* over *MaxAlign* is that the latter defines the fertility as the maximum number of aligned words, according to the training data, whereas the former is able to leverage context information and the actual number of aligned words.

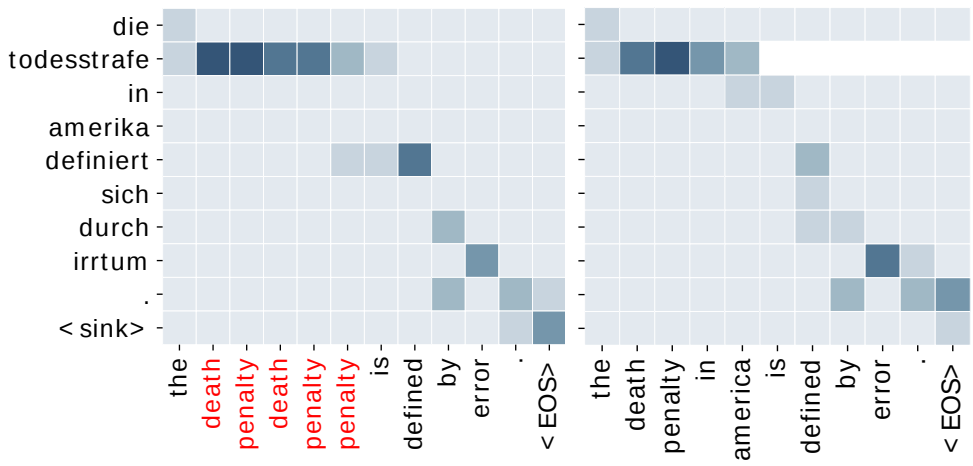
4.2.3 Sink Token and Exhaustion Strategies

At each decoding step, it is calculated a probability distribution over the source words of a given input sentence. These are the so called attention scores and, since they correspond to a probability distribution, they sum up to one. Thus, exactly a single "credit" of the upper bounds is exhausted at each decoding step. This leads to one important consideration: the output translation cannot be longer than $\sum_j f_j$. Such a limitation could become a liability during training, since target sentences longer than the sum of the source words' fertility values, render any of the constrained transformations infeasible. To overcome this situation, a $< \mathrm{SINK} >$ token is appended to every source sentence. This token has unbounded fertility, and therefore solves the previously detected limitation.

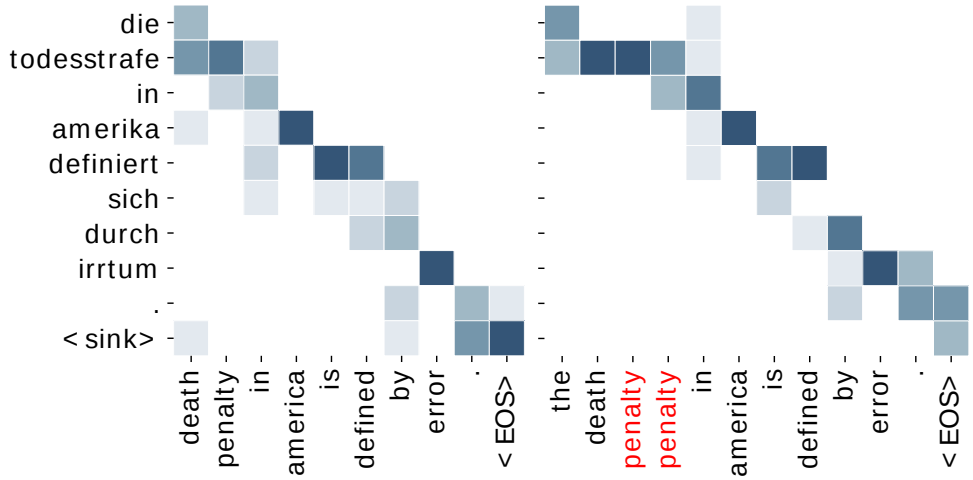
As previously mentioned, exhausting the attention a given source word has yet to receive, forces the attention scores to be spread over the remaining words. This helps the model address the issue of under-translations, by covering more source words during decoding. Nonetheless, this will only be possible after some words have their "credit" of attention exhausted. In order to have more attention spread over all source words during the whole process, it is introduced the following change: the scores $\boldsymbol{z_t}$, used in the attention transformation, are now defined as $\boldsymbol{z'_t} = \boldsymbol{z_t} + c\boldsymbol{u_t}$, where c is a constant. The higher the value of c, the more important are the source words which still have a large "credit" of attention. This will force the model to attend the words that are possibly being under-translated.

4.3 Summary

This chapter introduced several ideas that target the problems of over and under-translations in NMT. In particular, it was explored the idea of using attention transformations that produce sparse or/and con-



(a) Left: Softmax Right: Constrained Softmax



(b) Left: Sparsemax Right: Constrained Sparsemax

Figure 4.2: Exemplifying attention maps showing the different distributions for each proposed attention transformation. Reference sentence is: *"Death penalty in America is defined by error."*.

strained probability distributions. Regarding the constrained attention transformations, it is introduced the constrained sparsemax, based on the previously existing sparsemax (Martins and Astudillo [56]). In order to use constrained attention transformations it is necessary to define a vector of upper bounds $\boldsymbol{u} \in \mathbb{R}^J$. These upper bounds are defined as the difference between a vector of fertilities and a vector that keeps track of the cumulative attention over source words, up until a time step t. The benefits from using a vector of upper bounds are twofold: first, it helps avoiding repetitions by imposing a maximum value on the attention a given source word may receive; second, it should reduce the problem of dropping words, by forcing attention scores to be spread over different source words. The latter should be more noticeable as source words have their "credit" of attention exhausted. To further help the attention scores giving more importance to source words yet to be translated/attended, it is introduced a constant c, used to calculate a new set of pre-attention scores z'_t. Furthermore, three different ways of defining the fertility vector were presented: *Constant*, *MaxAlign*, and *Predicted*. The *Predicted* approach is the most complex of the three, but it is also the one that should, intuitively, lead to the best results.

Chapter 5

Guiding Neural Machine Translation Decoding with External Inputs

In the previous chapter, fertility-based neural machine translation was proposed as a way of addressing adequacy issues due to over and under-translation of source words. Nonetheless, the third adequacy issue identified in Section 1.1 also refers to the necessity of asserting that the translation obtained for a given input sentence is representative of the expected translation. This may be addressed by making NMT models more robust to two known issues: translating infrequent *n*-grams (Arthur et al. [20]), and producing the correct translation for out-of-domain sentences (Koehn and Knowles [21]).

This chapter introduces the approach by Zhang et al. [32]. The main advantage of this approach is that it only changes the decoding process by introducing the concept of *translation pieces*. Therefore, re-training the model is not necessary. This work will then be extended by introducing some changes that further improve its performance, and by addressing the case of domain adaptation, which was out of the scope of the authors' original work. The resulting approach will be referred to as *Guided-NMT*.

5.1 Overall Architecture

The proposed method of guiding neural machine translation with translation pieces (Zhang et al. [32]) is only used during the inference step of NMT and has two main parts:

- First, it uses an extra corpus to find the most similar sentences to a given input sentence. Using the retrieved sentences to obtain translation pieces, i.e., *n*-grams of target language subwords.
- Then, during decoding, it incorporates the translation pieces into the output layer, by biasing the corresponding log probabilities.

The overall procedure is summarized in Figure 5.1, where it is possible to observe many of its components (to be further explained along this section). In the following subsections the proposed approach will be presented in more detail, since some of the steps of the implementation differ from the one in Zhang et al. [32].

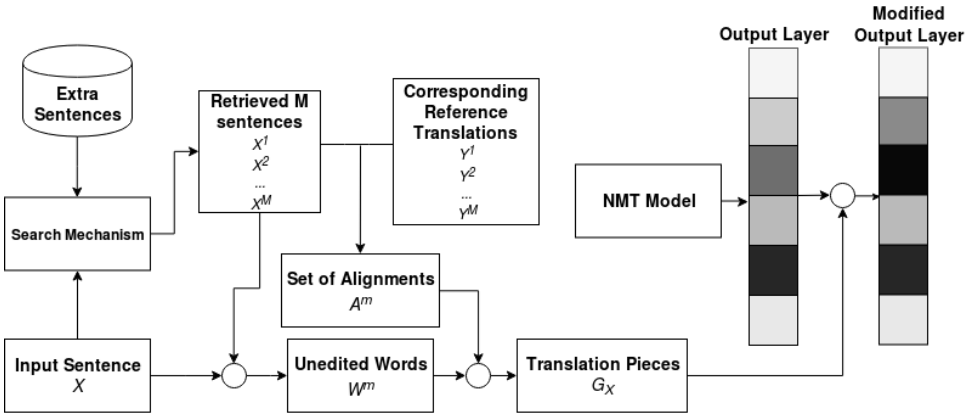


Figure 5.1: Overall architecture of the guided-NMT system with translation pieces.

5.1.1 Obtaining Translation Pieces

The first step to use this approach is to obtain the necessary translation pieces, i.e., *n*-grams of subwords of a given target sentence, which are deemed as possible partial translations of an input source sentence. The process of creating translation pieces is illustrated in Figure 5.2, where a simple example is presented. The overall process is explained throughout the next paragraphs.

First, given a source sentence X, it is necessary to create the corresponding embedding. There are plenty of possible approaches, as discussed in Subsection 5.1.4. In this work, we average the individual word embeddings (excluding *stopwords*), with each one being weighted by the inverted document frequency (IDF) of the corresponding word. The individual word embeddings are retrieved from *fastText* pre-trained embeddings (Bojanowski et al. [47]), available online.[1] *fastText* embeddings are particularly suitable for the task at hand, since they are trained using character-level *n*-gram information, which reduces the chance of having words without a corresponding vector in the pre-trained values. This is particularly relevant when domain-specific corpora are used. The sentence embeddings for every sentence of the extra corpus are also calculated, following the same procedure.

Using the vector representation for the input sentence, it is possible to efficiently collect the *M* closest extra sentences in terms of L2 distance using *Faiss* (Johnson et al. [72]). *Faiss*[2] is a library written in C++ by the Facebook AI Research group, with wrappers for Python. Some of the advantages of this library are: it is possible to use it with GPUs, allowing an even more efficient search for batches of sentences; when the size of the extra corpus grows large, there are several options of search indexes which are more memory efficient; and finally, it allows for semantic similarities between sentences to be explored, and not only counts and frequencies of words.

One important aspect to take note of, is that the sentences used to retrieve the most similar sentences do not have the byte pair encoding (BPE) (Sennrich et al. [17]) applied. This is important since pre-trained word embeddings are being used. Remember that with BPE some subwords have a "@@" separator appended, which is something unseen in the data used to pre-train the word embeddings. Therefore, using subwords during this step would harm the quality of the sentence embeddings.

[1] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md
[2] https://github.com/facebookresearch/faiss

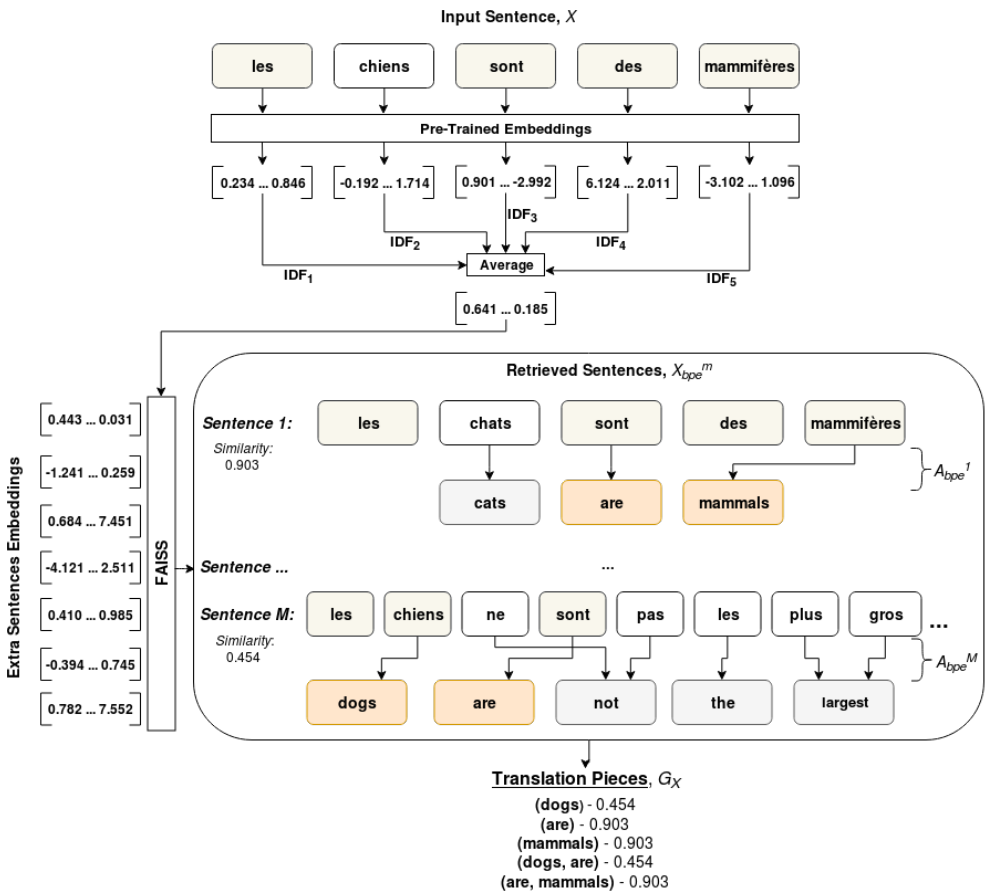


Figure 5.2: Visualization of the process of obtaining translation pieces. The examples uses French as source language, and English as target language. The input sentence *"Les chiens sont des mammifères"* is lowercased and tokenized. The yellow boxes represent words that are part of the set of unedited source words, $\boldsymbol{W}^m_{bpe}$. The orange boxes represent the target words aligned with the unedited source words.

Nonetheless, most of the subsequent steps require using BPE encoded sentences, since the log probabilities of the NMT output layer are distributed over a vocabulary of subwords and thus, translation pieces have to be *n*-grams composed of subwords. In order to provide an easier understanding, during this section it is introduced the subscript bpe, indicating whether the sentences have been encoded with BPE.

For each retrieved sentence $\boldsymbol{X}^m$, there also exists the corresponding sentence with BPE applied $\boldsymbol{X}^m_{bpe}$, where $1 \leq m \leq M$. Each retrieved sentence has a reference translation $\boldsymbol{Y}^m_{bpe}$, and a set of subword-level alignments between $\boldsymbol{X}^m_{bpe}$ and $\boldsymbol{Y}^m_{bpe}$, defined as $\boldsymbol{A}^m_{bpe}$. The alignments are obtained with $\mathrm{fast\_align}$ (Dyer et al. [71]). Moreover, it is also necessary to find the retrieved sentence's unedited subwords $\boldsymbol{W}^m_{bpe}$, i.e., subwords that are not changed when editing the input sentence $\boldsymbol{X}_{bpe}$ into a given retrieved sentence $\boldsymbol{X}^m_{bpe}$. The target language subwords aligned with the unedited subwords are used to find *n*-grams, from 1 to 4-grams, as suggested in Zhang et al. [32]. The set of subword *n*-grams that are possible partial translations for a given ($\boldsymbol{X}$,$\boldsymbol{X}^m$) pair are defined as $\boldsymbol{G}^m_X$. The union of these sets over all retrieved sentences yields $\boldsymbol{G}_X$, which is the total set of translation pieces for a given input sentence.

Finally, each translation piece in $\boldsymbol{G}_X$ is scored according to a similarity measure between the input

sentence and the corresponding retrieved sentence. When a given translation piece appears in more than one retrieved sentence, the highest similarity value is kept. The chosen similarity measure was proposed in Gu et al. [30] and is defined as,

$$simi(\boldsymbol{X}, \boldsymbol{X}^m) = 1 - \frac{d(\boldsymbol{X}, \boldsymbol{X}^m)}{\max(|\boldsymbol{X}|, |\boldsymbol{X}^m|)}, \tag{5.1}$$

where $d(\boldsymbol{X}, \boldsymbol{X}^m)$ is the edit distance between the input and a certain retrieved sentence, i.e., the minimum number of changes needed by the input sentence to transform it into the retrieved sentence. As intended, when the edit distance between two sentences is null, the similarity is maximum and the opposite happens when the the edit distance is maximum.

This process of creating translation pieces results in a list of tuples for each input sentence, where the first element of each tuple is a list of the subwords of a given translation piece, and the second element is the maximum similarity, in the range $[0, 1]$, between the input and the retrieved sentence where the translation piece was present. This list may be thought of as

$$G_X = \left[\left(\left[tp_{X11}, tp_{X12}, ..., tp_{X1n} \right], \max_{1 \leq m \leq M \wedge tp_{Xi} \in \boldsymbol{G}_X^m} simi(\boldsymbol{X}, \boldsymbol{X}^m) \right), \left(.... \right) \right], \tag{5.2}$$

where $tp_X \in \boldsymbol{G}_X$ is a translation piece of the input sentence $\boldsymbol{X}$, and tp_{Xij} corresponds to *j*-th subword of the *i*-th translation piece of the input sentence $\boldsymbol{X}$. The pseudocode for one input sentence is presented in Algorithm 3, where N is the number of sentences in the extra corpus, and it is followed the notation, $X^{i:j} = X^i, ..., X^j$.

Algorithm 3 Obtaining Translation Pieces

Input (Source Sentence): $\boldsymbol{X}$; $\boldsymbol{X}_{bpe}$;
Input (Extra Data): $\boldsymbol{X}^{1:N}$; $\boldsymbol{X}_{bpe}^{1:N}$; $\boldsymbol{Y}_{bpe}^{1:N}$; $\boldsymbol{A}_{bpe}^{1:N}$;
$E(\boldsymbol{X}) = \frac{1}{|\boldsymbol{X}|} \sum_{w \in \boldsymbol{X}} \mathrm{IDF}(w) E(w)$
for $\boldsymbol{X}^n$, $1 \leq n \leq N$ **do**
    $E(\boldsymbol{X}^n) = \frac{1}{|\boldsymbol{X}^n|} \sum_{w \in \boldsymbol{X}^n} \mathrm{IDF}(w) E(w)$
$\boldsymbol{X}^{1:M}$ = *Faiss*($E(\boldsymbol{X})$, $E(\boldsymbol{X}^{1:N})$)
for X^m, $1 \leq m \leq M$ **do**
    $G_X^m = \emptyset$
    $simi(\boldsymbol{X}, \boldsymbol{X}^m) = 1 - \frac{d(\boldsymbol{X}, \boldsymbol{X}^m)}{\max(|\boldsymbol{X}|, |\boldsymbol{X}^m|)}$
    Obtain $\boldsymbol{W}_{bpe}^m$ using $\boldsymbol{X}_{bpe}$ and $\boldsymbol{X}_{bpe}^m$
    Obtain aligned target subwords from $\boldsymbol{Y}_{bpe}^m$, using $\boldsymbol{W}_{bpe}^m$ and $\boldsymbol{A}_{bpe}^m$
    Obtain translation pieces, i.e., *n*-grams from aligned target subwords, with $1 \leq n \leq 4$
    for each translation piece, tp_{Xi} **do**
        if tp_{Xi} already in G_X^m **then**
            Choose $\max_{1 \leq m \leq M \wedge tp_{Xi} \in G_X^m} simi(\boldsymbol{X}, \boldsymbol{X}^m)$
        else
            Add tp_{Xi} to G_X^m with value $simi(\boldsymbol{X}, \boldsymbol{X}^m)$
Output: $G_X = \bigcup_1^M G_X^m$

5.1.2 Guiding NMT with Translation Pieces

The intuition behind translation pieces is that they represent *n*-grams of subwords that are likely to be present in good translation hypotheses of a certain input sentence. In order to use the previously created translation pieces, the corresponding subwords are rewarded during decoding, making it more likely that they are present in the output sentence. The reasoning behind a similar approach to this one in the context of NMT decoding may be seen in Stahlberg et al. [73]. More formally, this means that at each decoding time step *t*, the output layer's log probabilities of the subwords that correspond to translation pieces are rewarded. This way, those specific subwords will have an higher probability of being chosen during beam search. This update is given by

$$log_{p_{NMT}}(y_t|y_1, ..., y_{t-1}, \mathbf{X}) = log_{p_{NMT}}(y_t|y_1, ..., y_{t-1}, \mathbf{X}) + \lambda \sum_{n=1}^{4} tp_{value}(y_{t-n+1}, ..., y_t), \tag{5.3}$$

where λ is a weight that multiplies the value of the translation piece, and tp_{value} is a mapping between the translation pieces and its values. Note that if a given *n*-gram is not present in the collected translation pieces, then no change is made to the corresponding entry in the output layer.

One important aspect to take into consideration is that when $n \geq 2$, only the last subword of the *n*-gram is rewarded, with the previous subwords acting as context. When the full context matches the most recent subwords in the beam hypothesis, then the entry of the output layer corresponding to the desired subword is rewarded. This process may be visualized in Figure 5.3, where it is clear the aforementioned necessity of the context to be present in order to reward the last subword of a given translation piece. Furthermore, in this case, adding the translation pieces changes the output token. Namely, looking only at the log probabilities output by the NMT model, v_3 would be the next subword. However, taking into account the gathered translation pieces, the reward given to the subword v_4 made it the one with the highest probability, in fact changing the obtained translation.

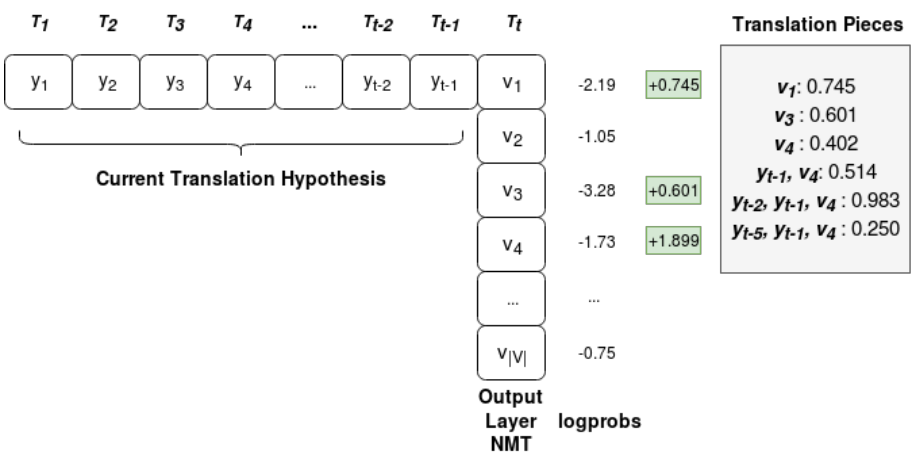


Figure 5.3: Visualization of the process of rewarding translation pieces during decoding.

5.1.3 Proposed Changes

Besides the changes already introduced in Section 5.1, where instead of using the *Lucene*[3] search engine we propose using sentence vectors and *Faiss*, the architecture suggests that some more changes should be considered. Namely,

1. The goal of decoding with translation pieces is to encourage the inclusion of specific partial *n*-grams in the output translation. Therefore, it seems plausible to reason that the translation pieces already present in a given beam hypothesis should no longer be rewarded during decoding. Namely, rewarding translation pieces regardless of their presence in the current beam hypothesis could lead to repetitions. This will be referred as the *Reward until Translated (RT)* strategy.

2. Adding translation pieces from every M similar sentence, even from the ones with low-similarity scores, not only makes the decoding procedure more cumbersome, but also rewards *n*-grams from sentences in which there is not a lot of confidence. Therefore filtering translation pairs from sentences below a given similarity threshold γ, might be beneficial for the performance of guided-NMT.

3. It should be possible to give different weights to uni-grams, and to the remaining *n*-grams. In particular, since *n*-grams, with $2 \leq n \leq 4$, have some context in the current beam hypothesis, it is possible to reason that they provide more interesting subwords to reward. Thus, Equation 5.3 is re-written as

$$log_{p_{NMT}}(y_t|y_1, ..., y_{t-1}, \mathbf{X}) + = \lambda_1 tp_{value}(y_t) + \lambda_2 \sum_{n=2}^{4} tp_{value}(y_{t-n+1}, ..., y_t). \tag{5.4}$$

   where λ_1 multiplies the value of uni-gram translation pieces, and λ_2 multiple the value of *n*-gram translation pieces, with $2 \leq n \leq 4$.

5.1.4 Sentence Embeddings

As mentioned in Subsection 2.2.3, the way natural language is represented numerically is critical to the success of neural approaches. This mainly refers to the encoding of natural language components, such as words or sentences, into fixed-size vectors. In order to encode a full sentence into a fixed-size vector, two main approaches may be followed:

- Average the vectors obtained for each word that is part of a given sentence.
- Use methods that produce a vector for a full sentence straightaway.

For the first approach, there are plenty of possible methods for obtaining the necessary word embeddings. The most common methods are *Word2Vec* (Mikolov et al. [45]) and *GloVe* (Pennington et al. [46]). Recently *fastText* (Bojanowski et al. [47]) has also provided good results by assuming that each

[3] https://lucene.apache.org/

word is further represented by its character *n*-grams. Finally, the current state-of-the-art are the *ELMo* embeddings (Peters et al. [48]). All of these approaches were more carefully presented in Subsection 2.2.3. An interesting behaviour of word embeddings is that, simply averaging them while being weighted by an inverse document frequency multiplier, or by using a maximum likelihood estimate approximation, has been shown to yield competitive sentence embeddings, with both approaches and results being presented in Arora et al. [74].

As for "pure" sentence embeddings, there are currently several approaches. The *skip-thought* model (Kiros et al. [75]), follows the same logic as *skip-grams* but applied to sentences, by being trained to predict a sentence s_i given its context s_{i-1} and s_{i+1}. Similarly, *FastSent* (Hill et al. [76]) uses the surrounding sentences to predict a given sentence, but in this case s_i is represented as the sum of the respective word embeddings (also learned by the model) and the context is treated as a bag of words, and not recursively like in *skip-thought*.

Albeit also an unsupervised approach, *quick-thought vectors* (Logeswaran and Lee [77]) tackle the issue differently, by suggesting to train a classifier that learns to pick the correct sentence s_{i+1}, that follows a given sentence s_i, from a pool of possible sentences. In this case the encoder will learn representations for sentences that should be together in a semantic space, and therefore close in the resulting embedding space.

Further work took advantage of the Stanford Natural Language Inference (SNLI) dataset (Bowman et al. [78]) to train a bi-LSTM encoder with max pooling for the natural language inference task, leading to *InferSent* (Conneau et al. [79]). In this case, the sentence representation obtained at the encoder level is used as sentence embedding. This approach is defined as supervised, since it relies on labelled data to train the encoder from which sentence representations are obtained.

Finally, Google's *Universal Sentence Encoder* (Cer et al. [80]) and Microsoft/MILA's *General Purpose Sentence Representation* (Subramanian et al. [81]) have leveraged the multi-task learning framework to create sentence embeddings. The premise of both works is that training neural models to be able to perform well on several different tasks related to natural language will enrich the representation learned for the sentences and therefore, make the embeddings more suitable for several downstream tasks.

The downside of the methods that generate sentence embeddings, when compared to averaging word embeddings, is that often the former do not offer pre-trained embeddings, or they only offer them for a strict set of languages. In comparison, pre-trained word embeddings exist for a wide array of languages. This fact, combined with the strong baseline offered by averaging individual word embeddings to create a sentence embedding (Arora et al. [74]), lead to the choice of that approach in this work. Moreover, as proposed in Arora et al. [74], the sentence embeddings used in this work were obtained by weighting each word by its inverse document frequency, more formally

$$E(s) = \frac{1}{|s|} \sum_{w \in s} \left(\log\Big(\frac{1+N}{1+N_W} \Big) + 1 \right) E(w), \tag{5.5}$$

where $E(s)$ is the final sentence embedding, $E(w)$ is the pre-trained word embedding, s is the input sentence, $|s|$ its length, N is the number of documents (in our case each sentence is a document)

and N_w is the number of documents in which a given word appears. Stopwords are excluded from the average. Summing over all words in a sentence gives us the term frequency, therefore, this could be seen as a TF-IDF approach to average the word embeddings.

Furthermore, we chose *fastText* embeddings, since the inclusion of character level n-grams reduces the amount of words out-of-vocabulary when using pre-trained embeddings. This may be particularly important when using corpus of a specific domain, as in this work, which may include several words that are not seen during training when following *Word2Vec* or *GloVe*'s approach.

5.2 Domain Adaptation using Guided-NMT with Translation Pieces

One possible application for the previous architecture that Zhang et al. [32]'s authors mention but do not explore is domain adaptation. It seems straightforward to assume that translating a set of in-domain sentences with a generic model, while using translation pieces to leverage in-domain data, should outperform using only the generic model. A possible issue with this approach lies in Equation 5.3, which is only able to reward subwords which are part of the vocabulary of the NMT model. Using a large vocabulary while employing BPE (Sennrich et al. [17]) should help reducing the number of out-of-vocabulary subwords. Nonetheless, it is possible that some of the translation piece's subwords are not part of the generic vocabulary, nor can be built using other subwords. Therefore, it is necessary to come up with a strategy to address that possibility. The proposed strategy is to incorporate in the vocabulary the translation pieces' subwords that appear during decoding, and that are not part of it. In terms of NMT models this also requires changing the decoder embedding layer and the output layer. The corresponding weights and bias are defined with the values the NMT model has learned for the *unknown* symbol.

Using the same values as the ones obtained for the *unknown* symbol means that these subwords will be given a low log probability value. Therefore, weighting the subwords that are translation pieces added during this procedure, with the same values of λ used to weight the subwords that are part of the original NMT model vocabulary, would have no practical implication in the output translation. It is therefore tested the necessity of using a specific weight value τ, for the translation pieces that were added to the vocabulary. This value opposes λ, which weights the subwords that were already part of the generic model vocabulary. The value of τ should be higher than λ due to the aforementioned log probability deficit for the unknown symbol.

To sum up, using translation pieces to perform domain adaptation requires the following:

- Use the same BPE encodings for the generic and in-domain data.
- Extend the generic vocabulary with the unseen subwords.
- Use a different weight τ, for the added subwords.

5.3 Summary

This chapter targeted two known problems of current NMT state-of-the-art approaches: translating infrequent *n*-grams (Arthur et al. [20]), and translating out-of-domain sentences (Koehn and Knowles [21]). In order to overcome those issues, it is presented an approach which leverages an extra corpus of data by creating *translation pieces*, i.e., *n*-grams of subwords that are likely to be present in the correct translation of a given input sentence (Zhang et al. [32]). The main advantage of this approach, when compared with others in the literature, is that it does not require re-training any model. In particular, the aforementioned approach only acts during the decoding. Besides describing the original work, we also go through a set of possible changes that could lead to improvements. These correspond to: no longer rewarding *n*-grams already present in a given beam hypothesis; filtering *translation pieces* based on sentence similarity; and splitting the λ weight into two different values. Finally, we presented some practical considerations on how to use *translation pieces* as way of performing domain adaptation.

Chapter 6

Experimental Analysis

In this chapter, the results of the proposed approaches to neural machine translation (NMT) are presented and discussed. First, it is presented an overview of the datasets used to conduct out experiments, as well as the description of the baseline model. Then, the automatic metrics used to measure the quality of the obtained translations are defined. Among them are $\mathrm{REP}_{\mathrm{Score}}$ and $\mathrm{DROP}_{\mathrm{Score}}$, two metrics first proposed in our work Malaviya et al. [33], and further improved in Chapter 3. The first model to be evaluated concerns the fertility-based neural machine translation approach (see Chapter 4), which also makes use of the sparse and constrained attention transformations (see Subsections 2.4.1 and 2.4.2; Section 4.1). Afterwards, the results of guiding neural machine translation (Guided-NMT) with translation pieces are presented and discussed (see Chapter 5). For this approach, two sets of experiments are devised. First, the model is evaluated in an in-domain setup, using in-domain data as both training, and extra data. Then, guided-NMT is applied to the problem of domain adaptation. This approach adds the interesting twist of using a generic model to translate an in-domain test set, while having translation pieces as the only method to leverage in-domain data. For each of the mentioned approaches, first the development set is used to fine-tune the hyperparameters, which are then used in the test set, followed by an analysis of the obtained results.

6.1 Datasets

The choice of datasets reflects the issues that the proposed models address. Namely, for the fertility-based NMT approach, the goal is to evaluate whether the existing issues related to over and under-translation of source words are mitigated. This kind of issues are mostly noticeable in small datasets. Thus, two small datasets are chosen for this set of experiments. In particular, we evaluate fertility-based NMT in two language pairs, German to English (De-En) and Romanian to English (Ro-En). For the former, it is used the IWSLT 2014 dataset (Cettolo et al. [82]).[1] For the latter, it is used the WMT 2016 data.[2] The dimensions of the training, development and test sets are summarized in Table 6.1. These datasets are going to be used throughout Section 6.4.

[1] https://wit3.fbk.eu/mt.php?release=2014-01
[2] http://www.statmt.org/wmt16/translation-task.html

Table 6.1: Training, development and test set dimensions for the fertility-based NMT experiments.

	De-En	**Ro-En**
Training	153326	560767
Dev	2169	1999
Test	6750	1999

In the case of guided-NMT, there are two main goals: the first is to enhance the capability of the model of translating words that do not appear as often in the training set; the second is to make models more robust to domain changes. As in Zhang et al. [32], we choose to work with in-domain datasets, in particular, from a domain challenging in terms of vocabulary. Thus, the choice is the UFAL Medical Corpus.[3] This corpus, which served as training data for the WMT17 and WMT18 Biomedical Task, covers several languages and includes both in-domain and out-of-domain data, accordingly tagged. We use the sentence pairs tagged as *medical_corpus*, which come from several different medical domain smaller corpus, to create the dataset to be used throughout the Guided-NMT experiments (Sections 6.5 and 6.6). From these sentence pairs, the following are removed:

- At least one of the sentences of the pair has 5 or less words.
- At least one of the sentences of the pair has 150 or more words.
- Source and target sentences are the same.
- Duplicates.

Two sets of data are created, one for the German to English (De-En) language pair, and other for the Spanish to English (Es-En) language pair. Regarding the German to English language pair, we use the top one million sentences as training data and three thousand sentences for both the development and test set. For the Spanish to English language pair, 2500 sentences are used for both the development and test set, and the remaining sentences are used as training data. The size of the created datasets may be seen in Table 6.2.

Table 6.2: Training, development and test set dimensions for the in-domain guided-NMT experiments.

	De-En	**Es-En**
Training	1000000	437922
Dev	3000	2500
Test	3000	2500

As previously mentioned, the guided-NMT approach is going to be applied to the problem of domain adaptation. Therefore it is necessary to use larger generic datasets to train the corresponding models. The generic models are going to be trained with the Europarlv7 data.[4] The generic data development set for the German to English language pair is from the WMT17, whereas for the Spanish to English experiments, it is from the WMT14. Table 6.3 shows the total number of sentences available. The generic models are used in Section 6.6.

[3] `https://ufal.mff.cuni.cz/ufal_medical_corpus`
[4] `http://www.statmt.org/europarl`

Table 6.3: Training and development set dimensions for the generic NMT model.

	De-En	**Es-En**
Training	1920209	1965734
Dev	3000	3057

Finally, every dataset is pre-processed accordingly to the same steps. First, the data is tokenized and lowercased with the Moses scripts (Koehn et al. [83]), and BPE (Sennrich et al. [17]) is applied, using 32000 merge operations. This vocabulary is joint in the fertility-based experiments, and language-specific in the guided-NMT experiments. For the fertility-based NMT experiments it is appended the $< \mathrm{SINK} >$ token to the source language sentences. The necessity of this token when using the constrained attention transformations was explored in Subsection 4.2.3. Before training the model, it is also necessary to use OpenNMT-py's pre-processing script(Klein et al. [84]), which is ran with the default parameters.

6.2 Baseline Model and Decoding

All the experiments presented in this chapter use models trained with the OpenNMT-py framework (Klein et al. [84]). In particular, the way the models are trained is kept constant. We train an Encoder-Decoder model, using a 2-layer Bi-LSTM for the encoder and a 2-layer LSTM for the decoder. Furthermore, we use dropout of 0.3 (Srivastava [85]), and use the default attention mechanism, Luong's General Attention (Luong et al. [15]). All the other parameters are kept as default, unless otherwise stated. While training, each model is evaluated on the development set after each epoch. We choose the model that yields the lowest perplexity on these evaluations. As for the decoder, we use beam search, with a beam size of 5. Any produced unknown symbols is replaced by the source word with the highest attention score, for a given decoding step (Luong et al. [55]).

6.3 Metrics

In this chapter, the translations are measured in terms of:

- **BLEU** (Papineni et al. [66]), a precision based automatic metric to measure the quality of machine translation outputs, presented in subsection 2.6.1.
- **METEOR** (Denkowski and Lavie [68]), which takes into account more features of the produced and reference translation, such as synonyms and word stems (see Subsection 2.6.2).
- **REP-Score** and **DROP-Score**, proposed in Malaviya et al. [33], and modified and validated in Chapter 3, which show how the model is behaving in terms of over and under-translations, respectively.

Furthermore, the length ratio (**LR**) between the proposed translation and the reference translation is also reported. This allows us to be aware of possible cases in which the length ratio of the translations

changes dramatically.

The choice of metrics reflects two different aspects. First, by reporting commonly used automatic metrics for NMT, such as BLEU and METEOR, we are able to draw conclusions about the overall increase in performance due to the proposed methods. Furthermore, there is no widely used metric in machine translation that argues targeting specifically adequacy. Secondly, by making use of the proposed $\mathrm{REP}_{\mathrm{Score}}$ and $\mathrm{DROP}_{\mathrm{Score}}$, it is possible to measure how impactful are the problems of over and under-translations. As mentioned in Section 1.1, these two problems are symptoms of adequacy issues.

Finally, when relevant, it is used paired bootstrap resampling to test the statistical significance of the BLEU scores obtained by the translation output by two different systems (Koehn [86]).[5] The null-hypothesis of this test is that the same system generated the two candidate translations. We use a p-value of 0.01, and thus, finding p-values below that value is considered a strong evidence against the aforementioned null hypothesis.

6.4 Fertility-Based Neural Machine Translation

The fertility-based approach to neural machine translation discussed in Section 4 and also part of our published work in Malaviya et al. [33] was implemented in OpenNMT-py (Klein et al. [84]). The corresponding implementation is online.[6]

6.4.1 Hyperparameter Tuning

Before obtaining results for the test set, it is necessary to find the optimal set of hyperparameters and the fertility strategy to use. In order to achieve the optimal set, first, we find the optimal value for the constant fertility approach, for both constrained softmax and constrained sparsemax. Then, it is found which of the fertility strategies is optimal. Finally, we find the value of c (see Subsection 4.2.3) for the corresponding fertility strategies. All the results in this subsection refer to the development set and we will use the German to English language pair for the hyperparameter tuning. It is important to take in consideration that each different set of parameters requires training a new model. Even though the datasets are small, a exhaustive search for values would be cumbersome while not adding much to the overall quality of the model. Therefore, a small, but informative, choice of hyperparameters is chosen for each of the following subsections.

Constant Fertility Value

Defining the fertility as a constant requires finding the optimal constant value before drawing comparison between the different strategies. In this case, it is necessary to find the best value for each of the constrained attention transformations. Looking at the results (Table 6.4) it is possible to observe that the differences between the different constant values, for the same transformation, are not large. In particu-

[5] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/analysis/bootstrap-hypothesis-difference-significance.pl
[6] Code available online at: `https://github.com/Unbabel/sparse_constrained_attention`

lar, it is possible to observe that there is not enough statistical evidence to reject the hypothesis that the yielded translations come from the same system, for each of the constrained attention transformations.

Choosing the best value for the constant fertility will then resort to an overview of the other available metrics. For those, it is possible to observe that a constant value of 1.5 has lower scores for $\mathrm{REP}_{\mathrm{Score}}$ while having the lowest, or close to the lowest, $\mathrm{DROP}_{\mathrm{Score}}$. Hence, this is the chosen constant for the fertility, for both attention transformations.

Table 6.4: Fertility-based NMT - Choice of constant fertility values. † indicates translations whose difference in BLEU, with regard to the model with the highest score, is not statistically significant ($p < 0.01$), for each constrained attention transformation.

	Model	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
De-En	CSoftmax-Fixed-1.5	**15.67**	**23.21**	**5.75**	**6.44**	1.035
	CSoftmax-Fixed-2	**15.56**†	23.09	6.61	6.62	1.032
	CSoftmax-Fixed-3	**15.66**†	23.12	5.84	6.75	1.026
	CSparsemax-Fixed-1.5	**16.30**†	23.48	**4.48**	6.44	1.006
	CSparsemax-Fixed-2	**16.36**	23.58	5.76	**6.39**	1.019
	CSparsemax-Fixed-3	**16.34**†	**23.61**	5.50	6.52	1.018

Fertility Strategy

The next step is to verify which of the proposed fertility strategies works best. As introduced in Subsection 4.2.2, the fertility values for a given source word may be defined: as a constant (*Constant*); as the maximum number of target words that aligned with that source word, in the whole corpus (*MaxAlign*); or may be predicted by an bi-LSTM tagger (*Predicted*). This bi-LSTM tagger is trained on data with the actual number of words that aligned with a given source word. All the previously mentioned alignments are obtained with $\mathrm{fast\_align}$ (Dyer et al. [71]). Regarding the *Predicted* strategy, it is also important to mention that at training time the corresponding model uses the actual fertilities, while the predicted ones are used during inference. This should help the model correctly learn its parameters, and avoid having the propagation of errors due to poor predictions by the fertility predictor.

The results obtained for the different types of fertility are shown in Table 6.5, for both the constrained softmax and constrained sparsemax. Once again, there is no statistical evidence that allows us to reject the hypothesis that the translations come from the same model, with regard to BLEU scores. Therefore, we are going to choose the model that seems to work better overall. For the constrained softmax transformation, using a fixed fertility yields the best results across all metrics and so it seems to slightly outperform the others. Regarding constrained sparsemax, no fertility strategy is able to clearly show improvements. Therefore we choose the *predicted* approach, as it is the fertility strategy that has the potential of producing the best fertility values.

Choice of c

Having defined a fixed value of 1.5 and *Predicted*, as the optimal fertility strategies for constrained softmax and constrained sparsemax, respectively, it is now necessary to tune the value of c. As presented in Subsection 4.2.3, this value will force the model to attend words that have received less attention up until a given decoding step.

The results presented in Table 6.6 allow us to draw some conclusions. In particular, for the con-

Table 6.5: Fertility-based NMT - Choice of fertility strategy. † indicates translations whose difference in BLEU, with regard to the model with the highest score, is not statistically significant ($p < 0.01$), for each constrained attention transformation.

	Model	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
De-En	CSoftmax-Fixed-1.5	**15.67**	**23.21**	**5.75**	**6.44**	1.035
	CSoftmax-MaxAlign	**15.51**†	23.17	5.91	6.71	1.020
	CSoftmax-Predicted	**15.39**†	23.17	7.04	6.80	1.037
	CSparsemax-Fixed-1.5	**16.30**†	23.48	**4.48**	6.44	1.006
	CSparsemax-MaxAlign	**16.37**†	**23.90**	5.90	6.43	1.023
	CSparsemax-Predicted	**16.53**	23.39	5.17	**6.29**	1.001

strained softmax transformation, a value of $c = 0.6$ is able to obtain the best scores across all metrics. As for constrained sparsemax, it seems that $c = 0.4$ and $c = 0.6$ have virtually the same performance, thus, we choose $c = 0.4$ over the other value, for no strong reason.

Table 6.6: Fertility-based NMT - Choice of the value c. † indicates translations whose BLEU is not statistically significantly different ($p < 0.01$) with regard to the model with the highest score, for each constrained attention transformation.

	Model	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
De-En	CSoftmax-Fixed-1.5-0.0	15.67	23.21	5.75	6.44	1.035
	CSoftmax-Fixed-1.5-0.2	15.38	23.04	5.86	6.41	1.037
	CSoftmax-Fixed-1.5-0.4	15.64	22.99	5.60	6.15	1.020
	CSoftmax-Fixed-1.5-0.6	**16.44**	**23.42**	**4.32**	**5.84**	1.000
	CSoftmax-Fixed-1.5-0.8	**16.06**†	23.09	4.47	6.27	0.994
	CSparsemax-Predicted-0.0	**16.53**	23.39	5.17	6.29	1.001
	CSparsemax-Predicted-0.2	16.01	23.47	5.63	6.26	1.024
	CSparsemax-Predicted-0.4	**16.33**†	**23.83**	4.16	**5.70**	1.021
	CSparsemax-Predicted-0.6	**16.39**†	23.78	**4.15**	6.13	1.010
	CSparsemax-Predicted-0.8	15.53	23.29	5.07	5.95	1.034

6.4.2 Evaluation and Analysis

In the previous subsection we were able to find the optimal strategy for both constrained attention transformations. Namely, for constrained softmax, the optimal strategy was to define a fixed fertility of 1.5 for every source word, and the value c as 0.6, whereas for constrained sparsemax, the optimal was predicting the fertility, with a c of 0.4. In this subsection we further evaluate the proposed fertility-based NMT model. The first step will be to observe how the optimal setup found for the development set behaves in the test set. Furthermore, the values for two other state-of-the-art approaches for the problem of coverage in NMT are shown. Then, some examples, where using the proposed approach yields significantly better translations, are going to be discussed. Finally, it is shown an attention plot, in particular, to highlight the sparse probability distributions yielded by the proposed attention transformations.

Test Set Results

Besides using the different attention transformations already mentioned throughout this work, two other approaches to mitigate coverage-related adequacy problems during translation are going to be used. These approaches are:

- *CovPenalty*, using Wu et al. [25]'s coverage penalty and length normalization (see Subsection 2.5.4). The corresponding α and β values are tuned in the development set, using grid search on $\{0.2k\}_{k=0}^{5}$.

- *CovVector*, adapted from Tu et al. [19] (see Subsection 2.5.1). To use this in the proposed fertility-based NMT, the calculation of pre-attention scores using Luong's general alignment model is changed to $\boldsymbol{z}_{t,j} = \boldsymbol{h}_{t-1}^{d}{}^{T}(\boldsymbol{W}\boldsymbol{h}_j^e + \boldsymbol{v}\boldsymbol{\beta}_{t-1,j})$, where $\boldsymbol{v}$ is a matrix of parameters that multiplies the coverage vector, $\boldsymbol{\beta}_{t-1,j}$.

Looking at the results in Table 6.7, it is possible to draw several conclusions. The sparse attention transformations (sparsemax and constrained sparsemax) have a positive impact in both $\mathrm{REP}_{\mathrm{Score}}$ and $\mathrm{DROP}_{\mathrm{Score}}$, as does constrained softmax, which is able to introduce sparsity in the probability distributions over time. Finally, the proposed fertility-based NMT models are able to, on average, outperform two of the approaches existing in the literature, here presented as *CovPenalty* and *CovVector*.

In particular, for the German to English language pair, constrained sparsemax with predicted fertility and $c = 0.4$ is able to outperform every other approach across all metrics. When compared with the baseline softmax, it shows an improvement of around 1.5 BLEU points, 0.8 METEOR points and it is able to reduce $\mathrm{REP}_{\mathrm{Score}}$ by 31.1% and $\mathrm{DROP}_{\mathrm{Score}}$ by 19.6%.

For the Romanian to English language pair, even though the difference in performance is not as sound as for the previous language pair, the fertility-based NMT is still able to produce strong BLEU and METEOR scores while effectively reducing the adequacy related metrics. When compared with the baseline softmax, using constrained softmax with a fixed fertility of 1.5 and $c = 0.6$ is able to reduce $\mathrm{REP}_{\mathrm{Score}}$ by 18.5% and $\mathrm{DROP}_{\mathrm{Score}}$ by 3.5%.

Table 6.7: Fertility-based NMT - Choice of attention transformation. † indicates translations whose difference in BLEU, with regard to the model with the highest score, is not statistically significant ($p < 0.01$), for each constrained attention transformation.

	Model	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
De-En	Softmax	29.18	31.21	4.47	6.07	0.956
	Softmax + CovPenalty	29.29	31.29	4.53	5.93	0.960
	Softmax + CovVector	29.66	31.54	4.03	5.47	0.960
	Sparsemax	29.56	31.33	4.41	5.77	0.949
	Sparsemax + CovPenalty	29.59	31.38	4.43	5.66	0.954
	Sparsemax + CovVector	29.98	31.68	3.70	5.65	0.953
	CSoftmax-Fixed-1.5-0.6	**30.26**†	31.86	3.49	5.35	0.947
	CSparsemax-Predicted-0.4	**30.52**	**32.07**	**3.08**	**4.88**	0.958
Ro-En	Softmax	**30.43**†	32.60	2.32	6.01	1.005
	Softmax + CovPenalty	**30.44**†	32.63	2.34	5.96	1.007
	Softmax + CovVector	**30.68**†	**32.79**	2.09	5.86	1.004
	Sparsemax	30.12	32.39	2.11	6.08	1.001
	Sparsemax + CovPenalty	30.09	32.46	2.12	5.92	1.004
	Sparsemax + CovVector	**30.44**†	32.54	1.91	5.93	1.001
	CSoftmax-Fixed-1.5-0.6	**30.87**	32.74	**1.89**	**5.80**	0.993
	CSparsemax-Predicted-0.4	**30.35**†	32.69	2.08	5.80	1.004

Examples

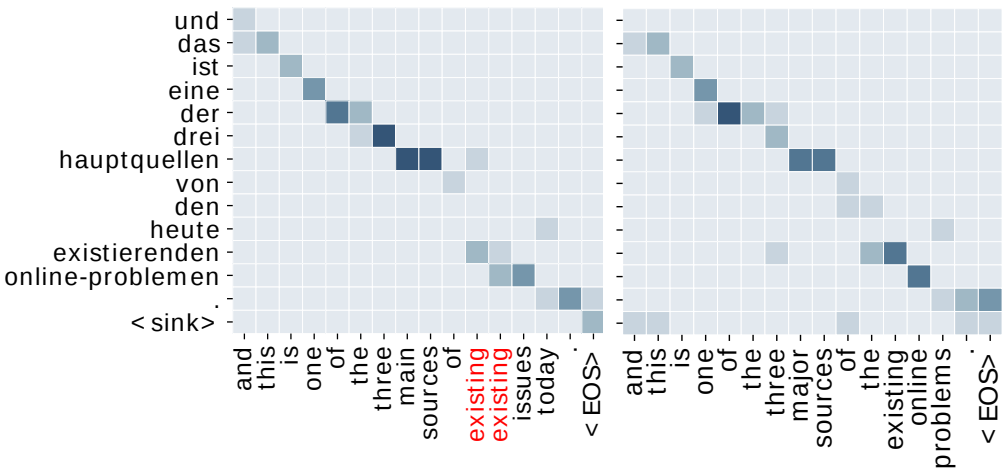
The main goal of fertility-based NMT is to improve the quality of translations with regard to the presence of repetitions and missing words. Table 6.8 introduces two examples where the obtained translations have improved when compared to the baseline softmax. Both examples are for the German to English language pair, and use the previously defined fertility strategies. In particular, it is possible to observe both over and under-translations present throughout the produced sentences. In the first example, only the constrained sparsemax model is able to avoid repeating any expression (such as *"existing"* when using the softmax), or dropping any word (such as *"online"* in the case of the softmax and sparsemax, and *"today"* in the case of constrained softmax). In the second example, even though none of the translations is completely correct, constrained sparsemax is able to avoid making any repetition or dropping any word. All the other attention transformations end up introducing some kind of over or under-translation error. More examples are available in Appendix B, Section B.1.

Table 6.8: Fertility-Based NMT Examples. The golden color indicates the portion of the reference that the example targets, the **red color** indicates the mistranslations/repetitions and finally, the *blue color* highlights correct translations of the parts of the sentence being focused. The <**MISS**> symbol refers to portions of the sentence that were missed.

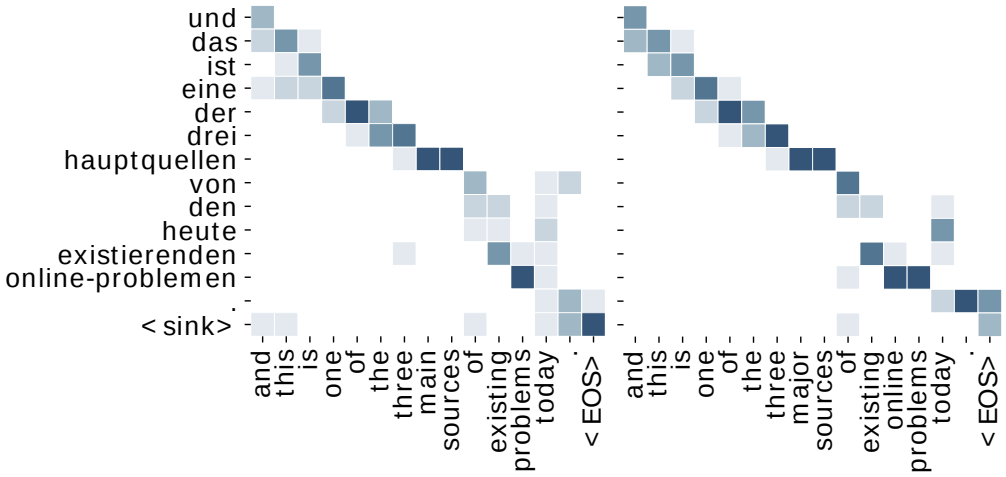
	Example 1
Reference	And this is one of the main three sources of online problems today.
Softmax	And this is one of the three main sources of **existing existing** <**MISS**> issues *today*.
Sparsemax	And this is one of the three main sources of existing <**MISS**> problems *today*.
CSoftmax	And this is one of the three major sources of the existing *online* problems <**MISS**>.
CSparsemax	And this is one of the three major sources of existing *online* problems *today*.
	Example 2
Reference	And we went to school districts where now companies are making their way into cash-strapped schools all across america .
Softmax	And we went to **school school in school school** <**MISS**>, where companies were moving their way into their **mouths** schools, all across america .
Sparsemax	And we went in **school school** <**MISS**>, where companies have their way into **their way into their homes**, all across america.
CSoftmax	And we drove in *school district*, where companies, **in their way**, are making their way into <**MISS**> *schools*, in all america.
CSparsemax	And we drove in *school districts*, where companies are getting their way out into **their hands** *schools*, across america.

Attention Visualizations

One of the side benefits of the attention mechanism is the possibility of creating attention plots, i.e., plots in which it is possible to easily observe the probability distribution over the source words, for each output word. Figure 6.1 shows the attention maps for the first example of Table 6.8 (German to English). One first interesting thing to notice is the effect of compound words in German, which lead to multiple target words attending correctly the same source words, e.g., *"hauptquellen"*. Also, it is possible to observe that, when using softmax, the repetition of *"existing"* is not entirely a coverage issue. In fact, despite some attention still being given to the source word *"existierenden"*, the largest portion is in *"online-problemen"*. So, it is partially fault of the language model as well. Another issue of the translations is dropping the word *"online"*, which is part of a compound word in German. For instance,



(a) Left: Softmax Right: Constrained Softmax



(b) Left: Sparsemax Right: Constrained Sparsemax

Figure 6.1: Fertility-based NMT - Attention maps showing the different distributions for each proposed attention transformation. Reference sentence is: *"And this is one of the main three sources of online problems today."*.

sparsemax is only capable of translating that compound word partially because it only aligns strongly with it once. Furthermore, the fact that softmax based approaches always attribute some probability mass to words, has two noticeable affects. First, even a small probability given to source words may affect the output word. This may be observed for constrained softmax, in the time step that outputs the word *"problems"*. Secondly, it prevents that more probability mass is given to the correct source words. This may be seen when comparing the attention plots from the softmax-based approaches, with the one from constrained sparsemax. In particular, the soft alignments for the constrained sparsemax match almost perfectly the correct translations. More examples of attention maps are shown in Appendix Section B.2.

6.5 Guided Neural Machine Translation

To test the effectiveness of Guided-NMT, we implemented the proposed architecture in the OpenNMT-py framework (Klein et al. [84]). The implementation largely follows Section 5.1 description and it is available online.[7] In this set of experiments we use the in-domain data (as described in Section 6.1) to train the model and also to evaluate its performance. Furthermore, guided-NMT is able to access the full training corpus while creating the translation pieces.

6.5.1 Hyperparameter Tuning

As discussed in Chapter 5, there are several tunable hyperparameters in order to obtain the best performance possible when using translation pieces to guide neural machine translation. In particular, it is necessary to define the amount of retrieved target sentences for each source sentence, M, the weights given to the translation pieces, λ, and the similarity threshold for filtering the translation pieces, γ.

In this subsection, we are going to introduce incremental changes, checking along the way whether the proposed changes are indeed improving the proposed guided-NMT approach. The German to English (De-En) language pair is used to find the optimal number of retrieved sentences and the similarity threshold. We use a single language pair to find the hyperparameters M, and in our case γ, whereas the values of λ are going to be fine-tuned individually for each language pair. This follows Zhang et al. [32]'s approach. All the values shown in this subsection are for the development set.

Baseline vs Guided-NMT

The baseline model, trained according to the described in Section 6.2 is tested against two setups:

- *Guided-10* - As proposed in Zhang et al. [32], i.e., collecting all the translation pieces concerning the 10 most similar sentences with regard to a given input sentence, using $\lambda = 1$.
- *Guided-10-RT* - The same as the previous, but applying the *Reward until Translated (RT)* strategy. In other words, this means that we no longer reward translation pieces that are already present in the corresponding beam hypothesis (see the first point of Subsection 5.1.3).

The obtained results are shown in Table 6.9. It is possible to observe that naively following Zhang et al. [32]'s approach leads to a large degradation of the scores. Namely, we observe a catastrophic increase in $\mathrm{REP}_{\mathrm{Score}}$, and a length ratio much larger than the one reported for the baseline. It is reasonable to assume that constantly rewarding the translation pieces causes the model to output them more often than desired. By manually inspecting the obtained translations, it is indeed possible to verify that this is the case. For example, the presence of repetitions is very noticeable in the following translation output by the model *Guided-10*:

> *"the adhesive bandage pad module of any one of claims 2 to 23 , wherein the rotation of the at least one servo motor is electrically connected to the delivery assembly . the adhesive*

[7] Code available online at: `https://github.com/PedroMLF/guided-nmt`

bandage pad of any one of claims 2 to 23 , wherein the rotation of the at least one servo motor is electrically connected to the delivery assembly . the adhesive bandage pad of any one of claims 2 to 23 , wherein the rotation of the at least one servo motor is electrically connected to the delivery assembly .".

The corresponding reference is:

"the adhesive bandage pad module of any one of claims 2 to 23 , wherein rotation of said at least one servo motor is electrically tied to said infeed assembly .".

It is clear that a large portion of the sentence is being repeated. Looking at the corresponding translation pieces, it is possible to find the words *"adhesive"*, *"bandage"*, *"pad"*, *"module"*, *"of"*, *"any"*, *"one"*, *"of"*, *"the"*, *"claims"*, *"2"*, *"to"*, *"wherein"*, *"said"*, *"infe@@"*, *"assembly"*, *"is"*, *"servo"* and several *n*-gram combinations of them. Despite being indeed parts of the correct translation, rewarding them several times during the translation has lead to repetitions. The proposed modification to the model architecture leads to the sentence *"the adhesive bandage pad module of any one of claims 2 to 23 , wherein rotation of said at least one servo motor is electrically connected to said delivery assembly .".* In this case repetitions are no longer an issue and a sentence close to the reference is produced.

Furthermore, when compared with the baseline, it is shown that the *Reward until Translated* strategy has a positive impact in BLEU, METEOR and $\mathrm{DROP}_{\mathrm{Score}}$, at the expense of a slight increase in $\mathrm{REP}_{\mathrm{Score}}$. Therefore, this modification will be used from this point onwards.

Table 6.9: In-domain guided-NMT - Baseline vs guided-NMT.

	Model	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
De-En	*Baseline*	52.18	40.86	**5.84**	11.50	0.916
	Guided-10	40.29	39.71	49.73	17.61	1.372
	Guided-10-RT	**53.86**	**41.48**	6.15	**11.42**	0.927

Choosing the Amount of Retrieved Sentences

In the last experiment we established the need of not rewarding the same translation piece multiple times. The next step is to define the amount of retrieved sentences M, for each source input sentence. This parameter impacts not only the performance of the model, but also its practical usefulness. Retrieving more sentences slows down the translation process, by both increasing the time spent retrieving the translation pieces, and using them during decoding. This slowdown is also reported in Zhang et al. [32]. Even though an optimized implementation is outside the scope of this work, some choices will take performance into consideration.

Before reporting the values obtained by varying the parameter M, the distributions of the retrieved sentence similarities are shown in Table 6.10. An immediate observation is that the majority of retrieved sentences has a similarity (calculated as defined in Equation 5.1) between 0.2 and 0.5. The larger the value of M, the higher the absolute number of sentences with high similarity that are retrieved. This means that lower values of M might not be able to create translation pieces that leverage all the very similar sentences available for a given input sentence. Nonetheless, in relative values, those sentences become less prominent as more sentences are retrieved.

The results obtained when varying the value M are seen in Table 6.11. The overall conclusion to be taken is that the improvements obtained from using larger values of M are almost negligible. Given the reported results, the value of $M = 5$ is chosen, since it has similar performance when compared with the other values while keeping the processes of gathering translation pieces and decoding faster.

Table 6.10: In-domain guided-NMT - Distribution of similarities between source and retrieved sentence, for different M values.

De-En								
	M = 5		**M = 10**		**M = 20**		**M = 50**	
Similarity	Sent	Perc	Sent	Perc	Sent	Perc	Sent	Perc
[0.0, 0.1[	132	0.88	325	1.08	759	1.26	2428	1.62
[0.1, 0.2[	1374	9.16	3115	10.38	6901	11.50	19538	13.02
[0.2, 0.3[	4898	35.65	10869	36.23	23794	39.66	64562	43.04
[0.3, 0.4[	4237	28.25	8620	28.73	17152	28.59	41418	27.61
[0.4, 0.5[	1654	11.03	3019	10.06	5339	8.90	11321	7.55
[0.5, 0.6[	800	5.33	1372	4.57	2231	3.72	4334	2.89
[0.6, 0.7[	513	3.42	815	2.71	1229	2.05	2271	1.51
[0.7, 0.8[	389	2.59	573	1.91	867	1.44	1435	0.96
[0.8, 0.9[	389	2.59	547	1.82	787	1.31	1316	0.88
[0.9, 1.0[	475	3.17	582	1.94	734	1.22	1130	0.75
1	139	0.93	163	0.54	207	0.34	247	0.16

Table 6.11: In-domain guided-NMT - Choice of the amount of retrieved sentences. † indicates translations whose difference in BLEU, with regard to the model with the highest score, is not statistically significant ($p < 0.01$).

	Model	**BLEU ↑**	**METEOR ↑**	**REP ↓**	**DROP ↓**	**LR**
De-En	*Guided-05-RT*	**53.87**†	41.47	**6.10**	11.38	0.923
	Guided-10-RT	**53.86**†	**41.48**	6.15	11.42	0.927
	Guided-20-RT	**53.93**	**41.48**	6.14	**11.33**	0.931
	Guided-50-RT	**53.91**†	41.46	6.21	11.38	0.933

Filtering Translation Pieces with Sentence Similarity

As mentioned in Subsection 5.1.3, rewarding all translation pieces, even the ones from sentences with low-similarity scores, may not be optimal. Therefore, it is tested how filtering out translation pieces from sentences with a similarity below different threshold values γ, affects the results. The results with $\gamma = \{0.2, 0.3, 0.4, 0.6\}$ are shown in Table 6.12. The nomenclature of the models is defined as *Guided-M-SimilarityThreshold-RT*.

Looking at the reported values it is noticeable that both BLEU and METEOR slightly increase when translation pieces from sentences with similarity up to 0.4 are removed, up until a point where performance degrades. Filtering sentences with a similarity below 0.4 yields the best scores in the aforementioned metrics, and makes it possible to filter translation pieces from around 75% of the extra sentences. Therefore, the positive outcome from this addition is twofold: BLEU and METEOR slightly increase without affecting too much $\mathrm{REP}_{\mathrm{Score}}$ and $\mathrm{DROP}_{\mathrm{Score}}$; both creating translation pieces and decoding are less time consuming. Also, since filtering translation pieces based on lower similarity values shows benefits, it might be hypothesized that low-similarity sentences are harming the quality of the output translations.

Table 6.12: In-domain guided-NMT - Choice of similarity threshold. † indicates translations whose difference in BLEU, with regard to the model with the highest score, is not statistically significant ($p < 0.01$).

	Model	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
De-En	*Guided-05-0.0-RT*	53.87	41.47	6.10	**11.38**	0.923
	Guided-05-0.2-RT	53.88	41.47	**6.09**	**11.38**	0.920
	Guided-05-0.3-RT	53.95	41.53	6.18	**11.38**	0.924
	Guided-05-0.4-RT	**54.18**	**41.61**	6.33	11.40	0.923
	Guided-05-0.6-RT	53.77	41.43	6.38	11.44	0.922

Using λ vs λ_1 and λ_2

Finally, we test whether it is beneficial to split the λ that weights the translation pieces into two different λ values, one weighting the uni-grams and the other weighting the remaining *n*-grams (up to 4 in these experiments). Table 6.13 shows that none of the single values of λ is able to increase BLEU significantly without degrading all other metrics. That is also the case when λ_1 and λ_2 are used. Even $(\lambda_1, \lambda_2) = (1.5, 1.0)$, which yields an improvement of around 0.5 BLEU points, clearly introduces more repetitions. It is noticeable as well that the length ratio increased in that case. This probably means that the proposed λ values are creating longer sentences, reducing the value of the brevity penalty in BLEU calculation, even though those added words include repetitions, as indicated by $\mathrm{REP}_{\mathrm{Score}}$. It is plausible that a large enough decrease in the brevity penalty outweighs lower precision values during BLEU calculation. Taking into consideration the reported values, both λ values are going to be used during the hyperparameter tuning of λ, even though for this language pair a unique $\lambda = 1$ is the most advantageous choice.

Table 6.13: In-domain guided-NMT - Choice of the λ values. Bold values represent the results for the chosen set of weights.

	De-En						
	λ_1	λ_2	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
Shared λ	0.8		53.89	41.53	5.84	11.38	0.918
	1.0		**54.18**	**41.61**	**6.33**	**11.40**	0.923
	1.2		54.15	41.51	7.20	11.49	0.930
	1.5		54.33	41.40	8.69	11.65	0.953
Separated λ	1.0	1.1	53.99	41.57	6.11	11.42	0.921
	1.1	1.0	54.28	41.57	7.19	11.40	0.934
	1.0	1.2	53.93	41.52	6.02	11.39	0.920
	1.2	1.0	54.34	41.56	7.94	11.43	0.942
	1.0	1.5	53.65	41.37	5.92	11.69	0.917
	1.5	1.0	54.67	41.38	11.87	11.52	0.991

6.5.2 Evaluation and Analysis

Having used the development set to find the optimal architecture it is now possible to obtain results for the test set. The results use the proposed *Reward until Translated* strategy, 5 sentences retrieved per input sentence, and filtering translation pieces from sentences with a similarity lower than 0.4. Finally, the values of λ are tuned using the development set, by choosing the set of values that yield high BLEU and METEOR scores, and at the same do not degrade noticeably $\mathrm{REP}_{\mathrm{Score}}$ or $\mathrm{DROP}_{\mathrm{Score}}$.

Test Set Results

The first step is to verify whether the reported gains for the development set also exist in the test set. Those results are shown in Table 6.14. Regarding the values of λ, for German to English we use $(\lambda_1, \lambda_2) = (1.0, 1.0)$ and for Spanish to English, the choice is $(\lambda_1, \lambda_2) = (1.1, 1.0)$.

It is possible to see that in terms of BLEU and METEOR, the guided-NMT approach always outperforms the baseline, adding around 2 BLEU points in both cases. The reported $\mathrm{DROP}_{\mathrm{Score}}$ is virtually the same in both cases and it is noticeable the increase in $\mathrm{REP}_{\mathrm{Score}}$. This increment is larger for the Spanish to English translations, and a similar increase was detected while tuning the λ values in the development set. It is possible to speculate that the $\mathrm{REP}_{\mathrm{Score}}$ reported for the baseline model is so low that, rewarding certain subwords during decoding, even when employing strategies to mitigate repetitions due to the translation pieces, slightly affects the translation in this regard. Nonetheless, the resulting value is still fairly low when compared, for instance, with the value reported for the German to English language pair. Also, the benefit across the other metrics outweighs the increase in $\mathrm{REP}_{\mathrm{Score}}$.

Table 6.14: In-domain guided-NMT - Results for the in-domain test set.

	Model	**BLEU** $\uparrow$	**METEOR** $\uparrow$	**REP** $\downarrow$	**DROP** $\downarrow$	**LR**
De-En	*Base*	52.12	40.89	**5.65**	11.45	0.911
	Guided	**54.01**	**41.50**	5.96	**11.41**	0.919
Es-En	*Base*	52.44	40.64	**2.38**	6.54	0.939
	Guided	**54.70**	**41.31**	3.77	**6.52**	0.955

Infrequent n-grams

One of the goals mentioned for the proposed approach is to be able to translate *n*-grams that appear in a test corpus, but that seldom occur in the data used to train the NMT model. In order to evaluate if the obtained translation is improving the rate of correct translations of infrequent *n*-grams, we follow an approach similar to the proposed in Zhang et al. [32].

The first step to calculate the amount of correctly translated *n*-grams, with $1 \leq n \leq 4$, is to obtain the intersection between the set of *n*-grams present in a candidate translation and the reference, at a corpus level. This yields the *n*-grams that are both present in the output translation and in the reference, i.e., the correctly translated *n*-grams. Then, it is possible to find how many times each of those *n*-grams appeared in the training corpus. This value is referred to as ψ. With this information, it is possible to count the number of instances of a given *n*-gram, both in the reference and candidate translations, knowing that it appeared ψ times in the training data. Finally, comparing the counts between the baseline NMT model and the guided-NMT approach, we can find if adding translation pieces indeed helps translating *n*-grams that seldom appear in the training corpus.

Table 6.15 shows the *n*-grams counts for the baseline NMT model, the guided-NMT model and the ratio between both, for *n*-grams that appear ψ times in the training corpus. Even though the ratios are not much higher than one, the guided approach is always able to surpass the baseline. It is also observable that the ratios are larger for *n*-grams that are less frequent in the training corpus. This was to be expected

due to the way NMT models are trained, which makes it more difficult to output words/patterns that are not seen very often during training, as opposed to more common ones.

Table 6.15: In-domain guided-NMT - Counts of correctly translated *n*-grams that appear ψ times in the training corpus.

	ψ	**1**	**2**	**5**	**10**	**20**	**50**	**100**
	Base	7061	4249	2169	1092	586	282	143
De-En	*Guided*	7895	4823	2353	1184	632	301	151
	Guided/Base	1.12	1.14	1.08	1.08	1.08	1.07	1.06
	Base	3933	2510	1450	843	470	214	103
Es-En	*Guided*	4542	2835	1650	932	502	228	104
	Guided/Base	1.15	1.13	1.14	1.11	1.07	1.07	1.01

Translation Pieces Statistics

During the tuning of the hyperparameters it was noticed the importance of using the right set of λ weights. This means that properly balancing the weights given to the similarities of the translation pieces is one of the most critical aspects when using guided-NMT. A possible issue may arise when the translation pieces collected for the development set are not representative of the ones collected for the test set. If that happens, it is possible that the λ values may be, either too high, impacting negatively the output, or too low, preventing the model from fully leveraging the translation pieces. To dismiss such a possibility in the reported set of results, Table 6.16 presents some statistics regarding the collected translation pieces.

It is possible to observe that indeed, for both language pairs, the set of translation pieces collected for the development set and the test set are fairly similar. This means that the reported scores for the test set take full advantage of the translation pieces. Another interesting aspect are the slightly higher similarities reported for the Spanish to English language pair. This might explain why the improvement in BLEU is a couple of decimal points higher for that language pair.

Table 6.16: In-domain guided-NMT - Statistics of the collected translation pieces for $M = 5$ and $\gamma = 0.4$.

	De-En		**Es-En**	
Value	*Dev*	*Test*	*Dev*	*Test*
Total	116930	112412	82517	86469
TPs per Sentence	38	37	33	34
Average Similarity	0.704	0.703	0.859	0.859
Similarity σ	0.207	0.206	0.186	0.184
Percentile 25 Simi.	0.500	0.500	0.758	0.757
Percentile 50 Simi.	0.696	0.693	0.961	0.959
Percentile 75 Simi.	0.913	0.915	1.000	1.000

Sentence Examples

To further illustrate how the use of translation pieces is able to output translations closer to the reference, four examples are shown in Table 6.17. These examples focus on specific adequacy issues, that the use of translation pieces helps overcoming. In particular, Examples 1 and 2 focus on the issue of dropping words, whereas the last two examples focus on errors that change the meaning of the sentence.

In the first example the word *"vildagliptin"* is not output by the baseline model. This word is critical in the provided context, since it is the class of drugs for which there is not enough human data. Dropping this word would be a major error in the translation of what appears to be part of a medical leaflet. In the second sentence a whole portion of the reference is not output by the baseline model. The dropped portion of the sentence refers to what seems to be a safety procedure, and therefore it is clearly necessary in the final translation. Guided-NMT is able to produce the correct translation in both cases.

The last two examples show how a different choice of words during translation might change the meaning of the sentence. Looking at the third example's reference sentence, it is mentioned that the symptoms in a given patient may be severe, whereas the baseline output sentence mentions that the patient itself is serious, which does not make sense. In the last example, the medical leaflet is guiding the reader to a given section in case he is considering stopping a particular medication. This is not the idea the baseline translation is conveying by using the expression *"should stop taking"*. Once again, guided-NMT is able to avoid the described issues.

Even though the provided examples seem like minor modifications to sentences, they are critical mistakes in terms of translations that would clearly be noticed by someone manually inspecting the quality of the outputs. In other words, these seemingly small changes could be the difference between having a user satisfied with a given MT service, or a user that stops using it. Fixing several of these mistakes throughout a *corpus* of sentences ends up leading to large improvement in terms of quality of the translations.

6.6 Guided Neural Machine Translation for Domain Adaptation

As discussed in Section 5.2, it is reasonable to assume that the proposed guided-NMT approach (see Chapter 5) performs well for the task of domain adaptation. Thus, throughout this section we are going to present a set of experiments and results to verify whether our assumptions hold. The staple idea of this section is that using a model trained with generic data should be able to improve its performance in an in-domain test set by leveraging in-domain translation pieces. Unless otherwise stated, guided-NMT is able to access the full in-domain training set to create the translation pieces, as in Section 6.5.

6.6.1 Hyperparameter Tuning

When using guided-NMT to perform domain adaptation, the problem is fairly different from the one discussed in the previous subsection. Therefore, it is expected that the optimal set of hyperparameters differs from the one found in Section 6.5. In this case, since the generic model is trained on data that has no instances of medical data, it might be worth using more retrieved sentences per input sentence, and having a lower threshold for sentence similarity. The hyperparameters are tuned with regard to the German to English language pair, using the in-domain development set.

Table 6.17: In-domain guided-NMT - Examples. The golden color indicates the portion of the reference that the example targets, the **red color** indicates the mistranslations and finally, the *blue color* highlights correct translations of the parts of the sentence being focused. The <**MISS**> symbol refers to portions of the sentence that were missed.

	Example 1
Reference	Due to both the potential risk of neonate hypoglycaemia related to metformin and the lack of human data with vildagliptin, icandra should not be used during lactation (see section 4.3).
Base	Because of the potential risk for hypoglycaemia associated with metformin and due to lack of data in humans <**MISS**>, icandra should not be used in breast-feeding women (see section 4.3).
Guided	Due to the potential risk of hypoglycaemia related to metformin and no human data with *vildagliptin*, icandra should not be used in women who are breast-feeding (see section 4.3).
	Example 2
Reference	The patient should be instructed to discard the pen safely, with the needle still attached, after each injection.
Base	The patient should be instructed to dispose of the pen safely <**MISS**> after each injection.
Guided	The patient must be instructed to dispose of the pen after each injection, *with the needle remaining on the pen* after each injection.
	Example 3
Reference	Generally these symptoms are mild to moderate, however, in some patients they may be severe in intensity.
Base	Symptoms are usually mild to moderate, **but some patients may be serious**.
Guided	Symptoms are usually mild to moderate, however, *in some patients they may be severe*.
	Example 4
Reference	- You are considering stopping cymbalta (see section 3).
Base	- You **should stop taking** cymbalta (see section 3).
Guided	- You are *considering stopping* cymbalta (see section 3).

Baseline vs Guided-NMT

The first step is to verify how translating with the generic model (*Base*), compares with using translation pieces (*Guided*), and further extending the target vocabulary by adding the subwords that are not part of it (*Guided-Extended*). The initial setup for the *Guided* approach is to retrieve the ten most similar sentences, $M = 10$, and to have no lower bound on the sentence similarity, $\gamma = 0.0$. The subwords added to the vocabulary during decoding time have their corresponding weight and bias layer's values defined with the same values learned for the unknown symbol. The weight given to the translation pieces is 1.0 if they are already present in the vocabulary, and 5.0 if they were later added to the vocabulary (*for the Guided-Extended* approach). This value of 5.0 may be tuned afterwards. The reason behind the larger weight given to the added words is that they are defined with the same parameters as the unknown symbol, which usually yields low log probabilities. To make them the most probable subword during beam search it is necessary that the reward given to them outweighs the low score given to the unknown symbol. Nonetheless, even with the higher reward, the translation will only be affected when the model is somewhat uncertain of which subword to output and the probabilities are evenly spread along the output layer (and therefore no subword has a high log probability). It is worth noting that the maximum reward given is the weight itself, since it multiplies the similarity between the source and the

retrieved sentence for the respective translation pieces, whose value is 1 at most.

Looking at the results in Table 6.18 it is possible to observe that, as expected, the baseline generic model performs much worse than the in-domain model (from Section 6.5) when translating the in-domain development set. This degradation in performance may be attributed to the fact that the model is now dealing with a particular domain of language that was not seen during the training procedure. Even though the language model has been trained for the same language pair, the model is not capable of generating subwords for such instances with the confidence and quality of a model trained with the actual medical in-domain data.

Adding the translation pieces leads to a significant increase, of approximately 2.5 BLEU points, while showing improvements across all the other reported metrics. Rewarding the translation pieces is even more important in the context of domain adaptation, since critical in-domain terms have not been seen during the training of the model. It is also shown that, extending the target vocabulary with the translation pieces' subwords that were not part of it, leads to no significant improvements. The importance of this addition depends on several factors, such as the uncertainty of the model when generating the probability over the target vocabulary (the more spread the probability is, the higher the chance that added subwords are useful), and the number of out of vocabulary subwords. In this case the number of added subwords is only 180 which is around 0.6% of the vocabulary size. Therefore, it is reasonable to assume that the number of additions is not significant enough to provide a noticeable improvement, for this specific case. Nonetheless, it makes sense to keep this addition to the model, as it does not add much computational complexity and it has the potential of being beneficial.

Table 6.18: Guided-NMT for domain adaptation - Baseline vs guided experiments. † indicates translations whose difference in BLEU, with regard to the model with the highest score, is not statistically significant ($p < 0.01$).

	Model	**BLEU** ↑	**METEOR** ↑	**REP** ↓	**DROP** ↓	**LR**
	Base	16.27	20.55	11.26	25.91	0.892
De-En	*Guided-10*	**18.75**†	21.70	10.07	25.61	0.870
	Guided-Extended-10	**18.76**	**21.71**	**10.06**	**25.58**	0.870

Choosing the Amount of Retrieved Sentences

Having the general approach defined, the next step is to choose the number M of retrieved sentences for each input sentence. In this section's approach, a generic model is being used to translate an in-domain development set. Since the translation pieces refer to in-domain data, it is reasonable to assume that, the higher the number of retrieved sentences for each input sentence, the larger the improvements reported. This is indeed verified, as it may be seen in Table 6.19. Taking in consideration that retrieving more sentences slows down the process, it is chosen to use $M = 20$, since it yields very similar results to an higher value of M, while not increasing tremendously the complexity.

Table 6.19: Guided-NMT for domain adaptation - Choice of the amount of retrieved sentences. † indicates translations whose difference in BLEU, with regard to the model with the highest score, is not statistically significant ($p < 0.01$).

	Model	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
De-En	*Guided-Extended-05*	18.61	21.62	10.21	25.63	0.870
	Guided-Extended-10	18.76	21.71	10.06	25.58	0.870
	Guided-Extended-20	**18.92**†	21.75	**9.86**	**25.53**	0.865
	Guided-Extended-50	**19.04**	**21.84**	9.96	**25.53**	0.867

Filtering Translation Pieces with Sentence Similarity

For the problem of guided domain adaptation, it is reasonable to think that higher similarity thresholds will progressively decrease the quality of the translations. To test this assumption, we report the values obtained by using a similarity threshold of 0.2, 0.3, and 0.4 in Table 6.20. It is possible to observe that the impact is almost negligible for $\gamma = 0.2$ and $\gamma = 0.3$, being slightly more noticeable for $\gamma = 0.4$. The gap between the results reported using thresholds of 0.3 and 0.4 should be more noticeable with higher values of λ. The final choice is then to use $\gamma = 0.3$, due to the impact across all metrics being minimal while reducing the complexity of the proposed approach.

One further noticeable pattern is that, when a generic model is used to translate the in-domain development set, increasing the similarity threshold seems to always have a negative impact in BLEU scores. Thus, it is likely that for this particular use case, even sentences with low-similarity are able to help the model improve its performance.

Table 6.20: Guided-NMT for domain adaptation - Choice of sentence similarity threshold. † indicates translations whose difference in BLEU, with regard to the model with the highest score, is not statistically significant ($p < 0.01$).

	Model	BLEU ↑	METEOR ↑	REP ↓	DROP ↓	LR
De-En	*Guided-Extended-20-0.0*	**18.92**	**21.75**	9.86	25.53	0.865
	Guided-Extended-20-0.2	**18.90**†	21.73	**9.81**	25.54	0.865
	Guided-Extended-20-0.3	**18.87**†	21.74	9.97	**25.51**	0.870
	Guided-Extended-20-0.4	18.62	21.56	10.47	25.68	0.874

Defining the impact of λ and τ values

The only step missing is to tune the weights that multiply the translation piece's values. In the case of domain adaptation it is also introduced a weight τ. This weight multiplies the value of the translation pieces added with the *extended* step. As mentioned in Section 5.2, the value of τ should be higher than the value of λ, since it is used to increase the reward given to subwords whose weight and bias vectors were defined with the same values as the ones learned for the unknown symbol during training.

Table 6.21 shows the results obtained by changing the values of λ and τ. Even though some other values of τ were tested, none improved the results nor significantly made any of the metrics particularly worse (much due to the small portion of the vocabulary that this represents, and the infrequent nature of the words that are added this way). As expected, since the development set is comprised of in-domain data and the base model is trained using generic data, the λ values in this experiment are

much higher than those reported for the experiments of section 6.5.1. It is also noticed that the optimal value of λ is around 4.0. Splitting λ into λ_1 and λ_2, that weight uni-grams and the remaining n-grams, respectively, and lowering the value that multiplies the translation pieces' uni-grams yields a similar value in BLEU and METEOR while lowering considerably the $\mathrm{REP}_{\mathrm{Score}}$. This is somewhat expected, since rewarding subwords whose context is already present, reduces the risk of rewarding *n*-grams that negatively influence the final translation. This hypothesis is backed up by the experiment in which λ_1 is set to higher, and then lower values than λ_2. Looking at the results, and taking into consideration the aforementioned conclusions, the final values for the German to English language pair are $(\lambda_1, \lambda_2, \tau) = (3.5, 4.0, 5.0)$. These values add around 3.5 BLEU points to the set of weights used until now, while also showing improvements across all other metrics.

One interesting aspect discovered during the tuning of the weight parameters is that the inherent necessity of higher weight values may affect the performance by giving too much relevance to *stopwords*. Since *stopwords* are common in any corpus, the generic model should be able to generate high enough log probabilities for those words. This is not true for the remainder translation pieces, that should regard in-domain data. Nonetheless, slightly rewarding the *stopwords* present in the translation pieces is important, as they may serve as context to in-domain subwords that may be part of the same translation piece. The discussed problem is not an issue for the approach followed in Section 6.5, since the values of λ are considerably smaller in that scenario. To overcome this situation it would be possible to, during decoding time, give a weight of 1.0 to the translation pieces' subwords that happen to be *stopwords*. Such a naive approach should be avoided, since it would have to be repeated at each decoding step, of each batch, for several different *stopwords*. The proposed alternative as a reasonable approach, is to use a fixed λ of 1.0 as the weight used in the reward given to the 45 most frequent subwords of the target vocabulary. These elements include the most common *stopwords* and therefore, help to address the aforementioned issue while introducing negligible overhead to the decoding. The value of 45 was obtained by both trying different reasonable values in the development set, and making sure the set of words was comprised mostly of *stopwords*. For the chosen set of parameters, $(\lambda_1, \lambda_2, \tau) = (3.5, 4.0, 5.0)$, not using such a strategy yields a translation that obtains a BLEU of 21.69, a METEOR of 22.87, a $\mathrm{REP}_{\mathrm{Score}}$ of 9.59 and a $\mathrm{DROP}_{\mathrm{Score}}$ of 26.76, thus, worse at every metric.

6.6.2 Evaluation and Analysis

In the previous subsections we found the optimal set of hyperparameters to use guided-NMT as a way of performing domain adaptation. The number of retrieved sentences for each input sentence is 20, and translation pieces coming from sentences with a similarity lower than 0.3 are filtered. Regarding the discussed weights, for the German to English experiments we use $(\lambda_1, \lambda_2, \tau) = (3.5, 4.0, 5.0)$, and for Spanish to English we use $(\lambda_1, \lambda_2, \tau) = (2.5, 4.5, 5.0)$. In this subsection, the results obtained with the proposed model are reported and analyzed. More concretely, the results for the in-domain test set are going to be presented, as well as values for the number of correct translations of infrequent *n*-grams. Finally, after presenting some examples of sentences where using translation pieces proved to be fruitful,

Table 6.21: Guided-NMT for domain adaptation - Choice of λ and τ values. Bold values represent the results for the chosen set of weights.

				De-En			
λ_1	λ_1	τ	**BLEU** ↑	**METEOR** ↑	**REP** ↓	**DROP** ↓	**LR**
				Shared λ			
1.0		5.0	18.87	21.74	9.97	25.51	0.870
2.0		5.0	20.72	22.70	9.10	25.27	0.843
3.0		5.0	21.84	23.36	9.72	25.31	0.851
3.5		5.0	22.01	23.46	9.85	25.45	0.851
4.0		5.0	22.35	23.62	10.76	25.87	0.858
5.0		5.0	22.33	23.66	12.03	26.26	0.883
6.0		5.0	21.87	23.41	13.85	27.21	0.921
				Separated λ			
3.5	4.0	5.0	**22.25**	**23.59**	**9.52**	**25.38**	0.848
4.5	4.0	5.0	22.27	23.59	11.49	25.81	0.865

a more realistic application of the proposed method will be tested. Namely, a smaller amount of extra data is going to be used incrementally, just as it would happen in a real context where data is being gathered from several different streams.

Test Set Results

The first step for evaluating the model is to present the results obtained for the in-domain test set (Table 6.22). Similarly to what was reported for the development set, using translation pieces as a way of doing domain adaptation also yields improvements in the test set. Namely, the results for German to English improve around 6 BLEU points, and for Spanish to English, the improvement is slightly more than 10 BLEU points. Also worth mentioning is that all other metrics show improvements, barring the $\mathrm{REP}_{\mathrm{Score}}$ in the Es-En experiments, where a very slight increase is reported. These results highlight the capacity of the translation pieces for domain adaptation, improving the quality of the translation of an in-domain test set across several metrics.

Also noticeable, are the larger improvements reported for the Spanish to English language pair. It may be hypothesized that this difference is related with the similarity distribution of the collected translation pieces. Further analysis on this is going to be presented in a later subsubsection, where some statistics regarding translation pieces are reported.

Table 6.22: Guided-NMT for domain adaptation - Results for the in-domain test set.

	Model	**BLEU** ↑	**METEOR** ↑	**REP** ↓	**DROP** ↓	**LR**
De-En	*Base*	16.09	20.62	11.41	25.10	0.897
	Guided	**22.21**	**23.63**	**9.68**	**24.46**	0.857
Es-En	*Base*	22.03	25.86	**4.81**	12.63	1.000
	Guided	**32.20**	**29.95**	4.95	**12.34**	0.989

Infrequent n-grams

As previously stated, the approach being currently analyzed uses generic data as training corpus, as opposed to the extra corpus and the test set, that correspond to in-domain data. When drawing com-

parisons between the obtained values and the ones reported in Table 6.15, two outcomes are expected: first, the absolute counts of *n*-grams should be lower, since the generic model will have trouble producing in-domain words; secondly, the ratios between the *guided* and *base* approach should be higher, as translation pieces are even more critical in this case.

Looking at Table 6.23 it is indeed possible to observe that for lower values of ψ (number of times a *n*-gram occurs in the training set), the guided-NMT approach is more capable of producing the correct *n*-gram than the baseline. The overall lower absolute values, when compared with the experiments that use a model trained on in-domain data, are also noticeable. Finally, using translation pieces is shown to be able to induce the production of *n*-grams that very rarely appear in the training corpus, close to more than twice as often as the generic NMT model. Therefore, we may conclude that the impact of translation pieces in the production of rare *n*-grams is particularly noticed when using a generic model to translate an in-domain test set.

Table 6.23: Guided-NMT for domain adaptation - Counts of correctly translated *n*-grams that appear ψ times in the training corpus.

	ψ	**1**	**2**	**5**	**10**	**20**	**50**	**100**
De-En	*Base*	1011	599	412	257	145	88	59
	Guided	1986	1254	682	379	199	123	78
	Guided/Base	1.96	2.09	1.66	1.47	1.37	1.40	1.32
Es-En	*Base*	1262	778	459	264	141	84	73
	Guided	2249	1378	871	466	241	134	85
	Guided/Base	1.78	1.77	1.90	1.77	1.71	1.60	1.16

Translation Pieces Statistics

As previously mentioned during this work, it is important that the translation pieces collected for the development set are representative of the later collected for the test set. Due to the high values used as weights in the context of domain adaptation, it is even more critical to verify whether this is in fact happening. Looking at the values reported in Table 6.24, it is possible to conclude that there are no significant differences between the development and test set values, for each of the language pairs. Therefore, the set of chosen weights fully leverages the extra in-domain data.

When the results for the in-domain test set were presented, there was a noticeable difference improvement-wise. Namely, the improvements for the Spanish to English language pair where higher than the ones reported for the German to English experiments. Looking at the reported statistics, there is a striking difference between the similarity of the collected translation pieces for each of the language pairs. So, one conclusion may be drawn: even when the set of weights is correctly tuned, higher similarities between the input and the retrieved sentences will lead to larger improvements.

Sentence Examples

Translating an in-domain test set using a generic model, instead of a model trained with in-domain data, will lead to very different outputs. Therefore, using translation pieces as a way of doing domain

Table 6.24: Guided-NMT for domain adaptation - Statistics of the collected translation pieces for $M = 20$ and $\gamma = 0.3$.

	De-En		Es-En	
Value	*Dev*	*Test*	*Dev*	*Test*
Total	619684	598533	215210	215949
TPs per Sentence	206	199	86	86
Average Similarity	0.455	0.457	0.632	0.642
Similarity σ	0.189	0.189	0.280	0.280
Percentile 25 Simi.	0.328	0.328	0.352	0.355
Percentile 50 Simi.	0.374	0.378	0.550	0.587
Percentile 75 Simi.	0.488	0.494	0.962	0.970

adaptation has the potential of introducing bigger changes in the obtained translations. The chosen examples (Table 6.25) convey that idea, showing clearly that the translation pieces are either able to guide the translation to be closer to the reference, or to output in-domain words or expressions that the baseline candidate translation completely misses.

The first example highlights a case in which the baseline translation completely misses its purpose, whereas the guided translation is able to perfectly convey the same idea as the reference. The second example shows more subtle yet interesting changes. More specifically, the guided translation is not only able to avoid dropping the word *"reconstituted"* but also able to correctly produce the word *"patient"* instead of repeating *"body"*. The last two examples are particularly interesting since guided translations are able to correctly translate terms which are clearly in-domain, such as *"2-adrenoreceptor"*, or at least able to produce a word which is very close to the correct term, such as *"glucocortiid"* instead of *"glucocorticoid"*, and *"bondronant"* instead of *"bondronat"*. The latter cases would be a very inexpensive errors to correct by someone editing these translations.

Impact of the Number of Extra Sentences

Even though the previous set of experiments is helpful and shows how translation pieces may serve as a possible approach to the problem of domain adaptation, they rely on large sets of extra sentences. When such a big amount of in-domain data is available, it is much more effective to train a model from scratch. This is easily verified when looking at the results from Section 6.5. In particular, for the German to English language pair, a BLEU of 52.12 is obtained for the in-domain test set, as opposed to the value of 22.21 when using a model trained with generic data and then extended with translation pieces.

A more realistic setting is one in which a NMT model is trained using a large generic corpus and then, as in-domain data is collected, it is fine-tuned to take advantage of that data. Since fine-tuning requires re-training the model, which may be a cumbersome task, having the chance of leveraging collected in-domain data using translation pieces, poses as a computationally appealing alternative. Namely, coupling guided-NMT with periodic re-training of the model may conjugate the best of both approaches.

In order to test this setup the following three scenarios are going to be tested:

- *Translation Pieces*, where the baseline generic model uses translation pieces created from an extra corpus of in-domain sentences.

Table 6.25: Guided-NMT for domain adaptation - Examples. The golden color indicates the portion of the reference that the example targets, the **red color** indicates the mistranslations and finally, the *blue color* highlights correct translations of the parts of the sentence being focused. The <**MISS**> symbol refers to portions of the sentence that were missed.

Example 1	
Reference	Do not let the tablet dissolve in your mouth.
Base	Do **you respect the fact that the taboo is not already** in the mouth.
Guided	Do *not let the tablet dissolve* in the mouth.
Example 2	
Reference	The required volume of the reconstituted solution depends on the patient's body weight.
Base	The volume of the solution required is dependent on the **body**'s body weight.
Guided	The volume of the *reconstituted* solution depends on the *patient*'s body weight.
Example 3	
Reference	Like all medicines, bondronat can cause side effects, although not everybody gets them.
Base	Like all medicinal products, there can be **bondrot** side effects, but they must **not be allowed to occur** on everyone.
Guided	Like all medicines, **bondronant** may cause side effects, although *not everybody gets* them.
Example 4	
Reference	Formulation for comprising a glucocorticoid and a beta 2-adrenoreceptor agonist.
Base	*Preparation* for **content is a glucose code** and a beta **2 adrene prime minister**.
Guided	**Preparing for inhalation** *comprising a* **glucocortiid** and a beta *2-adrenoreceptor agonist*.

- *Fine-tuning*, where the generic model is fine-tuned using a given amount of in-domain sentences.
- *Latest Fine-tuned + Translation Pieces*, where a model fine-tuned in the previous number of available in-domain sentences has access to translation pieces obtained from the currently available in-domain sentences. This simulates a more real context, where someone fine-tunes a model, later has access to more in-domain data, and wants to avoid re-training the model straightaway.

The experiments are conducted for the German to English language pair, using as baseline the corresponding generic model, that is evaluated on the hitherto used in-domain test set. The considered intervals of extra in-domain sentences are 5000, 10000, 15000 and 20000, which are simply the top sentences of the available training in-domain corpus. So, for the value of 15000 sentences, for example, *Translation Pieces* would use the baseline model with access to translation pieces based on 15000 in-domain sentences, *Fine-tuning* would use the baseline model fine-tuned on those 15000 sentences, and finally, *Latest Fine-tuned + Translation Pieces* would use the baseline model fine-tuned on the previously available 10000 sentences while also taking advantage of translation pieces created from the full 15000 in-domain sentences. The choice of interval between sentences simulates what a given company could collect every month in terms of sentences for a particular domain.

The obtained results are presented in Figure 6.2. A first observation to be made is that any of the approaches improves the obtained BLEU score, even when using a small amount of extra 5000 in-domain sentences. This is somewhat expected since we are combining a generic model, that should offer a solid baseline, with strategies that leverage in-domain data.

Previously, it has been observed that translating an in-domain test set with a model trained with in-

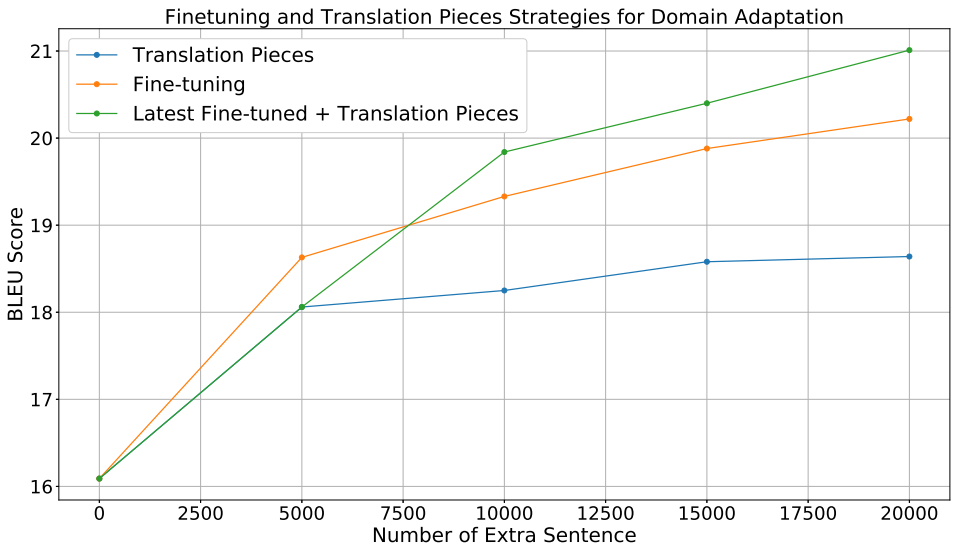


Figure 6.2: Guided-NMT for domain adaptation experiments.

domain data yields better results than using the generic model. Therefore it is expected that re-training the model, even if using a small amount of extra in-domain data, further improves BLEU scores. This is clearly observable, as the *Fine-tuning* results are always higher than the *Translation Pieces* ones. Also not surprising, due to the aforementioned reasoning, is that using only translation pieces performs always worse than the other two approaches.

The results of using fine-tuned models together with translation pieces are also particularly interesting. Namely, as long as we have fine-tuned models (corresponding to the last three measurements of the green line), this approach always outperforms simply using the fine-tuned model. This means that, not only the translations pieces are beneficial when using fine-tuned models, but also that the extra 5000 in-domain sentences have a larger impact when used as translation pieces (paired with a fine-tuned model) than when used to further fine-tune the generic model. This may be due to the fact that such a small amount of extra sentences might not be enough to push the model parameters that closer to the optimal for the in-domain scenario, when compared with the previously fine-tuned model, while being enough for the translation pieces algorithm to find much similar sentences and therefore, much more reliable *n*-grams to reward during decoding. The observed effect should be diluted over time, as the fine-tuned model learns to give the correct probability to the necessary subwords. Also, training a model for that specific domain is something to be considered after enough in-domain data is gathered.

Finally, it is also interesting to check the progression of some statistics related to translation pieces, as the amount of in-domain data increases. Table 6.26, shows both the number of translation pieces, and the corresponding average similarity, growing steadily with the amount of collected data. This adds further evidence to the hypothesis that, the more in-domain data is available, the better the generated translation pieces are and thus, the higher the positive impact in the produced translations. Continuously adding more data would eventually lead to the values reported for the German to English language pair in Table 6.24.

Table 6.26: Guided-NMT for domain adaptation - Statistics of the collected translation pieces created for the partial data.

	Total	TPs/Sentence	Average Simi.	Simi. σ
5k	304799	101	0.360	0.069
10k	335274	111	0.366	0.080
15k	353212	117	0.370	0.087
20k	363204	121	0.373	0.093

6.7 Summary

Throughout this chapter we presented the results from three sets of experiments, each of them with a different objective. The first set of experiments concerns the proposed fertility-based NMT (see Chapter 4). The goal of this model is to reduce the issues of over and under-translation in neural machine translation by making use of the concept of fertility and different attention mechanisms, that produce sparse probability distributions. Using two relatively small datasets, we obtained results for two language pairs: German to English; and Romanian to English. The proposed model was shown to be able to effectively reduce coverage-related adequacy issues, measure by $\mathrm{REP}_{\mathrm{Score}}$ and $\mathrm{DROP}_{\mathrm{Score}}$, while also improving or obtaining strong values for BLEU and METEOR.

The second set of experiments tests whether the guided-NMT approach (Chapter 5) is indeed able to improve the quality of translations in an in-domain setup, namely, by increasing the number of correctly translated *n*-grams that are rare in the training corpus. For that purpose we used the UFAL medical corpus (described in Section 6.1) to create datasets for two language pairs, German to English, and Spanish to English. The obtained results were conclusive, showing that using translation pieces to guide the translation is indeed able to produce *n*-grams that seldom appear in the training corpus more often than the baseline model. Furthermore, using guided-NMT lead to an increase in BLEU scores of around two points for both language pairs, while also performing well in the other reported metrics. During these experiments we also validated some of the introduced changes to the original model (Zhang et al. [32]). In particular, it was shown that introducing a strategy to avoid over-rewarding subwords, filtering the translation pieces from low-similarity sentences, and using a different weight to reward uni-grams or *n*-grams whose context is present in the current translation, have a positive impact in the obtained results.

The final set of experiments uses guided-NMT in the context of domain adaptation, namely, by using a generic NMT model, trained in a large and generic corpus, to translate the same medical in-domain test sets. Once again, guided-NMT was able to clearly improve the results over the baseline, both in terms of the reported metrics, and in the ratio of translations of infrequent *n*-grams. Furthermore, it was shown that translation pieces may be helpful in terms of domain adaptation, in a situation closer to what is found in the industry. More concretely, the results showed that translation pieces are able to improve the performance of the generic model, even after fine-tuning it with some small amount of in-domain data. Since guided-NMT is not as computationally expensive as re-training the model, it could be used as a way of leveraging the in-domain data as it is collected. This shows to be particularly helpful when there is a low amount of in-domain data available.

Chapter 7

Conclusions

In this chapter we start by making a summary of the contributions of this dissertation. Afterwards, we suggest some of the possible directions for future work regarding the proposed methods.

7.1 Summary of Contributions

In Section 1.1 we raised three open questions regarding the problem of adequacy in neural machine translation. Our contributions in this work aim to answer those same questions. Namely, three major sources of poor adequacy were identified: over-translations, under-translations, and mistranslations.

We started by introducing $\mathrm{REP}_{\mathrm{Score}}$ and $\mathrm{DROP}_{\mathrm{Score}}$ as a way of measuring how problematic over and under-translations are in the produced translations. Furthermore, these metrics were shown to correlate well with the human judgment regarding those same identified problems.

Then, we proposed a fertility-based approach to NMT, which leverages both fertility, and sparse and constrained attention transformations, such as the constrained softmax and the constrained sparsemax. Using fertility-based NMT, it was possible to improve the obtained translation with regard to, not only the commonly used MT metrics, such as BLEU and METEOR, but also also with regard to the two proposed metrics, $\mathrm{REP}_{\mathrm{Score}}$ and $\mathrm{DROP}_{\mathrm{Score}}$.

In order to improve the performance of NMT with regard to mistranslations, we followed Zhang et al. [32], and further enhanced it by addressing some of the identified shortcomings of the original work. To evaluate its performance, first we used an in-domain setup, where it was able to yield improvements in terms of the overall MT metrics, as well as capable of translating more often *n*-grams that are rare in the training corpus. Then, we used it as a way of performing domain adaptation, something that had not been done in the past. In this case it was possible to obtain an even larger increase in terms of performance, with regard to the baseline. Furthermore, we have shown that a common domain adaptation technique, fine-tuning, is able to leverage translation pieces, implying an interesting synergy between both approaches. The mentioned conclusions are part of a comprehensive evaluation and error analysis we carried for all the proposed metrics on three language pairs, with overall encouraging results.

7.2 Future Work

7.2.1 Machine Translation Metrics

In this work we introduce metrics that target two symptoms of adequacy issues, namely, over-translations and under-translations. One interesting direction to be explored in future metrics with the same goal, would be to verify which words are being dropped during the translation. Namely, it could be useful to use extra information about them, such as the part of speech, or whether they correspond to a named entity. Assuming this kind of features may be extracted, it would be possible to highlight cases in which more impactful words are being dropped.

7.2.2 Fertility-Based NMT

In theory, sparse and constrained transformations may be applied to any problem involving structured prediction. In this work we defined the upper bounds of the constrained transformation as the fertility of each source word. Nonetheless, it would have been possible to let the upper bounds vector be learned together with the other parameters of the network. This leads to an interesting line of thought: if those learned upper bounds were close to the actual fertilities found by our fertility predictor module, then, one could wonder if those upper bounds are learning interesting features about the data. Namely, devising experiments where these parameters are learned in other problems could be used to gain fruitful insights about structure in language, or other types of data.

7.2.3 Guided-NMT

Recently, there has been a growing interest in the possibility of incorporating cache or memory-like methods in NMT, mainly to allow better translations at a document-level (Sukhbaatar et al. [87], Tu et al. [88], Maruf and Haffari [89]). It is possible to hypothesize that the guided-NMT approach could be adapted to handle similar cases. In particular, by choosing a suitable representation for the document-level context of the sentence being currently translated, it would be possible to find similar contexts in an extra corpus and then just apply the guided-NMT approach as it was presented in Chapter 5. This document-level context could be defined either by a bag of words over the last n sentences, or by the respective sentence embeddings.

As for the approach followed in this work, it is possible to highlight some particular aspects that could be further studied in the future. *Faiss* (Johnson et al. [72]) is an interesting approach to the problem of retrieving similar sentences, since it is not based on exact matches or counts of words, but rather on similarities between dense vectors. Thus, to take the most out of this approach it is important to have strong representations for the sentences. As discussed in Subsection 5.1.4 there are plenty of options to consider when creating sentence embeddings. Therefore, it would be beneficial to have an empirical study showing how the different approaches to the creation of sentence embeddings affect the obtained results.

The strategy of no longer rewarding translation pieces after they are part of the current translation hypothesis proved to be important for the success of guided-NMT. Nonetheless, it is possible to use more robust solutions. In particular, since subwords may appear more than once in the retrieved sentences' translations, it may be necessary to keep rewarding them. A possible solution would be to store the maximum number of times each *n*-gram appeared in the translations of the retrieved sentences. Thus, each subword may be rewarded up until appearing a maximum number of times in the current translation.

Another possible source of improvements is due to the rewarding step of guided-NMT. In particular, it would be interesting to explore new ways of biasing the NMT output layer. Two promising strategies for further consideration are presented in Arthur et al. [20]. In particular, this work proposes: biasing the output probabilities by adding a reward to the desired subwords' scores before the softmax transformation is applied; and to interpolate the model probabilities with existing probabilities for given lexicons.

Finally, when using guided-NMT to perform domain adaptation, the unseen subwords corresponding to translation pieces were added on-the-fly to the original vocabulary. Nonetheless, those additions were not able to be very impactful, due to their scarcity. Thus, it would be valuable to test this strategy in datasets where more unseen subwords, or words that cannot be built using the available subwords, appear at inference time.

Bibliography

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

[2] P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, 2003.

[3] D. Marcu and W. Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, 2002.

[4] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[6] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[7] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[9] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 2014 Advances in Neural Information Processing Systems*, 2014.

[10] M.-T. Luong. *Neural Machine Translation*. PhD thesis, Stanford University, 2016.

[11] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

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

[13] J. Gehring, M. Auli, D. Grangier, and Y. Dauphin. A convolutional encoder model for neural machine translation. In *Proceedings of the 2017 Annual Meeting of the Association for Computational Linguistics*, 2017.

[14] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*, 2015.

[15] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 2017 Advances in Neural Information Processing Systems*, 2017.

[17] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 2016 Annual Meeting of the Association for Computational Linguistics*, 2016.

[18] M. Snover, N. Madnani, B. J. Dorr, and R. Schwartz. Fluency, adequacy, or hter?: exploring different human judgments with a tunable mt metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2009.

[19] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. Modeling coverage for neural machine translation. In *Proceedings of the 2016 Annual Meeting of the Association for Computational Linguistics*, 2016.

[20] P. Arthur, G. Neubig, and S. Nakamura. Incorporating discrete translation lexicon into neural machine translation. In *Proceedings of the 2016 Empirical Methods in Natural Language Processing*, 2016.

[21] P. Koehn and R. Knowles. Six challenges for neural machine translation. In *Proceedings of the 2017 Workshop on Neural Machine Translation*, 2017.

[22] H. Mi, B. Sankaran, Z. Wang, and A. Ittycheriah. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[23] Z. Tu, Y. Liu, Z. Lu, X. Liu, and H. Li. Context gates for neural machine translation. In *Transactions of the Association for Computational Linguistics*, volume 5, pages 87–99, 2017.

[24] Z. Tu, Y. Liu, L. Shang, X. Liu, and H. Li. Neural machine translation with reconstruction. In *Proceedings of the 2017 AAAI Conference on Artificial Intelligence*, 2017.

[25] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. Technical report, Google Brain Team, 2016.

[26] C. Hokamp and Q. Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 2017 Annual Meeting of the Association for Computational Linguistics*, 2017.

[27] R. Chatterjee, M. Negri, M. Turchi, M. Federico, L. Specia, and F. Blain. Guiding neural machine translation decoding with external knowledge. In *Proceedings of the 2017 Conference on Machine Translation*, 2017.

[28] M. A. Farajian, M. Turchi, M. Negri, and M. Federico. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the 2017 Conference on Machine Translation*, pages 127–137, 2017.

[29] X. Li, J. Zhang, and C. Zong. One sentence one model for neural machine translation. *arXiv preprint arXiv:1609.06490*, 2016.

[30] J. Gu, Y. Wang, K. Cho, and V. O. Li. Search engine guided non-parametric neural machine translation. In *Proceedings of the 2018 AAAI Conference on Artificial Intelligence*, 2018.

[31] Y. Feng, S. Zhang, A. Zhang, D. Wang, and A. Abel. Memory-augmented neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[32] J. Zhang, M. Utiyama, E. Sumita, G. Neubig, and S. Nakamura. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 North American Chapter of the Association for Computational Linguistics*, 2018.

[33] C. Malaviya, P. Ferreira, and A. Martins. Sparse and constrained attention for neural machine translation. In *Proceedings of the 2018 Annual Meeting of the Association for Computational Linguistics*, 2018.

[34] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the 2011 International Conference on Artificial Intelligence and Statistics*, 2011.

[35] G. Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. Technical report, Carnegie Mellon University, 2017. `https://arxiv.org/abs/1703.01619`.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Volume 1: Foundations*, pages 318–362. MIT Press, 1987.

[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[38] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 2015 International Conference for Learning Representations*, 2015.

[39] Y. Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[40] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[41] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*, 1(1):237–243, 2001.

[42] Y. Goldberg. *Neural Network Methods for Natural Language Processing*. Morgan & Claypool, 2017.

[43] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. *Acoustics, Speech, and Signal Processing*, 1:181–184, 1995.

[44] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

[45] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of the 2013 International Conference on Learning Representations*, 2013.

[46] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing*, 2014.

[47] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[48] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[49] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155, 2003.

[50] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Proceedings of the 2010 Annual Conference of the International Speech Communication Association*, 2010.

[51] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[52] P. Koehn. Draft of chapter 13: Neural machine translation. `https://arxiv.org/abs/1709.07809`, 2017.

[53] H. Petrie, J. Darzentas, and T. Walsh. *Universal Design 2016: Learning from the Past, Designing for the Future: Proceedings of the 3rd International Conference on Universal Design (UD 2016), York, United Kingdom, August 21–24, 2016*, volume 229. IOS Press, 2016.

[54] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

[55] T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 2015 Annual Meeting of the Association for Computational Linguistics and the 2015 International Joint Conference on Natural Language Processing*, 2015.

[56] A. Martins and R. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 2016 International Conference on Machine Learning*, 2016.

[57] A. Martins and J. Kreutzer. Learning what's easy: Fully differentiable neural easy-first taggers. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[58] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[59] V. Sadler and R. Vendelmans. Pilot implementation of a bilingual knowledge bank. In *Proceedings of the 1990 International Conference on Computational Linguistics*, 1990.

[60] M. Nagao. A framework of a mechanical translation between japanese and english by analogy principle. *Artificial and Human Intelligence*, pages 351–354, 1984.

[61] C. Chu and R. Wang. A survey of domain adaptation for neural machine translation. *arXiv preprint arXiv:1806.00258*, 2018.

[62] M.-T. Luong and C. D. Manning. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 2015 International Workshop on Spoken Language Translation*, 2015.

[63] C. Chu, R. Dabre, and S. Kurohashi. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 2017 Annual Meeting of the Association for Computational Linguistics*, 2017.

[64] S. Feng, S. Liu, M. Li, and M. Zhou. Implicit distortion and fertility models for attention-based encoder-decoder nmt model. *arXiv preprint arXiv:1601.03317*, 2016.

[65] Z. Zheng, H. Zhou, S. Huang, L. Mou, X. Dai, J. Chen, and Z. Tu. Modeling past and future for neural machine translation. *Transactions of the Association of Computational Linguistics*, 6:145–157, 2018.

[66] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 2002 Annual Meeting of the Association for Computational Linguistics*, 2002.

[67] S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the Association for Computational Linguistics 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005.

[68] M. Denkowski and A. Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the European Chapter of the Association for Computational Linguistics 2014 Workshop on Statistical Machine Translation*, 2014.

[69] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher. Non-autoregressive neural machine translation. In *Proceedings of the 2018 International Conference on Learning Representations*, 2018.

[70] J. Yang, B. Zhang, Y. Qin, X. Zhang, Q. Lin, and J. Su. Otem&utem: Over-and under-translation evaluation metric for nmt. In *Proceedings of the 2018 CCF International Conference on Natural Language Processing and Chinese Computing*, 2018.

[71] C. Dyer, V. Chahuneau, and N. A. Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 North American Chapter of the Association for Computational Linguistics*, 2013.

[72] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[73] F. Stahlberg, A. de Gispert, E. Hasler, and B. Byrne. Neural machine translation by minimising the bayes-risk with respect to syntactic translaton lattices. In *Proceedings of the 2017 Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

[74] S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the 2017 International Conference on Learning Representations*, 2017.

[75] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Proceedings of the 2015 Advances in Neural Information Processing Systems*, 2015.

[76] F. Hill, K. Cho, and A. Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[77] L. Logeswaran and H. Lee. An efficient framework for learning sentence representations. In *Proceedings of the 2018 International Conference on Learning Representations*, 2018.

[78] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[79] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[80] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. In *Proceedings of the 2018 Conference on Empirical Method in Natural Language Processing: System Demonstrations*, 2018.

[81] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. In *Proceedings of the 2018 International Conference on Learning Representations*, 2018.

[82] M. Cettolo, C. Girardi, and M. Federico. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 2012 Conference of the European Association for Machine Translation*, 2012.

[83] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 2007 Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, 2007.

[84] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush. Opennmt: Open-source toolkit for neural machine translation. *Proceedings of the 2017 Association for Computational Linguistics: System Demonstrations*, 2017.

[85] N. Srivastava. *Improving neural networks with dropout*. PhD thesis, University of Toronto, 2013.

[86] P. Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

[87] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *Proceedings of the 2015 Advances in Neural Information Processing Systems*, 2015.

[88] Z. Tu, Y. Liu, S. Shi, and T. Zhang. Learning to remember translation history with a continuous cache. *Transactions of the Association of Computational Linguistics*, 6:407–420, 2018.

[89] S. Maruf and G. Haffari. Document context neural machine translation with memory networks. In *Proceedings of the 2018 Annual Meeting of the Association for Computational Linguistics*, 2018.

[90] P. M. Pardalos and N. Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46(1-3):321–328, 1990.

Appendix A

Constrained Sparsemax

Constrained Sparsemax was presented in Section 4.1 as a constrained approach to sparsemax, by allowing to use upper bounds on the sparse probability distribution that is yielded. It is defined as

$$\begin{aligned} &\mathrm{csparsemax}(\boldsymbol{z};\boldsymbol{u}) := \underset{\boldsymbol{\alpha}\in\Delta^{J-1}}{\arg\min}\, ||\boldsymbol{\alpha}-\boldsymbol{z}||^2, \\ &\text{subject to} \quad \boldsymbol{\alpha} \leq \boldsymbol{u} \end{aligned} \tag{A.1}$$

where $\boldsymbol{z} \in \mathbb{R}^J$ is a vector of scores, $\boldsymbol{u} \in \mathbb{R}^J$ is the upper bounds vector and $\boldsymbol{\alpha}$ are the resulting attention scores. We now give further proof of its behavior, following Malaviya et al. [33].

Defining $\boldsymbol{\alpha}^\star$ as the solution of Equation A.1, and the auxiliary sets $\mathcal{A} = \{j \in [J] \mid 0 \leq \alpha_j^\star \leq u_j\}$, $\mathcal{A}_L = \{j \in [J] \mid \alpha_j^\star = 0\}$ and $\mathcal{A}_R = \{j \in [J] \mid \alpha_j^\star = u_j\}$, it is possible to make some considerations about the forward propagation and backpropagation steps.

Forward Propagation

A solution for constrained sparsemax may be obtained in $O(J)$ time, making use of Pardalos and Kovoor's algorithm (Pardalos and Kovoor [90]). The yielded solution is defined as $\alpha_j^\star = \max\{0, \min\{u_j, z_j - \tau\}\}$, being τ a normalization constant.

Proof

The problem defined in Equation A.1 may be written as

$$\begin{aligned} \mathrm{csparsemax}(\boldsymbol{z},\boldsymbol{u}) = \arg\min \quad & \tfrac{1}{2}\|\boldsymbol{\alpha}\|^2 - \boldsymbol{z}^\top\boldsymbol{\alpha} \\ \text{s.t.} \quad & \begin{cases} \mathbf{1}^\top\,\boldsymbol{\alpha} = 1 \\ \mathbf{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{u}, \end{cases} \end{aligned}$$

and the corresponding Lagrangian function is:

$$\mathcal{L}(\boldsymbol{\alpha}, \tau, \boldsymbol{\mu}, \boldsymbol{\nu}) = \frac{1}{2}\|\boldsymbol{\alpha}\|^2 - \boldsymbol{z}^\top \boldsymbol{\alpha} + \tau(\mathbf{1}^\top \boldsymbol{\alpha} - 1) - \boldsymbol{\mu}^\top \boldsymbol{\alpha} + \boldsymbol{\nu}^\top(\boldsymbol{\alpha} - \boldsymbol{u}). \tag{A.2}$$

From here, it is possible to use the Karush-Kuhn-Tucker conditions to find the solution to the Lagrangian function. First, from the stationary condition we can define $0 = \boldsymbol{\alpha} - \boldsymbol{z} + \tau\mathbf{1} - \boldsymbol{\mu} + \boldsymbol{\nu}$, which due to the primal feasibility condition, has a solution is given by $\boldsymbol{\alpha} = \boldsymbol{z} - \tau\mathbf{1} + \boldsymbol{\mu} - \boldsymbol{\nu}$. From the complementary slackness condition we have $-\boldsymbol{\mu}\boldsymbol{\alpha} = 0$ and $\boldsymbol{\nu}(\boldsymbol{\alpha} - \boldsymbol{u}) = 0$. Paired with the dual feasibility condition, which makes $\boldsymbol{\mu} \geq 0$ and $\boldsymbol{\nu} \geq 0$, and with the constraint $\mathbf{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{u}$, there are several different scenarios. Namely,

1. If $0 < \alpha_j < u_j$, then $\mu_j = \nu_j = 0$, and $\alpha_j = z_j - \tau$.
2. If $\mu_j > 0$, then $\alpha_j = 0$.
3. If $\nu_j > 0$, then $\alpha_j = u_j$.

Therefore, the solution may be written as $\alpha_j = \max\{0, \min\{u_j, z_j - \tau\}\}$. Notice that the first situation refers to a probability value between 0 and the upper bound, the second refers to the case in which the sum of all probabilities is already one, and thus, the constraint $\mathbf{1}^\top \boldsymbol{\alpha} = 1$ is active, and the final one refers to when the upper bound constraint for a given probability is active. The value of τ is given by

$$\tau = \frac{\sum_{j\in\mathcal{A}} z_j + \sum_{j\in\mathcal{A}_R} u_j - 1}{|\mathcal{A}|}, \tag{A.3}$$

with $\mathcal{A} = \{j \in [J] \mid 0 < \alpha_j < u_j\}$ and $\mathcal{A}_R = \{j \in [J] \mid \alpha_j = u_j\}$. This value τ may be obtained efficiently by making use of the Pardalos and Kovoor's algorithm (Pardalos and Kovoor [90]).

Backpropagation

The backpropagation step may be solved in sublinear time, more concretely, $O(|\mathcal{A}| + |\mathcal{A}_\mathcal{R}|)$. Being $L(\boldsymbol{\theta})$ the chosen loss function, and by simplifying the notation of the gradients by using $\mathrm{d}x = \Delta_x L(\boldsymbol{\theta})$, it is possible to write the gradient of the inputs, $\boldsymbol{z}$ and $\boldsymbol{u}$ as

$$\mathrm{d}z_j = \mathbb{1}(j \in \mathcal{A})(\mathrm{d}\alpha_j - m) \tag{A.4}$$

$$\mathrm{d}u_j = \mathbb{1}(j \in \mathcal{A}_R)(\mathrm{d}\alpha_j - m) \tag{A.5}$$

where $m = \frac{1}{|\mathcal{A}|}\sum_{j\in\mathcal{A}} \mathrm{d}\alpha_j$.

Proof

Backpropagating the gradients requires calculating the Jacobian matrix, i.e., the derivatives $\frac{\partial \alpha_i}{\partial z_j}$ and $\frac{\partial \alpha_i}{\partial u_j}$ for $i, j \in [J]$. The first step is to define the vector of attention probabilities, $\boldsymbol{\alpha}$, given by

$$\alpha_i = \begin{cases} 0, & i \in \mathcal{A}_L, \\ z_i - \tau, & i \in \mathcal{A}, \\ u_i, & i \in \mathcal{A}_R, \end{cases} \tag{A.6}$$

Notice that the probability values for each branch are accordingly to the expected and easily interpretable. Notice as well the presence of τ, as defined in Equation A.3. Therefore, it is necessary to obtain the corresponding partial derivatives. These are given by

$$\frac{\partial \tau}{\partial z_j} = \mathbb{1}(j \in \mathcal{A})/|\mathcal{A}| \tag{A.7}$$

$$\frac{\partial \tau}{\partial u_j} = \mathbb{1}(j \in \mathcal{A}_R)/|\mathcal{A}|, \tag{A.8}$$

making it possible to write the desired partial derivatives as

$$\frac{\partial \alpha_i}{\partial z_j} = \begin{cases} 1 - 1/|\mathcal{A}|, & \text{if } j \in \mathcal{A} \text{ and } i = j \\ -1/|\mathcal{A}|, & \text{if } i, j \in \mathcal{A} \text{ and } i \neq j \\ 0, & \text{otherwise,} \end{cases} \tag{A.9}$$

$$\frac{\partial \alpha_i}{\partial u_j} = \begin{cases} 1, & \text{if } j \in \mathcal{A}_R \text{ and } i = j \\ -1/|\mathcal{A}|, & \text{if } j \in \mathcal{A}_R \text{ and } i \in \mathcal{A} \\ 0, & \text{otherwise.} \end{cases} \tag{A.10}$$

Finally, it is possible to write:

$$\begin{aligned} \mathrm{d}z_j &= \sum_i \frac{\partial \alpha_i}{\partial z_j} \mathrm{d}\alpha_i \\ &= \mathbb{1}(j \in \mathcal{A}) \left(\mathrm{d}\alpha_j - \frac{\sum_{i \in \mathcal{A}} \mathrm{d}\alpha_i}{|\mathcal{A}|} \right) \\ &= \mathbb{1}(j \in \mathcal{A})(\mathrm{d}\alpha_j - m), \end{aligned} \tag{A.11}$$

$$\begin{aligned} \mathrm{d}u_j &= \sum_i \frac{\partial \alpha_i}{\partial u_j} \mathrm{d}\alpha_i \\ &= \mathbb{1}(j \in \mathcal{A}_R) \left(\mathrm{d}\alpha_j - \frac{\sum_{i \in \mathcal{A}} \mathrm{d}\alpha_i}{|\mathcal{A}|} \right) \\ &= \mathbb{1}(j \in \mathcal{A}_R)(\mathrm{d}\alpha_j - m), \end{aligned} \tag{A.12}$$

where $m = \frac{1}{|\mathcal{A}|} \sum_{j \in \mathcal{A}} \mathrm{d}\alpha_j$.

Appendix B

Examples

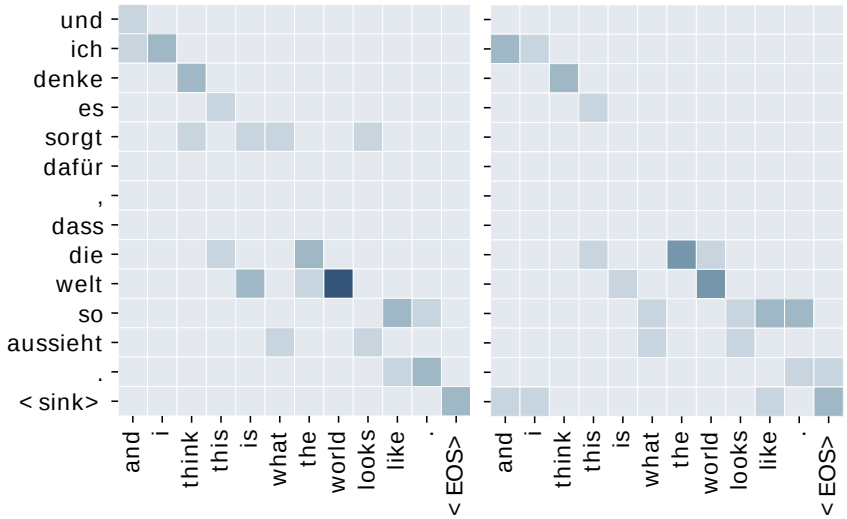
B.1 Fertility-Based NMT - Translation Examples

Table B.1: Fertility-based NMT - Extra examples. The golden color indicates the portion of the reference that the example targets, the **red color** indicates the mistranslations/repetitions and finally, the *blue color* highlights correct translations of the parts of the sentence being focused. The <**MISS**> symbol refers to portions of the sentence that were missed.

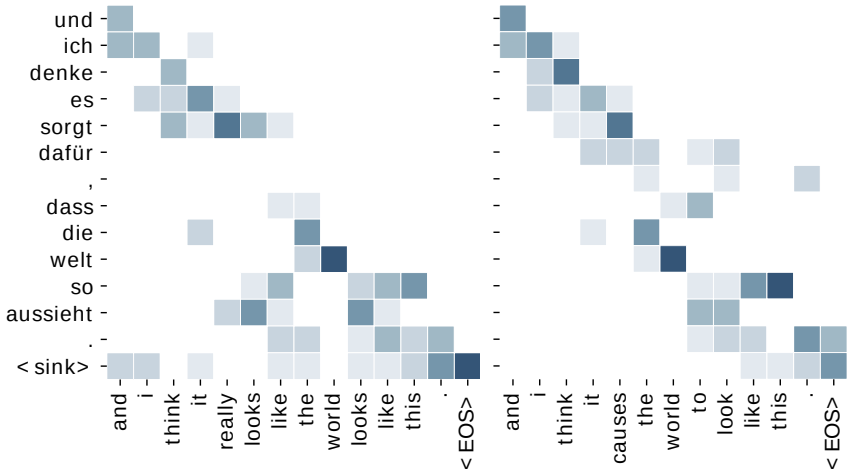
	Example 3
Reference	Death penalty in America is defined by error.
Softmax	*The death penalty* **death penalty penalty** <**MISS**> is defined by error.
Sparsemax	*Death penalty in America* is defined by error.
CSoftmax	*The death penalty in America* is defined by error.
CSparsemax	*The death penalty* **penalty** *in America* is defined by error.
	Example 4
Reference	It's not that simple, but it's also not impossible, and if you are curious enough and put enough energy into it, you can go far.
Softmax	It's not so easy, **but it's not impossible, but it's not impossible**, if you have enough curiosity, you **get** enough energy <**MISS**>, you **get there**.
Sparsemax	It's not so easy, *but it's not impossible*, if you have enough curiosity, and **there's** enough energy in it, then you **get there too**.
CSoftmax	It's not so simple now, *but it's not impossible*, if you've got enough curiosity and **there's** enough energy <**MISS**>, you *get far* **there**.
CSparsemax	It's not so simple now, *but it's not impossible*, if you've got enough curiosity and you *put* enough energy **inside**, you **get too** *far*.
	Example 5
Reference	This is energy farming, so you are talking about many square miles, thousands of time more area than you think of as a normal energy plant.
Softmax	That's energy <**MISS**>, you are talking about many square miles, <**MISS**> more <**MISS**> than a normal <**MISS**> **power power**.
Sparsemax	This is energy **force**, you are talking about many square miles, *thousands times* more <**MISS**> than a normal *power station*.
CSoftmax	That's energy <**MISS**>, you are talking about many square miles, a *thousand times* more *surface* than an normal **power power**.
CSparsemax	These are energy **products**, you are talking about many square miles, a *thousand times* more *area* than an ordinary *power station*.

B.2 Fertility-Based NMT - Attention Maps

We provide further attention maps using the German to English language pair.

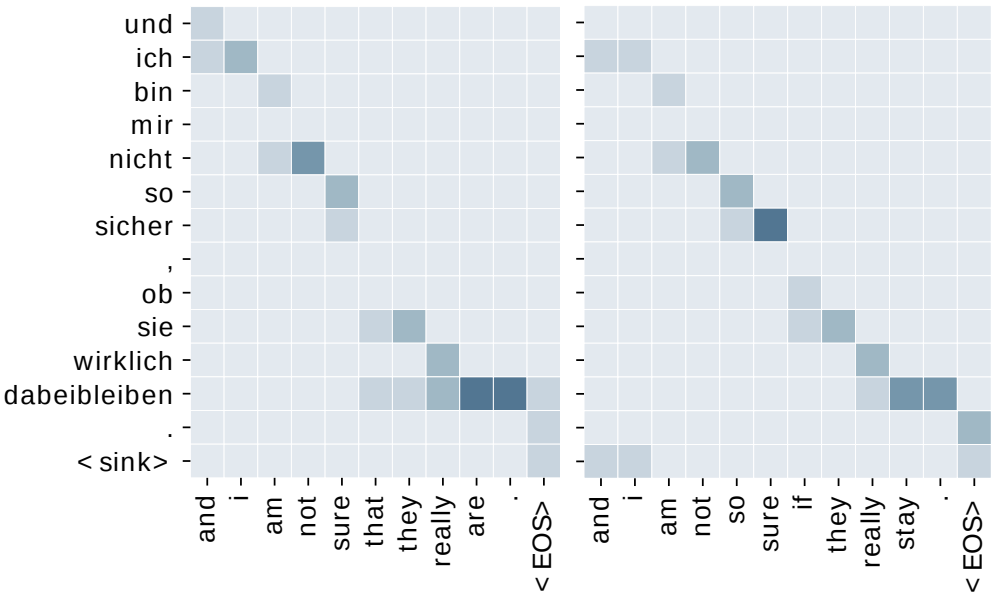


(a) Left: Softmax Right: Constrained Softmax

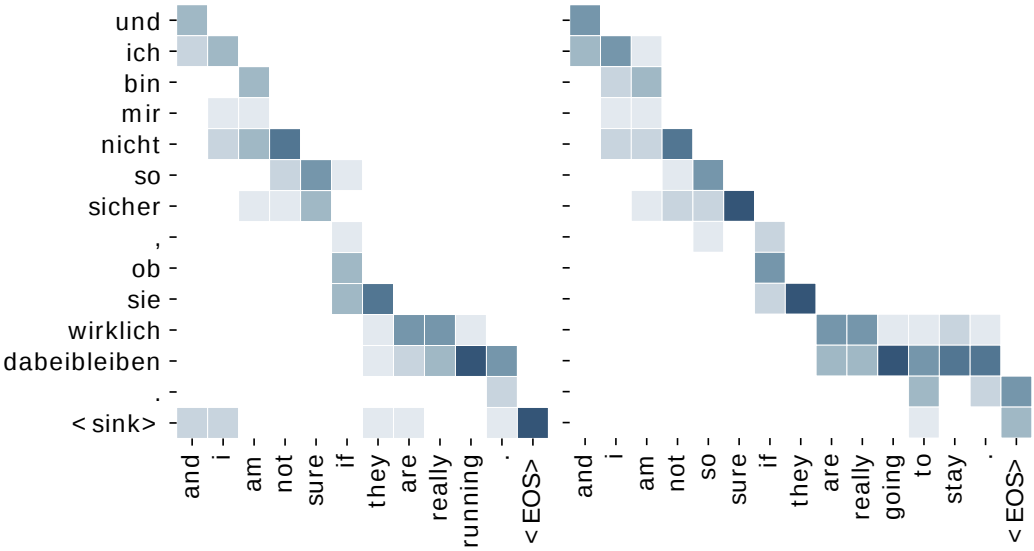


(b) Left: Sparsemax Right: Constrained Sparsemax

Figure B.1: Fertility-based NMT - Attention map showing the different distributions for each proposed attention transformation. Reference sentence is: *"And I think it makes the world look like this."*.



(a) Left: Softmax Right: Constrained Softmax



(b) Left: Sparsemax Right: Constrained Sparsemax

Figure B.2: Fertility-Based NMT - Attention map showing the different distributions for each proposed attention transformation. Reference sentence is: *"And I am not so sure they are going to stick with it."*.

B.3 In-domain Guided-NMT - Translation Examples

Table B.2: In-domain guided-NMT - Extra examples. The golden color indicates the portion of the reference that the example targets, the **red color** indicates the mistranslations and finally, the *blue color* highlights correct translations of the parts of the sentence being focused. The <**MISS**> symbol refers to portions of the sentence that were missed.

	Example 5
Reference	The forceps according to any preceding claim, wherein said bullet is comprised of a biodegradable material.
Base	The forceps according to any one of the preceding claims, wherein **the socket** is made of a biodegradable material.
Guided	The forceps according to any preceding claim, wherein said *bullet* is comprised of a biodegradable material.
	Example 6
Reference	System and apparatus for extracting drugs
Base	System and device for **taking** pharmaceuticals
Guided	System and apparatus for *extracting* pharmaceuticals
	Example 7
Reference	A compound according to claim 8 for use as an antitumor agent.
Base	A compound according to claim 8 for use as an **anti-cancer** agent.
Guided	A compound according to claim 8 for use as an *antitumor* agent.
	Example 8
Reference	Tenofovir disoproxil fumarate did not show any carcinogenic potential in a long-term oral carcinogenicity study in rats .
Base	Tenofovir disoproxil fumarate demonstrated no carcinogenic potential in a long-term <**MISS**> carcinogenicity study in rats .
Guided	Tenofovir disoproxil fumarate did not show any carcinogenic potential in a long-term *oral* carcinogenicity study in rats .

B.4 Guided-NMT for Domain Adaptation - Translation Examples

Table B.3: Guided-NMT for domain adaptation - Extra examples. The golden color indicates the portion of the reference that the example targets, the **red color** indicates the mistranslations and finally, the *blue color* highlights correct translations of the parts of the sentence being focused. The <**MISS**> symbol refers to portions of the sentence that were missed.

	Example 5
Reference	Agent according to claim 10, characterized in that the ready-to-use oxidation colorant has a ph of from 3 to 11.
Base	**Funds under the age of 10** are characterised by the fact that the **use of used oxiants is a value worth** 3 to 11.
Guided	**Funds** *according to claim 10*, characterized in that the *ready-to-the-to-use oxidation dye has a ph* value of 3 to 11.
	Example 6
Reference	If signs and symptoms of tardive dyskinesia appear, the discontinuation of all antipsychotics, including paliperidone, should be considered.
Base	If there are signs or symptoms of **targeted dybel**, **it is intended to take into account all anti-psychologists**, including **paliadon**.
Guided	If signs or symptoms of *tardive dyskinesia* appear, *the discontinuation of all antipsychotics*, including **palissia**, *should be considered*.
	Example 7
Reference	Your doctor will advise you and decide how long you should be treated with zoledronic acid teva generics.
Base	Your doctor will advise **her** and decide how long **they** should be treated with **customs acid acid** <**MISS**>.
Guided	Your doctor will advise *you* and decide how long **they** should be treated with *zoledronic acid teva* <**MISS**>.
	Example 8
Reference	Elderly patients may have increased trimetazidine exposure due to age-related decrease in renal function.
Base	Older patients can have higher **trimetaic** exposure due to a **deterioration in the** *age* **of retirement**.
Guided	Elderly patients may have increased **trimetazidan** exposure due to *age-related decrease in the kidney function*.