Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

# DEEP LEARNING MODELS APPLIED TO MACHINE TRANSLATION UNDER LOW-RESOURCE SETTINGS: A PORTUGUESE-ENGLISH CASE STUDY

Arthur Telles Estrella

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: João Baptista de Oliveira e Souza Filho, D.Sc.

Rio de Janeiro
Dezembro de 2022

# DEEP LEARNING MODELS APPLIED TO MACHINE TRANSLATION UNDER LOW-RESOURCE SETTINGS: A PORTUGUESE-ENGLISH CASE STUDY

Arthur Telles Estrella

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: João Baptista de Oliveira e Souza Filho, D.Sc.

Aprovada por: Prof. João Baptista de Oliveira e Souza Filho, D.Sc.
Prof. Marcello Luiz Rodrigues de Campos, Ph.D.
Prof. José Alfredo Ferreira Costa, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2022

Telles Estrella, Arthur

Deep learning models applied to machine translation under low-resource settings: a Portuguese-English case study/Arthur Telles Estrella. – Rio de Janeiro: UFRJ/COPPE, 2022.

XIII, 120 p.: il.; 29, 7cm.

Orientador: João Baptista de Oliveira e Souza Filho, D.Sc.

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2022.

Referências Bibliográficas: p. 110 – 119.

1. Tradução automática. 2. Aprendizado profundo. 3. Processamento de linguagem natural. 4. Aprendizado de máquina. 5. Redes neurais. I. Baptista de Oliveira e Souza Filho, D.Sc., João. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

*Dedico este trabalho a minha*
*família, principalmente meus*
*pais Luciene e Guilherme e ao*
*meu amor, Débora*

# Agradecimentos

Agradeço ao empenho dos meus pais em me orientar pelos caminhos corretos. Eles foram responsáveis por me incentivar a entrar em uma busca interminável por conhecimento, e que me levou a alcançar essa conquista. Agradeço ao carinho, bons valores e conforto que foram sempre prioridades da minha tia Sueli (*in memoriam*), avó Sidnéia (*in memoriam*) e Tia Maria (*in memoriam*). Agradeço também ao meu avô Leonídio (*in memoriam*), que serviu de exemplo de que o trabalho pode render grandes frutos, independentemente do nível de conhecimento com que se começa. Homenageio também minha irmã pela parceria disponível a qualquer momento e ao meu amor Débora pelo forte apoio e incentivo na reta final deste trabalho.

Sou grato ao professor João Baptista Filho pela inserção no mundo da Estatística e Aprendizado de Máquina, pela sua orientação, que já vem desde os primeiros passos da graduação e o alto valor por ele agregado em importantes decisões tomadas durante o curso desta dissertação. Agradeço às experiências de trabalho privado que tive e especialmente aos meus principais mentores durante esta trajetória. Eles foram responsáveis por me capacitar e revelar diversas oportunidades, abrindo incontáveis portas e contribuindo para o profissional que sou hoje.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MODELOS NEURAIS PROFUNDOS PARA TRADUÇÃO EM CENÁRIOS DE BAIXO RECURSO: UM ESTUDO DE CASO PORTUGUÊS-INGLÊS

Arthur Telles Estrella

Dezembro/2022

Orientador: João Baptista de Oliveira e Souza Filho, D.Sc.

Programa: Engenharia Elétrica

Com o avanço das técnicas de Processamento de Linguagem Natural, os modelos de tradução automática ganharam espaço, porém os erros produzidos por tais modelos são pouco explorados ou analisados. O problema se agrava ao usar poucos dados para o treino (*low-resource*), devido a uma limitação de vocabulário e contexto. Há técnicas para lidar com as limitações de domínios de baixo recurso, tais como *Subword Embeddings*, *Pre-trained Word Embeddings* e *Back Translation*. Porém, os vieses inseridos por elas na tradução não são amplamente estudados. Esta dissertação se diferencia dos trabalhos trabalhos mais recentes, que normalmente focam em aumentar o BLEU (Bilingual Evaluation Understudy) escore, negligenciando os erros produzidos. Deste modo, são propostas técnicas qualitativas e quantitativas de avaliação, utilizando-se de testes de hipóteses e de ferramentas de Aprendizado de Máquina para se avaliar os vieses de modelos de tradução automática, submetidos à baixa disponibilidade de dados e treinado em uma única GPU. O estudo é focado no par Português-Inglês, sendo o BLEU usado como métrica quantitativa de referência. Uma análise qualitativa é conduzida junto a um tradutor humano para um melhor entendimento dos padrões de erros em frases e complementada com uma estratificação em diferentes níveis de complexidade, por meio da escala CEFR (Common European Framework of Reference for Languages), para se avaliar a correlação entre estas variáveis. Modelos de baixo recurso derivados do *Transformer* são usados e comparados com o *Google Translate*, sendo que o melhor dentre eles é capaz de atingir 40,26 pontos no BLEU, correspondendo a 77.1% do referencial em 01/2022. Este resultado reforça o potencial de uma abordagem eficiente *low-resource* em alcançar uma performance comparável a modelos complexos no estado da arte, mesmo sujeito a restrições de dados, energia e infraestrutura.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DEEP LEARNING MODELS APPLIED TO MACHINE TRANSLATION UNDER LOW-RESOURCE SETTINGS: A PORTUGUESE-ENGLISH CASE STUDY

Arthur Telles Estrella

December/2022

Advisor: João Baptista de Oliveira e Souza Filho, D.Sc.

Department: Electrical Engineering

Neural Machine Translation models have flourished with the advancement of the Natural Language Processing techniques, however, the errors produced by these models are little explored or analysed. This issue aggravates when using a small dataset for training (low-resource), leveraged by a vocabulary limitation. There are techniques to deal with low-resource domain limitations in the context of Machine Translation, like Subword Embeddings, Pre-trained Word Embeddings and Back Translation; but the bias inserted by them in the translations hasn't been widely studied. This dissertation differs from recent works, that usually focus on increasing the BLEU (Bilingual Evaluation Understudy) score, overlooking the errors produced by the models. We propose qualitative and quantitative evaluation frameworks, exploring hypothesis tests and Machine Learning tools to evaluate the biases of the translation models exposed to a low-data availability and trained on a single GPU. The study focuses on the Portuguese-English pair, and BLEU is used as a quantitative benchmark metric. A qualitative analysis is conducted along with a human translator to understand the patterns of errors in the sentences and is also complemented with a stratification of the content in different complexity levels, using the CEFR (Common European Framework of Reference for Languages) scale, to evaluate the correlation between these factors. Transformer-based models are used and compared with Google Translate, where the best model built is capable of reaching 40.26 points of BLEU, 77.1% of the value achieved by the reference at 01/2022. This result reinforces the potential of using efficient low-resource approaches to reach a performance comparable to much more complex state-of-the-art models, even under constraints over data availability, energy and infrastructure.

# Contents

# List of Figures

# List of Algorithms

# List of Tables

# Chapter 1

# Introduction

## 1.1 The reenactment of Machine Translation

Machine translation is a research field that until 2013 has mainly invested in statistical based models, but the breakthrough promoted by the Sequence to Sequence (*Seq2Seq*) algorithms followed by the use of Transformer-based models has significantly changed the focus of the field. Before Neural Networks, Machine Translation systems were rule-based, syntax-based, phrase-based or a blend between more than one of these techniques. Probabilistic models used to be considered the state-of-the-art before the first *Seq2Seq* model paper. The increase in performance promoted by *Seq2Seq* and Transformers received some attention, and soon other variants were developed.

The challenge of translating text got constrained by computational power issues for many years, until the rise of the Transformers, which turned possible to perform the high amount of computation required in these cases in a truly parallel schema. With this new architecture, the operations performed during training are not totally dependent, allowing them to be parallelized in Graphic Processing Units (GPUs). Thus, by removing the constraint that some operations must wait for others to finish, NMT (Neural Machine Translation) models could scale largely and reach higher quality translations.

Just in 3 years, NMT became the dominant approach to Machine Translation, inducing a major transition from statistical to neural models. It was also the root cause for an increase in the demand for data, which subsequently drove attention to the importance of aiming for resource efficient models.

## 1.2 Challenges for the Portuguese language

Traditionally, the Machine Translation datasets and related conferences usually focus on languages from countries that are actively investing on NLP, which biases and narrows the potential of such algorithms towards a specific domain. Unfortunately, as of 2020, Portuguese is a language that does not dispose of abundant supervised translation data, specially considering a diversity of domains. This issue increases the struggle to build a model that can successfully translate text in this idiom to other languages. Another obstacle is the existence of the European, Brazilian and African Portuguese variants, which leads to a challenge for the model since generalization is harder if several sentences with different dialects can have the same meaning. We have performed an extensive search for works with a qualitative analysis of translation results for Brazilian Portuguese, and only one single paper was found [1], indicating a vast unexplored terrain when considering the analysis of translation errors produced by Machine Translation.

The branch of NMT inside the big topic of Natural Language Processing (NLP) is also a field with few papers and academic works among many Brazilian universities. This can be partially explained by the challenge that this environment imposes over practitioners: most models require cutting-edge GPUs and usually only one GPU is not enough for medium-sized models required by many WMT competition datasets. The scarcity of these resources to many researchers require them to circumvent these limitations and search for cloud solutions without sponsorship.

Finally, Portuguese is a complex language that uses accents which can change their meaning (*e.g.* "e" and "é"), has different pronoun placements (*e.g.* "realizar-se-á" equals "se realizará") and irregular verb inflections (*e.g.* the "pôr" and "haver" verbs) so text preprocessing and tokenization plays an important role. Disregarding these details by applying some generic preprocessing steps that eliminates accents, for instance, can lead to a worse model performance. On the other hand, exploiting some domain knowledge on NLP often helps the model to better translate or classify, depending on the task at hand.

## 1.3 Contributions of this dissertation

Many previous works for solving the NMT task have focused on surpassing the state-of-the-art, disregarding the associated computational burden for achieving this goal. The research under this scope is often solely driven by increasing scores such as BLEU, which have limitations in assessing the quality of the translations generated. This dissertation tries to be an initial effort in filling this gap, in the sense that our focus is both on assessing quantitatively and qualitatively the translations automatically produced by an NMT model, as well as discussing the techniques that can be potentially associated with these purposes, whilst considering resource efficiency under low-resource settings.

The motivation for the concentration on low-resource scenarios arises from the emergence of IoT applications, whose computing power is typically restricted. Besides, for most researchers, the task of developing a competitive translation model without a robust infrastructure is becoming increasingly hard over time. As the access to such machinery is restricted to a selected group of researchers, there is an active trend in the NLP community of investing in accessible AI, whose one of the priorities is investing on experiments that could be performed using low-cost GPUs.

As a result of the effort put on this dissertation, we published a conference paper at the Symposium in Information and Human Language Technology (STIL) [2][1]. This paper reports our early work on exploiting resource efficient Natural Language Processing (NLP) techniques to leverage translation performance, whilst also analysing the quality of resulting translations both quantitatively and qualitatively. This analysis was performed in resource constrained environments, using statistical hypothesis tests targeting the translation error patterns of each model variant.

This work concentrates on the Transformers model [3], currently the state-of-the-art in NLP, and experimentally evaluate to what extent techniques such as transfer learning, modelling of subsets of words (Subword Embeddings), and dataset augmentation with artificial sentences (Back Translation) can leverage an NMT model to excel in a low-resource environment. This experimental study is conducted with only one average-size GPU and small to medium-sized datasets, considering the English-to-Portuguese pair. In a quantitative sense, the challenge of low-resource is emulated considering samples of the dataset with a range of sizes. The goal is to infer how cost-effective are the aforementioned techniques under such settings. A qualitative analysis is also conducted in the study, where classes of errors are defined with the help of a Brazilian English translator. His participation was crucial to spot and classify mistakes in grammatical patterns and put them under well defined clusters. The translator also applied his knowledge to stratify the sentences using a complexity drill down, rated according to the CEFR scale [4]. The qualitative dimensions of sentence complexity and error patterns were subjected to 3 experiments that analysed the bidimensional associations between the model variants and these factors. In the end of the analysis we concluded that the techniques associated with the model indeed have an influence over the error patterns.

For comparison purposes, the model with the most promising score is compared with the Google Translate service, acessed by an API [5], over the test set of the main

[1]The full paper is publicly available at `https://sol.sbc.org.br/index.php/stil/article/view/17807`.

datasets used. Despite the significantly higher but precisely unknown quantity of data as well as human and computing efforts required for training the Google model, our low-resource model reached 77.1% of the Google's model BLEU score.

This work is not only an attempt to explain qualitatively and quantitatively the pros and cons of applying a specific technique in low-resource environments. We also hope that it can stimulate the community to dive deeper into error patterns and biases of specific techniques in NMT, as our belief is that this process may aid on leading the community towards producing clearer, more fluid and grammatically correct translations.

## 1.4 Dissertation Organization

The fundamentals of NLP that are the pillars for NMT, such as the types of word representations and how the translation task is performed are presented in Chapter 2. Chapter 3 introduces the Sequence to Sequence neural architectures and the Transformer model, along with some interesting properties that helped them to learn richer word representations and output more coherent translations.

In Chapter 4, aspects of low-resource domains and related constraints are outlined, along with a brief review of NLP techniques that can potentially help to reduce their side-effects on models' performance. A description of the datasets used in this work is provided in Chapter 5, which also depicts quantitative and qualitative experiments and the corresponding results for the strategies considered here. Finally, in Chapter 6, we derive some conclusions and outline further possible improvements and study directions.

# Chapter 2

# A brief introduction to Natural Language Processing

Natural language was considered by the Machine Learning community as one of the most complex data types to represent. Practical applications with meaningful feature representations took time to evolve, even longer than computer vision related tasks, which also required customized representations. In the sections below, a brief introduction to the different kinds of features used for word representations will be given to contextualize the reader over some of the historical challenges and evolution observed in this field.

## 2.1 Feature representation

Natural language can be represented either by words or tokens. The scope of tokens includes not only words, but also punctuation marks, acronyms and any other text that can be used as a placeholder. Despite that subtle distinction, both terms are used almost interchangeably when speaking of natural language representation in the literature. When referring to words or tokens throughout the next sections, we will be considering any of these sorts of natural language instances.

In order to keep the unique and original meaning of each word, one of the oldest and most popular ways to represent natural language is through one-hot encoding. Let $V$ be the number of distinct words present in a vocabulary. When this set of words is vectorized, each word will be represented by an unique not-null dimension inside a binary vector of size $V$. In other words this means that when representing one word, the vector would contain 1 for the dimension of that word and all the other positions are filled with zeroes. In this case, the vectors of two different words would be always orthogonal, i.e., have their dot product equal to 0, not allowing any measure of similarity between them. One-hot encoding is a type of sparse representation that has only one position different than zero. Another method disadvantage is employing a high number of dimensions to represent even small and medium-sized vocabularies, thus imposing relevant challenges to algorithms that cannot properly handle high-dimensional inputs.

Another possibility to represent a set of input words is using fixed-length real vectors, where the number of dimensions is pre-specified, regardless the vocabulary size. These real numbers may widely vary in range depending on how there are generated, and there is a number of techniques that can provide them. For instance, such vectors can be randomly initialized using uniform distributions and then adjusted using an optimization algorithm based on the statistics over the co-occurrence of words. In the most simple neural-based algorithms, such real number vectors are usually extracted from the weights related to the first network layer, thus can be referred to as weight vectors or neural embedding. This word encoding strategy is also known as dense representation.

There are pros and cons of using either sparse or dense vectors to represent words. A brief summary can be found at Table 2.1.

Table 2.1: Pros and Cons per Feature Representation

| Feature representation | Encoding Type | Dimensionality | Information sharing | Curse of dimensionality |
|---|---|---|---|---|
| One-hot encoding | Sparse | High | None, as the vector dimensions are independent | Sparse and high-dimensional vectors may hinder model performance |
| Neural embedding | Dense | Fixed-Length | Similar vectors (low cossine angle) for related words | Dimensionality can be experimentally tuned to a sweet spot |

We should stress that if the dimension is too high in one-hot encoding, some dimensionality reduction may be applied. However, depending on the dataset and the algorithm considered, this process may not be recommended due to the computational burden involved, as potential quality issues may arise. On the other hand, the weight encoding alternative lets the user to tune the embedding dimension, but this may require several experiments to achieve an optimal performance. The algorithm used to generate such embeddings also plays an important role in this process.

## 2.2 Neural Word Embeddings

Natural language representation using neural Word Embeddings was responsible to allow Machine Learning algorithms to scale to a range of applications, representing a watershed for the research community. Neural networks can be recognized as intrinsic dense vector learners, due to the weight update operations performed during

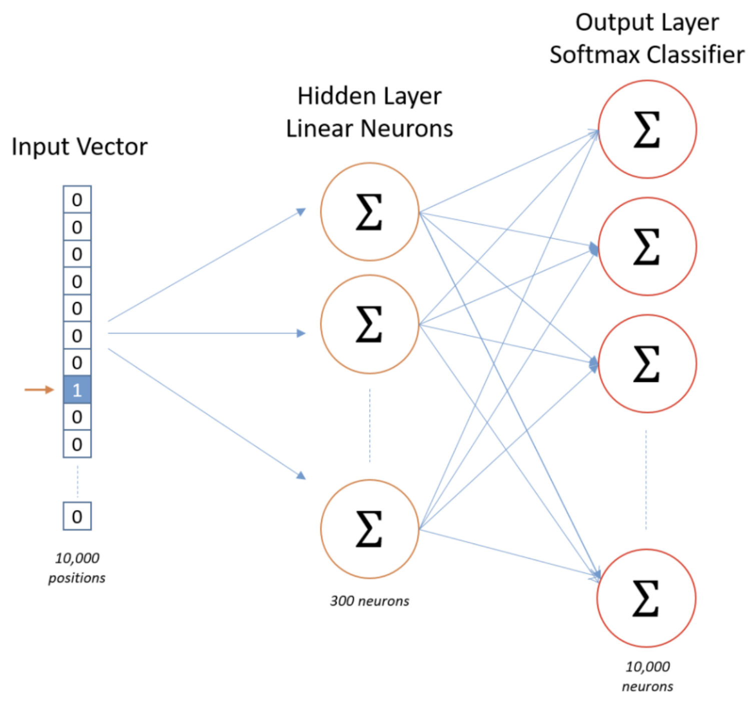


Figure 2.1: Architectural diagram for calculating word probabilities using a 2-layer Neural Network for an arbitrary 10,000 words dictionary.

backpropagation. In a neural network capable of modelling words, the weights themselves when concatenated into a vector can be optimized to represent a specific word.

The properties of each modality of feature representation previously reported in Table 2.1 are clear, but one question remains: How dense vectors are able to capture the similarity in meaning between words? One quote by John Firth from 1952 may shed light on the potential answer:

"You shall know a word by the company it keeps."

Basically, the meaning of a word can be defined by the context around it, and usually this context is extracted using a fixed-size window, whose length is a hyper-parameter of the algorithm, here denoted as $c$. To gradually understand how words relate to their context, let us first consider the association between two arbitrary words in a sentence.

### 2.2.1 Single-word context

One popular technique to obtain word weights is the two-layer neural network architecture illustrated in Figure 2.1. It is a simplified representation of a neural word embedding, taking an one-hot encoded vector $\mathbf{v}$ of size $v$ as input and generating a dense vector at the outputs of its hidden layer neurons. In this case, the weights between the input vector and the hidden layer can be represented by a $n \times v$ matrix $\mathbf{W}$, where $n$ and $v$ are the and the dimensionality of the embedding and vocabulary sizes, respectively. Therefore, each column of $\mathbf{W}$ is a $n$-dimensional vectorial representation $\mathbf{v}_w$ of the respective word in the dictionary (here all vectors are assumed as column vectors). This enables the hidden layer to become analogous to a lookup table, where each column represents a different word.

In Figure 2.1, the hidden layer outputs are simply defined by $\mathbf{h} = \mathbf{W}^T\mathbf{x}$. From the hidden layer to the output layer, there is another weight matrix $\mathbf{W}^{'} = \mathbf{w}^{'}_{Cj}$ with dimensions $n \times v$. Weights from both matrices can be used to compute scores $u_{j'}$ and $u_j$ for a given word:

$$u_{j'} = \mathbf{v}^{'T}_{w_j}\mathbf{h}, \tag{2.1}$$

and

$$u_j = \mathbf{v}^{T}_{w_j}\mathbf{h}, \tag{2.2}$$

where $1 \leq j \leq v$, whilst $\mathbf{v}^{'T}_{w_j}$ and $\mathbf{v}^{T}_{w_j}$ are the $j$-th column of the matrix $\mathbf{W}^{'}$ and $\mathbf{W}$, respectively. Now, consider the following problem: what is the most likely word that succeeds a given word? Assuming that the network is trained, one may just set the corresponding $\mathbf{v}_x$ to this word, generate the corresponding embedding $\mathbf{h}$, and apply $\mathbf{h}$ to the output layer to infer the likelihood of each vocabulary word in succeeding it. Before the likelihood inference, the score of 2.2 is then passed through a softmax layer, that calculates the probability of a set of words that could be fit into that context.

In this case, the output layer is responsible for inferring the level of association of a word with another word in its context (in this example, the word succeeding it), or vice versa, depending on the approach used. Thus, the probability associated with each vocabulary word given a single word is given by the softmax function as follows:

$$p(w_o|w_j) = \frac{exp(\mathbf{u}_{o'})}{\sum_{j'=1}^{v} exp(u_{j'})}, \tag{2.3}$$

where $w_o$ denotes an arbitrary word from the vocabulary, $w_j$ is the current word, and $u_{j'}$ is given by 2.1.

### 2.2.2 Multi-word context

Better embeddings may be achieved by considering context windows composed of multiple words instead of a single one. Figure 2.2 illustrates some examples. Now consider that the central word is immersed in a context of $C$ other words, where

$c = \frac{C}{2}$. As the output word $o$ lies within the target word's context, it can be denoted by $i + o$, with $o$ being the number of positions ahead or behind the center word, and $i$ is the position of the center word.

The idea here is the same as before: starting the weights of the word embedding randomly and updating them based on how they interact with nearby words, using an optimization algorithm. For the sake of simplicity, such iterations will be simply evaluated by quantifying the pairwise associations (*i.e.* between the word and those integrating the context window), in terms of the likelihood that both words might co-occur. Therefore, the model's task is calculating the probability of the occurrence of a surrounding word $w_{i+o}$ given a word $w_i$, denoted as $p(w_{i+o}|w_i)$. Once all the words in the context of a given center word are defined by the length of the fixed window, the goal is to maximize the likelihood of this set of context words given the center word. Assuming a multiple independence between the multiple pairs of central and context words, the likelihood function to be maximized is given by:

$$L(\theta) = \prod_{i=1}^{C} \prod_{\substack{-c \leq o \leq c \\ o \neq 0}} P(w_{i+o}|w_i). \tag{2.4}$$

The same equations previously presented also apply for the multi-word case. The difference is that now 2.1 is calculated for each surrounding word, and Eq. 2.3 is applied using a sliding window.

### 2.2.3 Word2Vec: CBOW and Skipgram

The explanation so far constitutes a simplified version of the architecture proposed in the seminal work of MIKOLOV *et al.* [6], popularly known by the alias of Word2Vec. In the original proposal, it operates in a multi-word setting, which in turn has 2 variations: Continuous Bag of Words (CBOW), whose rationale is illustrated by the left-side of Figure 2.3; and Skipgram, on the right. The main difference between them is that in CBOW, the task performed consists in predicting the target word based on the context, whilst in Skipgram the context words are predicted based on the central word. A more detailed explanation will be provided for Skipgram, since the embeddings exploited in this work will be based on it, and it has consolidated

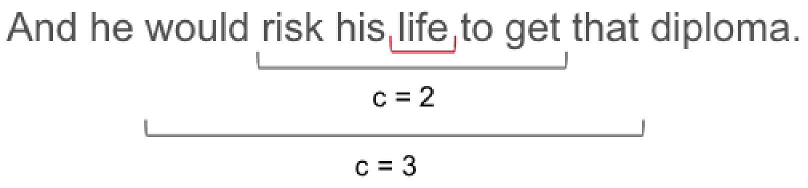


Figure 2.2: Illustration of a word context window of sizes 2 and 3.

itself in the literature as a more effective approach for a wide range of experiments.

Remember that our previous modelling approach constitutes of calculating the probability of the co-occurrence of the output word $o$ with the center word $i$, considering $-c \leq o \leq c$. In order to obtain the update equations for the parameters of this model, we need to maximize Eq. 2.3. The loss function $E$ associated with the Skipgram is given by:

$$\begin{aligned} E &= -\log \prod_{o=1}^{C} \frac{exp(u'_{o_c})}{\sum_{j'=1}^{V} exp(u_{j'})} \\ &= -\sum_{o=1}^{C} u'_{o_c} + C \log \sum_{j'=1}^{V} \exp u_{j'}, \end{aligned} \tag{2.5}$$

where $o_c$ represents the index of the $c$-th context word in the current window. This algorithm updates the word vectors for each element in the context, but the final representation of the word reflects its neighbors, since it is influenced by all of them during the optimization process. If interested in further details on how Word2Vec equations are derived, the reader is referred to RONG [7].

Eq. 2.3 also represents the output of the softmax function in Word2Vec, explored in the vanilla implementation of the Skipgram embedding. There are also some tricks that may be explored to improve this model scalability: the hierarchical softmax and negative sampling. The hierarchical version of softmax mitigates the expensive number of operations involved in computing the denominator of Eq. 2.5, with a computing mechanism based on binary trees. In such trees, the leaves represent words and the calculation of a word probability is decomposed into a sequence of probability calculations. This saves the algorithm from having to calculate the expensive normalization factor over all dictionary words.

Another alternative is the negative sampling. Considering a context window where the words $A$ and $B$ are being compared, the algorithm's goal is to approximate or move away their word vectors into a multidimensional space, and for this it has to consider the relation of the target word with other words. Usually all the other words in the vocabulary are used as spurious samples, which is computationally expensive, but negative sampling provides a method that samples only a subset of these words. The output word (the most probable word) is kept in the sample and gets updated, and the negative samples explored in this case are drawn following a probabilistic distribution that can be arbitrarily chosen. This distribution is called noise distribution, and to make sure that a good sample will be selected, an empirical approach is proposed by the authors. The result is a simplified training objective that is capable of producing high-quality embeddings [6]. The probabilistic distribution used is based on a posterior multinomial distribution [7]. Negative

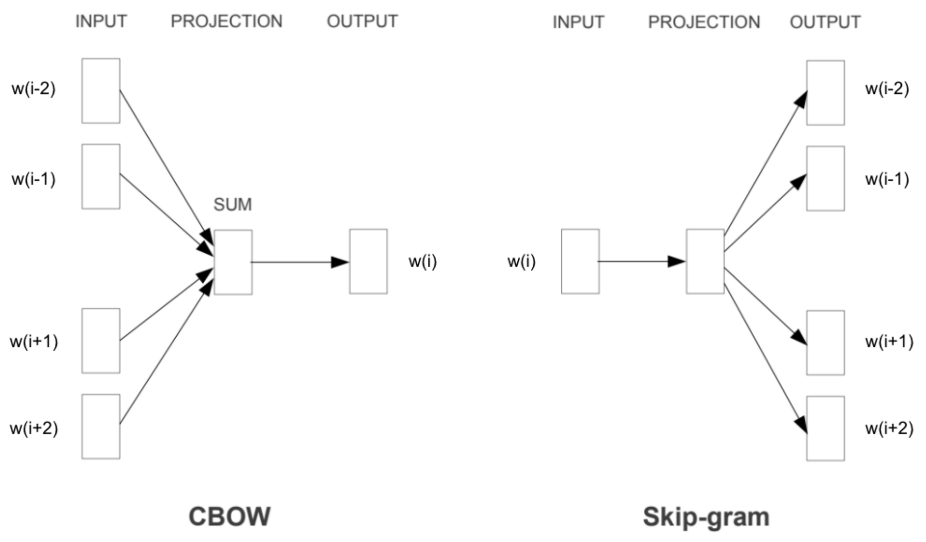


Figure 2.3: Multi-word mechanisms CBOW and Skipgram.

sampling and hierarchical softmax were crucial techniques that enabled a huge number of practical NLP applications explore the Word2Vec, as well as also attracting the attention of many other Machine Learning branches.

## 2.3 Machine Translation and its challenges

Some aspects of the human language seem to be universal, or statistically universal, since only a subset of the languages created in the whole mankind's history is known in the 21th century. When speaking about language diversity, only widely spoken and recently created languages are often considered. Many of these languages employ the use of verbs and nouns, and specific words to refer to animals, emotions, attitudes, and other aspects. Besides, there is a number of differences between them. They can be idiosyncratic or systematic for instance. Idiosyncratic accounts for which words are usually combined and how common they sound together, since different languages are usually biased to distinct sets of words. Systematic represents the word order chosen, for example if the verb is usually put before the direct object or not. The study of systematic differences is called linguistic typology. These and other challenges imposed by language nuances discussed in this section are inspired by the chapter over Neural Machine Translation from the unfinished book of JURAFSKY and MARTIN [8], shared as a draft.

### 2.3.1 Word order typology

Languages may differ in the way they order verbs, subjects, and objects in simple declarative clauses. Some are considered SVO (subject-verb-object) , i.e., their sentences start with the subject, the verb is put in the middle and it ends with an object. Portuguese, Chinese and English are SVO languages, whereas German and Japanese are SOV (subject-object-verb). There are also VSO (verb-subject-object) languages, such as Arabic and Irish.

One of the most challenging scenarios for an NMT model to operate is when translating from a language with one typology type to another, which luckily isn't the case in this work, as both languages are SVO.

### 2.3.2 Morphological typology

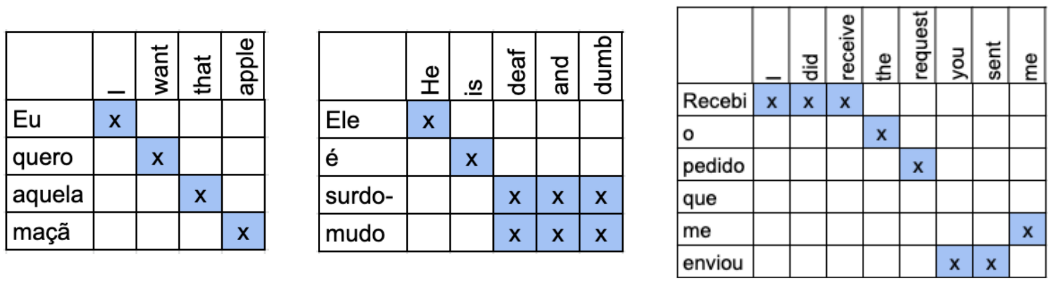
Regarding typology, languages can be categorized in 2 dimensions. The first is the number of morphemes per word: isolating languages like Vietnamese usually contain one morpheme (one enclosed meaning or reference to something) per word. They are simpler to deal with than polysynthetic languages, which may contain a variable number of morphemes per word. This classification isn't binary, though. Portuguese and English are somewhere in between both, but closer to isolating rather than polysynthetic. Both languages can add suffixes or prefixes that change the meaning: for instance happy is an adjective, but when united with "ness" becomes a noun "happiness".

The second dimension refers to which degree the morphemes are segmentable. Languages can be agglutinative (clear boundaries between morphemes) such as Turkish; or fusional, such as Spanish, where the boundaries between the morphemes are unclear or lost. One example for this is the verbal inflection "comí" (I ate), where the morphemes "comer" and "yo" have lost their boundaries between each other. Portuguese and English are considered to be fusional, but not as much as other fusional benchmarks such as Classical Hebrew.

### 2.3.3 Word alignment

Alignment in the context of translating means how word correspondences are made between the reference (source language) and the corresponding translation (target language). The aforementioned typological properties substantially increase the complexity of the alignment process, specially for situations where distinct typologies must be matched.

Not all words in the source or target language necessarily need to be aligned, sometimes the role of one word in a translation pair is just to maintain fluency, contributing to a correct sentence in terms of grammar. Figure 2.4 contains 3 alignment examples, wherein the source language is on the left and the target one is on top. From left to right, the first rectangle represents a simple one-to-one scenario; the second shows a many-to-many relation and the third, one-to-many, all



| | I | want | that | apple |
|---|---|---|---|---|
| Eu | x | | | |
| quero | | x | | |
| aquela | | | x | |
| maçã | | | | x |

| | He | is | deaf | and | dumb |
|---|---|---|---|---|---|
| Ele | x | | | | |
| é | | x | | | |
| surdo- | | | x | x | x |
| mudo | | | x | x | x |

| | I | did | receive | the | request | you | sent | me |
|---|---|---|---|---|---|---|---|---|
| Recebi | x | x | x | | | | | |
| o | | | | x | | | | |
| pedido | | | | | x | | | |
| que | | | | | | | | |
| me | | | | | | | | x |
| enviou | | | | | | x | x | |

Figure 2.4: One-to-one, many-to-many and one-to-many alignments using Portuguese and English sentences

of them considering the Portuguese-English pair. Note that the last scenario has one spurious word in the source language which does not align with any word of the target translation.

### 2.3.4 Lexical divergences

Sometimes, languages may contain a specific expression or word that cannot be accurately translated to another language without losing some of its meaning. Whenever this happens, this word represents a lexical gap. This is pretty common with slang words, which have a very specific use case, making it harder to express the same meaning in another language, even when using several words.

Another language characteristic that may increase the chance of happening a lexical gap is the fact that some of them are verb-framed. This means that they mark the path of motion in the verb and leave the satellites to express the manner of motion. The path of motion for instance means the direction of the movement, such as move into, out of or across, whilst the manner of motion refers to a specific type of motion, such as running, walking, or crawling.

On the opposite side of verb-framed, there are languages that are satellite-framed, which means that the manner of motion is expressed in the verb whilst the path of motion is expressed in the satellite. As happened with other language characteristics, these classifications don't mean that a given language cannot mix verb-framed and satellite-framed expressions. English and Portuguese for instance can generate expressions using both styles.

Consider the translation of language A that mainly uses verb-framed sentences to a language B, which mainly uses satellite-framed ones. The difference in the framing pattern between A and B languages does not necessarily mean that the chance of lexical gaps increases. It means that A may have a range of specificities by aggregating verbs that may be complex to reproduce with the correspondent aggregation of satellites in B. This diversity might pose a challenge to generate an accurate translation, and it may result in a partial loss of the original meaning.

### 2.3.5 Qualitative aspects to evaluate translations

Defining qualitative and quantitative characteristics for accurately evaluating how successful was a translation is hard. The quantitative matter will be further discussed in an another section, but the qualitative side is usually broken into 2 dimensions, namely adequacy and fluency.

Adequacy represents how well the meaning of the source sentence was captured, and it is sometimes referred to as fidelity. Synonyms hardly ever affect this dimension, and the complexity of correctly evaluating this dimension is what mostly hinders the development of related quantitative metrics. Fluency means how correct and fluid the translation is in the target language. Thus, it involves evaluating the sentences in categories such as grammaticality, readability, and naturality of the words chosen.

### 2.3.6 Ambiguity

The challenge of identifying and properly representing ambiguity is among one of the most complex for natural languages. This happens because the model heavily depends on the diversity and quality of the dataset considered for its development, since it should be able to assign distinct features to different meanings. Ambiguity can be found at the word-level, for instance, the word sink can be a noun or a verb, which is known as lexical ambiguity. It can also occur for a whole sentence or a clause, what is called as structural ambiguity. Models inherently do not possess knowledge of the whole world to identify in which situations ambiguity may occur. If one could provide such knowledge to a model beforehand, it should be capable of handling such situations, but no solution so far has proved to be robust enough to achieve this.

A plausible critic for the effectiveness of the vector structure derived by MIKOLOV *et al.* [6] is that it creates a single vector representation for words that may contain multiple meanings, hence capturing only the most frequent cases in the training set. It lacks of flexibility to deal with occurrences of polysemy or homonymy in a given context. The word pike is a classic case of homonymy, also used to explain ambiguity in the course material of MANNING [9], since it can have any of the following meanings:

1. A sharp edge or staff
2. To piece or kill with a pike
3. A type of elongated fish
4. A railroad line or system

5. One type of road

6. To make one's way (pike along)

This word may also admit additional meanings, based on the speaker location. Polysemy accounts for the coexistence of many possible meanings for a word or phrase, a phenomenon that happens in either Portuguese or English. In addition to the challenges of polysemy and homonymy, Portuguese provides some specific challenges such as distinct pronoun organizations carrying the same meaning, and some others already mentioned in a previous section (1.2).

### 2.3.7 The open vocabulary challenge

NMT models typically learn from supervised data, hence they are constrained to a fixed vocabulary. When dealing with an unseen word in some test scenario, traditional word-level models produce an $< unk >$ (unknown) token. This phenomenon is referred to as out-of-vocabulary (OOV).

There is a number of ways to deal with the open vocabulary issue. Speaking of word-level models, the only way to reduce the chance for a model to produce $< unk >$ tokens is to guarantee that the training set has enough words to cover those that might appear in the evaluation phase. One conclusion that can be derived from this statement is the following: the bigger the number of words a dataset has, the bigger the challenge of addressing open vocabulary is. Another issue to outline is that as higher is the complexity and diversity of the domains present in a text, higher will be the probability of existing more words in it. One of the datasets used in this work corroborate with this hypothesis.

If the word-level restriction is removed, then subword models can increase the occurrence of smaller tokens, since they are capable of breaking the words into smaller units. These units can be characters (at the character-level), syllables or prefixes (at the subword-level). By acting upon smaller pieces of a word, the model incorporates the ability of generating custom tokens, since any combination of these pieces become possible when generating translations.

### 2.3.8 Addressing MT challenges

This chapter performed an overview of the qualitative theory that is intrinsic to the study of natural languages, which is relevant to the context of Machine Translation. It sheds some light on the practical differences that one language may have as compared to another, as well as it helps to understand some of the challenges that MT researchers have to face when creating systems based on expert knowledge. Modelling all these nuances was a complex task, since many of the first MT systems were rule-based or phrase-based, belonging to a MT branch called Statistical Machine Translation. Therefore, the development of such systems required that specific language phenomena have to be compiled in a set of rules unique to each language pair, like tables of equivalent expressions, thus requiring a heavy manual effort to maintain and update these systems. The alignment, for instance, used to be treated using latent variables. This hugely used to increase the complexity of the translation task, since the goal has switched from predicting $P(x|y)$ to $P(x, a|y)$, where $a$ is the alignment variable.

Neural systems represent a breakthrough in this area, since they removed the manual work of modelling language nuances, learning them on their own. This came along with the flexibility to adapt their architectures according to the available resources and performance target. Among other advantages of NMT are the better use of context and phrase similarities, enhanced fluency of the translations and the necessity of optimizing only a single network end-to-end. NMT has also some disadvantages: the neural networks are known for being a black-box algorithm very hard to interpret and debug. It is also difficult to control possible model biases (which can be gender or race related), strongly depending on the dataset used as well as the process of learning as a whole.

Despite these disadvantages, the solid associated benefits of this approach have motivated a strong investment in this research branchby the NLP community, leading to a big increase on the number of publications of the field since 2014. Further technical details and practical behaviour of neural networks will be discussed in the following chapter.

# Chapter 3

# Neural networks and Machine Translation

Since their ideation in 1958, Neural Networks have seen peaks and valleys of research interest in the Machine Learning field. They have gone out of the tar pit (term borrowed from [10] that originally mentioned learnings of "the dark times" in Software Engineering) in the eighties with feedforward and recurrent variants, and evolved with contributions such as the convolutional and LSTM (Long Short-Term Memory) variants in nineties. After such progress, they started to leverage promising results when applied to numerical and categorical data. Specially after 2013, Neural Networks are growing at an unseen rate, driven by the widespread use of Deep Networks. Applications in image processing and natural language areas were massively explored and soon consolidated themselves as the state-of-the-art approaches.

Before going into further details of the pioneer research that applied Neural Networks to solve the Machine Translation problem the task of translating from one language to another will be formally explained. Even though are a few alternative approaches that may reach the same result, we will focus the core of the explanations on the neural alternative.

## 3.1 Machine translation as a task

The ability to use an automated procedure to translate from a source language to a target language had precursors since the 1930s, with the use of mechanical dictionaries, later coming up with some pioneer applications in the 1950s [11]. These applications were heavily funded by the military sector and were focused on Russian-English translation or the other way around, mostly motivated by the rivality between the United States and the Soviet Union in the Cold-war period. Roughly, the translation task can be formulated as the following optimization problem:

$$\hat{y} = argmax_y P(y|x), \tag{3.1}$$

where $x$ is the representation of a sentence in the source language, $y$ represents another sentence in the target language, and $P$ is a probability value. Note that $P$ includes parameters to be trained using supervised learning. After the training has been completed, these parameters will represent the model's word representation knowledge for a vocabulary previously seen. Based on this prior knowledge, the model will generate its guess $\hat{y}$ for the sentence $x$. Using the Bayes' theorem to decompose $P(y|x)$, it follows that:

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}. \tag{3.2}$$

Since the denominator is independent of $y$, finding $\hat{y}$ is the same as to make the product $P(y)P(x|y)$ as large as possible. The following main components derived from this process are the numerical representations of different challenges for the translation process: $P(x|y)$ and $P(y)$.

The probability indicated by $P(x|y)$ represents the translation model part, which is basically a mapping of which translation matches with the source sentence, considering what the model has learned with the training set. Accordingly, $P(y)$ can be interpreted as a language model, basically responding for how fluently some meaning is reproduced in a set of words. To accurately reach a high probability for $P(x|y)$, a large amount of high-quality parallel data is required, whereas the other component $P(y)$ requires only proficiency under the scope of a single language (the target one).

Another approach to address this problem is to narrow the scope of the translation task by breaking it into more steps. This may be achieved, for instance, by introducing a latent variable $a$ that connects the source and target clauses and words. This variable is the mathematical representation of the alignment model described in the previous Section 2.3.3, and it isn't explicitly specified in a dataset a priori. The alignment model basically connects particular words between both languages.

The process of obtaining an efficient representation for $\hat{y}$ (the most probable translation guess) is performed with the help of a decoding algorithm. Decoding isn't a trivial task as it requires a robust algorithm to deal with both the translation and language models. Matching the reference with the use of an algorithm is hard since human translators can reword a phrase with synonyms, reorder and rearrange words, replace single words with multi-word phrases, and vice versa. Decoding has also been proven to be a NP-hard problem, even in relatively simple translation models [12]. The task of reaching an adequate degree of correspondence between source and target sentence parts is aggravated by the Machine Translation challenges briefly discussed in Section 2.3.

When decoding the source sentence, there are some special tokens used to represent the start of the sentence and the end of the sentence (<sos> and <eos>). The decoding process always starts with the <sos> token, then each word from the source sentence is analyzed and the model tries to generate its correspondent word in the target language. This process only ends when the <eos> is produced or the maximum sentence length in words is reached. When the model does not have a clear translation for a source word based on its current knowledge, it may output the token <*unk*>, which stands for unknown.

In this section a formal definition of the Machine Translation task was provided, as well as the qualitative challenges for this task were addressed. The reader now may have a clearer picture of the challenges faced by the first neural algorithms applied to solve this task. The sections in the following will show how the first algorithms addressed these questions and their main flaws and strengths.

## 3.2 The rise of Neural Machine Translation

For a long time, neural networks haven't seen any applications involving translation tasks, until 2013 when CHO *et al.* [13] came up with the RNN (Recurrent Neural Network) Encoder-Decoder architecture. At a time where statistical models were the main alternative for Machine Translation, the adoption and later maturity of RNN algorithms gradually started to prevail in the published papers.

There are some interesting properties of the recurrent neurons that justify their application to solve the Machine Translation task. The next section clarifies them to the reader by going through equations and properties of this neuron.

### 3.2.1 A brief introduction to the recurrent neuron

Traditional feedforward neurons treat the data points over which they are trained on as independent instances. This means that switching the order of them within a same batch does not affect the final update of network parameters. Conversely, recurrent neurons contain an interesting property that extended the scope of applications successfully addressed by a Neural Network: their outputs are dependent on the previously presented samples; therefore, switching the order of inputs leads to distinct results.

Basically, a RNN neuron contains a hidden state and the concatenation of all hidden state vector. This vector, conjugated with the current input, defines the value of the hidden state vector for the next network iteration. This is shown

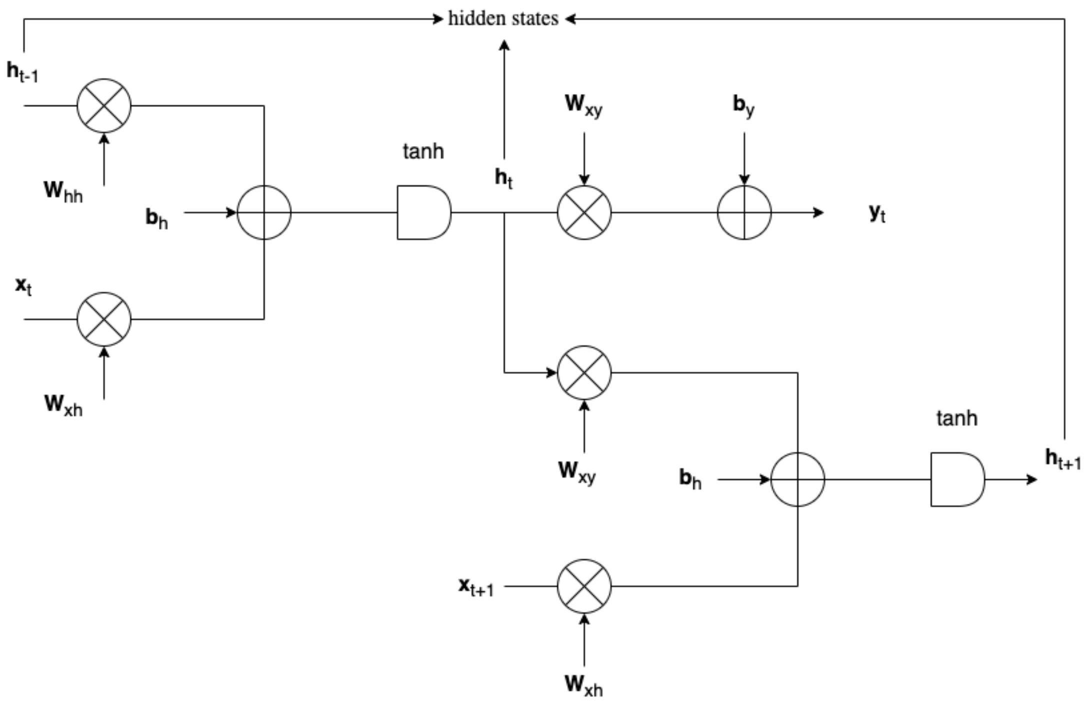


Figure 3.1: Computational graph of a RNN (adapted from SHERSTINSKY [14]).

by the current neuron output after going through some activation function. The computational graph of a RNN is illustrated in Figure 3.1. In this graph, $\mathbf{W}_{hh}$ is a weight matrix that defines the contribution of the previous hidden state into the current one; $\mathbf{W}_{xh}$ is a matrix that maps the contribution of the input $\mathbf{x}_t$ to the current hidden state $\mathbf{h}_t$ and $\mathbf{W}_{hy}$ defines how the current hidden state must be reflected into the neuron output. At each addition operation in this graph, there is a correspondent bias vector summed to produce the final outcome.

This schema is responsible for introducing the concept of "memory" to RNNs. Note that this feedback loop can be "unrolled" to evidence the network dependence on previous inputs (considering different timestamps for instance) and hidden state values. For computing convenience, in order to perform backpropagation, we must limit the depth of this unrolling procedure, propagating the gradients from the last output from the unrolled graph backwards up to this point. Following this procedure, the resulting unrolled network turns analogous to a simple feedforward layer network, easing the related computing procedures.

The order-dependence property of a RNN is an extremelly desired characteristic for processing Word Embeddings, as previously discussed in Section 2.2, since the neighbouring words often affect or contribute to the meaning representation of a given word in the sentence.

### 3.2.2 The first RNN-based machine translator

According to the Cho et al's approach, one RNN sequentially processes each symbol of a sequence of inputs represented by vectors $\mathbf{x}_t$, that represent symbols (typically words or token Embeddings), $t \in \mathbb{N}$, encoding it into a fixed-length vector representation defined by the encoder hidden state. The equation that abstracts the encoder hidden state computation at a given timestamp $t$ is

$$\mathbf{h}_t^e = f(\mathbf{x}_t, \mathbf{h}_{t-1}^e), \tag{3.3}$$

where $f$ represents some nonlinear mapping function implemented by the RNN. For a variable-length sequence with size $T_x$ defined by $\mathcal{S} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{T_x}\}$, the information delivered by the encoder to the decoder corresponds to the last hidden state obtained with the Eq. 3.3, which in turn is called as the context vector $\mathbf{c}$.

The second RNN has the task of decoding this representation into a sequence of symbols by predicting the next symbol $y_t$ given the context vector $\mathbf{c}$ plus all the previously predicted words $\{y_1, \ldots, y_{t-1}\}$. In other words, it is possible to compute the probability over a given translation $\mathbf{y}$ according to the following decomposition of the joint probability associated with $\mathbf{y}$:

$$p(\mathbf{y}) = \prod_{t=1}^{T_y} p(\mathbf{y}_t | \{\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}\}, \mathbf{c}), \tag{3.4}$$

where $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_{T_y})$ (the sequence in the target idiom) and $\mathbf{c}$ is the previously mentioned context vector. The iterative equation for defining the decoder's hidden state at the time $t$ resembles the one from the encoder as follows

$$\mathbf{h}_t^d = g(y_{t-1}, \mathbf{h}_{t-1}^d, \mathbf{c}), \tag{3.5}$$

with $g$ representing the function implemented by the decoder RNN. Based on $\mathbf{h}_t^d$, the $t$th translated word is calculated using

$$y_t = \text{softmax}(f(\mathbf{h}_t^d)), \tag{3.6}$$

where $f$ represents one or more feedforward network layers. These equations provide a holistic view of the first encoder-decoder RNN-based translator. In this setup, the encoder and the decoder are jointly trained targeting to maximize the conditional log-likelihood of a target word given the sequence of words of some source sentence and the previous translated words.

The drawback of this architecture is that some of the information available at the encoder RNN is lost, since only the last hidden state is used, whilst other hidden state values are discarded. Another issue relates to the fact that the encoder has to compress all the relevant sentence source information into a fixed-length vector. This increases the challenge of dealing with long sentences, which is already complex for any RNN neuron due to the vanishing gradient effect [14].

## 3.3 NMT by jointly learning to align and translate

To address the fixed-length and "information compression loss" issues identified in the previous implementation, the authors of the first paper proposed another architecture that changes how the encoder is connected to the decoder in BAHDANAU *et al.* [15]. Now, the context vector provided by the encoder to the decoder is multiplied by a weighted "attention" function and integrates multiple hidden-states, not just the last one, defining a variable-length context vector such that $\mathbf{c} = \{h^e_1, \dots, h^e_{T_x}\}$.

In the following sections, we will briefly discuss features and issues related to the main contribution of this paper: the attention mechanism. First, this process will be described using high-level illustrated steps to help the reader in grasping the intuition. Then, in the sequence, we will show the equations related to them.

### 3.3.1 Some intuition behind the attention mechanism

Roughly, the attention mechanism simply assigns weights to word embeddings, allowing a straightforward implementation of the alignment model presented in Section 2.3.3. Essentially, these weights define how much attention must the decoder pay for a specific word from the encoder sequence when guessing the next word of the translated sentence.

Consider the translation of the Portuguese sentence "Vou às dez." to English. In Figure 3.2 at the step 1, the rectangles resembling a traffic light represent the RNN neuron at a given sentence iteration, where the circles correspond to the components of the hidden state vector (see 3.1 to remind its dynamics). Note that each neuron is fed with the hidden state from the previous iteration plus the current input. The attention scores are computed by the dot product between the embedding (produced by the decoder RNN) of the token *Start-Of-Sequence* ("SOS") (written as "<sos>") that must be feed to the decoder inputs for starting the generation of the translated sequence. This process defines a particular weight for each part of the encoder sequence to compose the context vector used by the decoder, conjugated with it's current input, when defining the next word of the translated sequence. Naturally, the attention weights are recalculated at every step of the translation process, i.e., this process is repeated for every next token generated in the output or translated sequence. For computing the attention scores, a softmax layer is commonly used, as discussed further later, and the weights attributed in this case are represented by the green bars on the top right of Figure 3.2 at step 2.

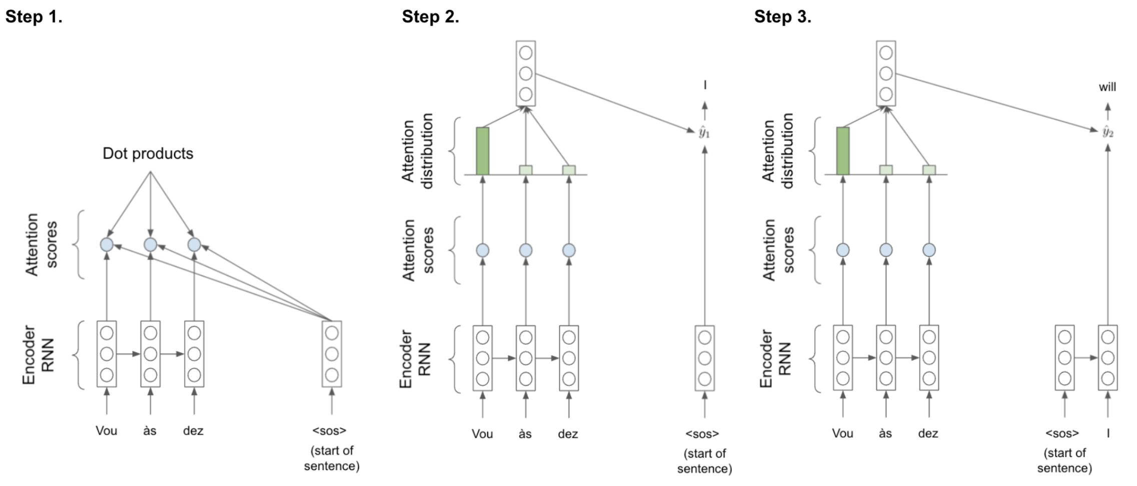


Figure 3.2: The first steps of how the attention mechanism acts when translating an example (adapted from MANNING [16]).

Since the model has already been trained on a supervised dataset, it presumedly knows that the word "Vou" must be translated to one or a given set of words in English. In this particular case, when starting with the token "<sos>", the subject of this verb is represented by the personal pronoun "I".

The translation task then keeps on producing the next words based on the inputs, generating the word "will" after "I" to complement the meaning of the original word "Vou" at step 3.

Note that the attention weight distribution changes for every new step. In Figure 3.3 at step 4, the decoder has already generated the sentence fragment "I will go", corresponding to the original "Vou". Thus, the embedding associated with the token/word "às" becomes the focus.

In step 5, the other words or tokens related to "ten" and "." are skipped to illustrate how the algorithm ends. When the weights are considered to be low in absolute value with respect to the inputs, it may be a signal that the algorithm has converged. The translation process ends when the <eos> token is produced.

Now that the dynamics of the attention mechanism during the translation is clear, a mathematical walkthrough on the attention layer must be performed.

### 3.3.2 Attention: Calculus background

Consider the new variable-length context vector defined by a linear combination of several hidden states of the encoder, in opposition to only the last one from the first RNN-based translator model. Now, the context vector will vary according to

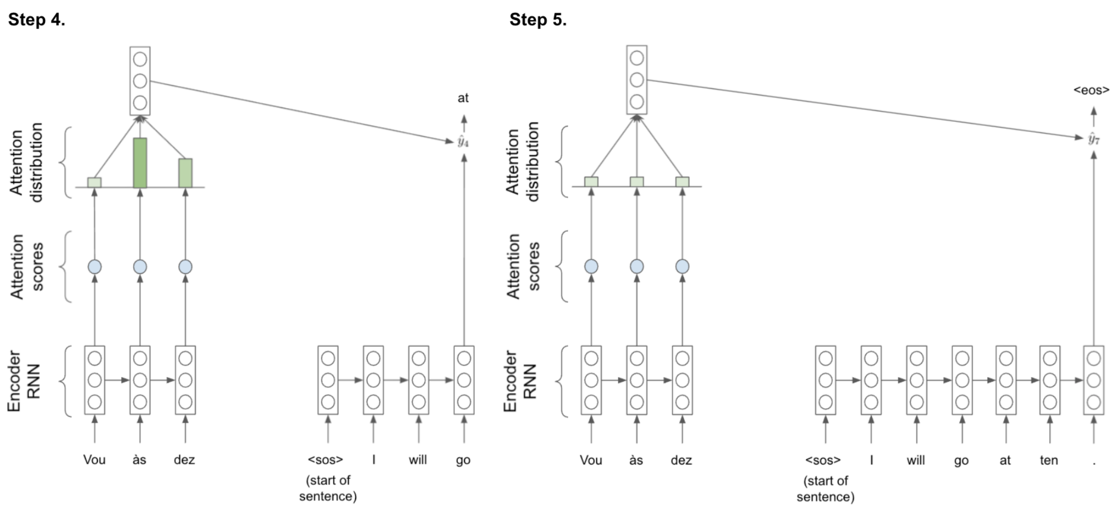


Figure 3.3: The last 2 steps of attention during the translation of an example sentence (Adapted from MANNING [16]).

every input token from the encoder. Let us interpret such words as a set of vector annotations $\{\mathbf{h}^e_1, \mathbf{h}^e_2, \ldots, \mathbf{h}^e_{T_x}\}$. The context vector after applying attention will be computed by a weighted sum of such annotations $\mathbf{h}^e_i$ at a given time $i$ as follows:

$$\mathbf{c}_i = \sum_{j=1}^{T_x} \alpha_{ij} \mathbf{h}^e_j, \tag{3.7}$$

where the variables $\alpha_{ij} \geq 0$ (or attention weights) represent a measure of how relevant is each input annotation when composing $\mathbf{c}_i$ for a given token currently processed at the decoder. Such weights are also known as attention coefficients, and represent the probability that a target token $y_i$ is aligned to, or translated from, a source token $x_j$. These weights shed a light on how the model is aligning and relating words between the source and target language, helping it to address the challenge stated at Section 2.3.3. The attention coefficients can be computed in many forms but a common approach is the following

$$\alpha_{ij} = \frac{\exp\left(a(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j)\right)}{\sum_{k=1}^{T_x} \exp\left(a(\mathbf{h}^d_{i-1}, \mathbf{h}^e_k)\right)}, \tag{3.8}$$

where $a$ is an alignment model that relates the embedding $\mathbf{h}^e_j$ referent to the encoder position $j$ with the embedding of $\mathbf{h}^d_i$ of the decoder at the position $i$. The variable $k$ iterates over the entire range of the input sentence $T_x$. The alignment scores can also be interpreted as a way to distribute attention over the input, and $a$ is often parameterized by a feedforward neural network jointly trained with the encoder and decoder parts. The authors have also stressed that the attention mechanism relieves the encoder from the burden of having to encode all the information from the source sentence, hence contributing to the resilience of this approach when dealing with long sentences.

Another improvement in this work that contributed to the success of this approach is the use of bidirectional RNNs to encode the input sequence. The key difference from this to the unidirectional RNN is that the context vector now aggregates information relative to the representation of not only the preceding words but also from the following words, allowing a more efficient attention weight distribution. This vector is simply defined by a concatenation of the representations obtained from the forward and the backward RNN hidden states. Despite these improvements, the attention mechanism is still rather simple, since it is based only on an additive attention, whilst later variants compose additive with multiplicative attention and even statistical distributions to define the $\alpha$ factors [17]. Due to this gap, the importance attribution isn't performed as well as in its variants.

Another detail that contributed to a better model performance despite not clearly highlighted in the paper (it is only mentioned) is the use of a beam search decoding algorithm. This algorithm was first proposed as a generic sequence transducer [18], but only a few years later was exploited for MT [19]. The main benefit brought by it is a more efficient and accurate search of the most likely next word given a set of preceding words. In other words, it reduces the chance of the algorithm to get trapped in local optimal solutions compared to the Greedy Search algorithm, thus allowing the NMT model to explore more translation variants and converging to more realistic translations.

### 3.3.3 Search algorithms for NMT

After converging to a local optimum and learning vector representations that reflect semantical and syntactical attributes of the words, the model can be tested and generate translations. When accomplishing the task of generating translations, the quality of the search performed and the number of possibilities considered by the algorithm make a difference on the final result. The traditional approach and a more clever one will be discussed here: the greedy search and the beam search, respectively.

#### Greedy Search

This was the approach used for the first RNN-based translator described in Section 3.2.2. It may be considered as the simplest way to generate a translation given the context, as it predicts the most likely word by considering a given set of previous words. This rather simple search is used since the beginning of the field of Statistical Machine Translation. The goal is to obtain the next $\hat{y}_t$ iteratively such that

$$\hat{y}_t = argmax_{y_t} P(x_t|y_{t-1})P(y_{t-1}), \tag{3.9}$$

in each timestep $t$, considering $x_t$ as the representation of a sentence in the source language and $y_{t-1}$ is the current composition of the target sentence built until the timestamp $t$. This algorithm has the drawback of being impossible to go back after generating an unnatural or sub-optimal sentence. Basically, a token that looks good to the decoder at a given moment might turn out later to have been a wrong choice. When this happens, the model gets limited to generating new words on top of the current sub-optimal sentence, hence the translation loses fluency. Nonetheless, it was the top of mind algorithm for the MT community for a long time.

Ideally, the only method that guarantees to find the best translation is the exhaustive search. Given a vocabulary with size $V$, the exhaustive search would be performed by computing and evaluating each one of the $V^T$ word combinations, with $T$ being the length of the sentence translated in words. Going over such a large word search is obviously too expensive. Hopefully, a more cost-effective search variant that considers more words than the greedy variant and that is less complex than the exhaustive one was created.

### Beam Search

In this algorithm, instead of choosing the best token, $k$ possible tokens are kept at each step. This fixed-size parameter is called as the beam width. It works by computing a probability of each vocabulary word to be appended to the current translated sequence, for each of the $k$ available candidates, wherein this probability is obtained through a softmax calculated over the entire vocabulary. These numbers are then sorted from the most probable to the least, and only the $k$ best possible tokens are retained for the succeeding step. Such candidate tokens are called hypotheses, and this process of calculating probabilities, sorting the sentences and progressively adding words at the end of the sentence at each step means that the algorithm is simultaneously evaluating a set of $k$ different sentences.

A hypothesis only stops being evaluated if it ends with an <eos> token, or if it is not within the top $k$ options. If an $< eos >$ token is reached, the sentence is placed aside and other hypothesis continue to be explored. This process is repeated until all the hypothesis reach an $< eos >$ token or the maximum predetermined sentence length (in words). Figure 3.4 shows an example for a small sentence using beam width of size 2, where "arrived" and "the" are the first guesses, later producing 4 words where only the 2 most probable were selected. After picking "green" and "witch" by sorting probabilities in a descending order, another 4 words are generated, and the process is repeated until convergence.

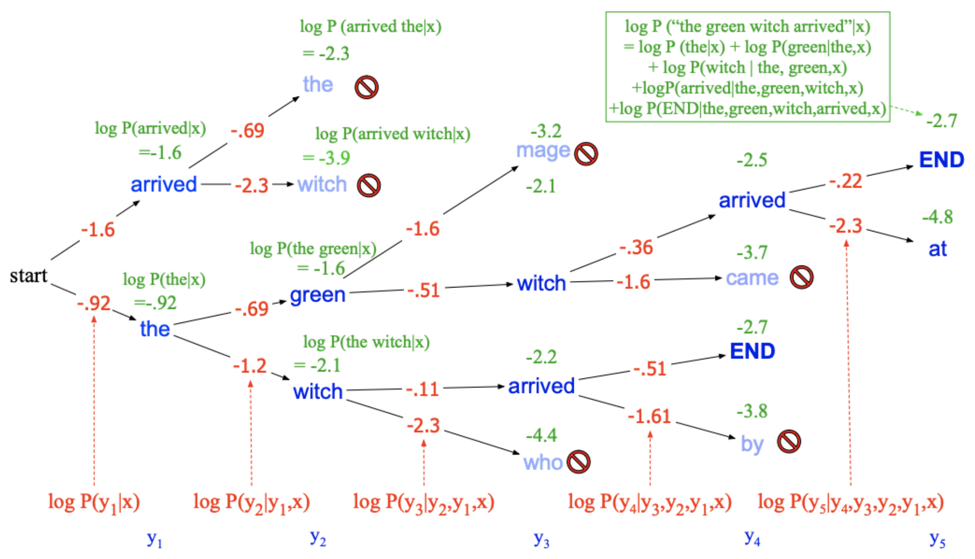


Figure 3.4: Example of beam search with beam width of size 2, taken from JURAFSKY and MARTIN [8].

Besides the practical gains regarding fluency, this algorithm has the drawback of penalizing longer sentences with lower scores. Moreover, this issue can be partially addressed by a length normalization factor, but not totally mitigated. Beam search has become the standard approach in NMT, with a good starting point being the use of a beam width of size 3, the same adopted in our experiments.

### 3.3.4 Exploring RNN-based Sequence to Sequence architectures

An interesting work that illustrates the potential of the previously presented RNN models is due to BRITZ *et al.* [20]. They use [13] as a base model and evaluate the tuning of several hyperparameters, like embedding size, RNN cell variant, i.e., LSTM and GRU (Gated Recurrent Unit), encoder and decoder depth, unidirectional vs. bidirectional RNN encoders, attention mechanism and beam width parameters. One of the claims of this work is that careful hyperparameter tuning can yield better results than exploring architectural variations.

Here are some interesting findings reported by BRITZ *et al.* [20]: (1) larger embeddings consistently outperform smaller ones by a thin margin, (2) LSTM cells consistently outperform GRU cells, (3) deeper decoders tend to lead to some performance increases, (4) additive attention achieves slightly better results than multiplicative attention and (5) beam searches when tuned to a "sweet spot" can increase the model performance up to 5%.

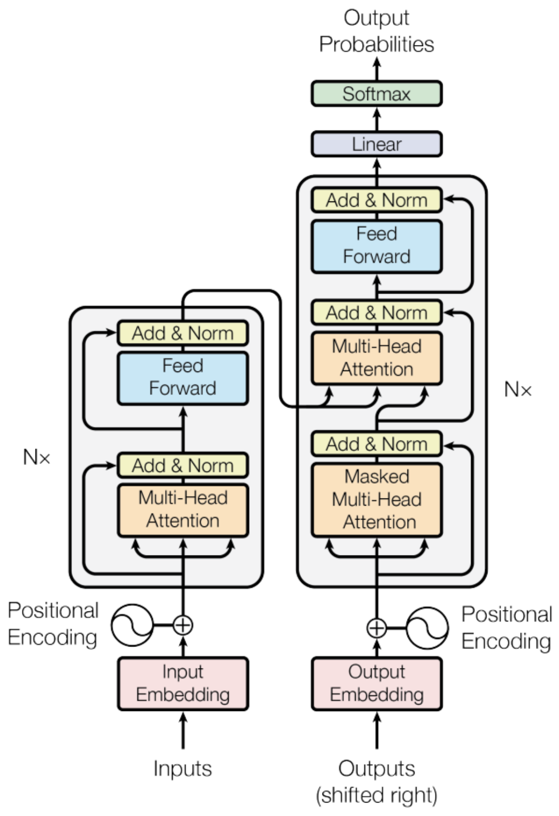


Figure 3.5: Transformer model architecture, taken from the original paper [3].

## 3.4 Transformer models

One contribution switched the focus on recurrent and convolution-based neural network models for NMT in 2017: the Transformer architecture. Introduced in the seminal work of VASWANI *et al.* [3], this model is not subjected to the same issues faced with RNNs. Due to their sequential processing structure, RNNs have to wait for some internal operations to finish before beginning with another one, which is not the case of Transformers. This new approach gained traction and adherence from the community, since its massive parallelization is one of its major strengths.

In the original implementation, the encoder and the decoder exploit 6 stacked Transformer layers. Regarding the encoder, each layer contains a multi-head self-attention mechanism, followed by a layer normalization and a position-wise feed-forward fully connected layer, represented by the "Add & Norm" block depicted in Figure 3.5. A residual connection is also employed around each of the 2 blocks inside a Transformer layer, represented by the arrow that goes into the "Add & Norm" blocks. The structure of the decoder is quite similar, differing by the existence of an extra masked multi-head attention layer, before the multi-head attention block. This mask procedure, combined with an offset in the target sentence (output embedding) by one position, ensures that the predictions for a given position $i$ depend solely on the known inputs for positions less than $i$. Backpropagation operates end-to-end for training this architecture.

### 3.4.1 Unveiling self-attention

The attention mechanism adopted here differs from the previous formulation, but the overall goal is the same: enabling the model to distribute the influences of every input word on the output words generated. Consider a scenario where there are a set of $q$ feature vectors available, with dimensionality $q$ stored in the rows of $\mathbf{V} \in \mathbb{R}^{q \times k}$. The model can dynamically map a feature vector to a key, based on how similar the input query vector $\mathbf{q} \in \mathbb{R}^q$ is to a set of $k$ keys in the columns of $\mathbf{K} \in \mathbb{R}^{q \times k}$. The mapping is performed depending on the level of similarity between the input and the query matrix. If $\mathbf{Q}$ is highly similar to $\mathbf{k}_i$, and has some similarity with $\mathbf{k}_{i+1}$ (respectively, the $i$th and $(i+1)$th column of $\mathbf{K}$), but it is uncorrelated with the rest, the value vectors $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ will be used in the weighted sum explored by the attention mechanism with the highest weight [21].

The query ($\mathbf{Q}$), keys ($\mathbf{K}$) and values ($\mathbf{V}$) are all matrices whose dimensions are model's hyperparameters. Attention's "keys and values" would loosely correspond to hidden state representations learned by the Sequence to Sequence architecture [22], and "queries" will be defined by input token embeddings or intermediary representations of them provided by some immediately previous Transformer layer.

The core of this new mechanism is still pretty much the same as the one used by BAHDANAU *et al.* [15]. The attention weights are defined by an equation similar to Eq. 3.8, whereas the attention coefficients are now given by Eq. 3.10:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{i'} \exp(e_{i'j})}, \tag{3.10}$$

where

$$e_{ij} = \mathbf{q}_j^T \mathbf{k}_i. \tag{3.11}$$

The output of the self-attention mechanism for the $q$th input query is given by:

$$\mathbf{y}_j = \sum_i \alpha_{ij} \mathbf{v}_i. \tag{3.12}$$

#### Scaled dot-product attention

The name scaled dot-product attention comes from the strategy of normalizing the dot products between the query matrix and the key matrix entries, normalized by the factor $\sqrt{d_k}$, before submitting them to a softmax layer to produce the weight factors $\alpha_{ij}$, which will be responsable for linearly combining the vectors stored in $\mathbf{V}$.

This mechanism is illustrated in Figure 3.6, where MatMul represents a matrix multiplication block and the mask is optional, used only in the decoder. Mathematically speaking, we have

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{V} softmax\left(\frac{\mathbf{K}\mathbf{Q}^T}{\sqrt{d_k}}\right). \tag{3.13}$$

One of the major advantages from the dot-product attention is that it is space-efficient. It can be implemented using optimized matrix multiplications; thus, benefiting from parallel architectures as compared to the additive attention, for instance. The scaling factor is a plus from the authors to the original dot-product attention, as they claim that it helps in reducing the magnitude of the dot-products, which may push the softmax function to flat regions, wherein the associated gradients may be small.

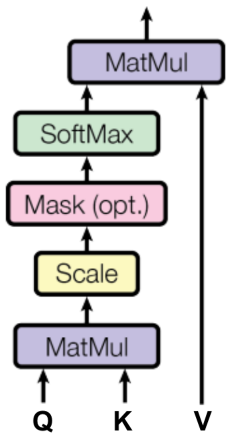


Figure 3.6: Illustration of the Scaled Dot-Product Attention mechanism, taken from VASWANI *et al.* [3].

## Multi-head attention

Essentially, this technique implements the attention mechanism by a concatenation of scaled dot-product attention sub-blocks. The assumption behind it is that different learned projections of the same **Q**, **K**, **V** matrices yield representations that may complement each other, as an ensemble of attention heads, delivering distinct features from the same data. Figure 3.7 illustrates this trick. One may readily note that the outcomes of the of the multiple heads goes through a linear feedforward layer on the top.

Multi-head self-attention also plays a major role on the overall performance of the Transformer. The authors claim that it is beneficial to decompose the original attention matrices **Q**, **K** and **V** in $h$ submatrices, where $h$ is the number of attention heads, thus exploiting different learned projections. The attention weights are adjusted to such different representation subspaces, depending on the query and available keys. The original paper uses $h = 8$ heads. An usual recommendation in this case is that the original hidden size $d_{model}$ must be divisible by the number of attention heads.

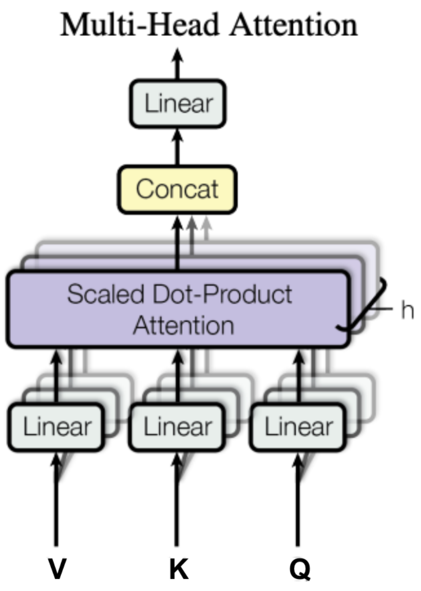


Figure 3.7: Illustration of the Multi Head Attention mechanism, taken from VASWANI *et al.* [3].

At the time of the original paper writing, multi-head self-attention was thought to consistently contribute to the Transformer performance. Years later, after an analysis performed by VOITA *et al.* [23], this hypothesis was rejected. The authors evaluated how each attention head contributes to the solution of a translation task, and found out that only a small subset of the heads would be enough to sustain the Transformer translation scores. Their analysis was based on the importance distribution, a heatmap that correlates each output word with every input word, which allowed they discover that several heads have learned similar dependency mappings. By using this map as a qualitative index of the feature importance of each head, they pruned the heads with similar dependency maps, concluding that there is no noticeable loss in the translation quality in this case.

This paper also raised a relevant question to the community: are deep learning models usually biased towards increasing complexity and model parameters unnecessarily? Less complex models have a smaller carbon footprint and energy waste, which can lower their impact on the nature, bringing the NMT community closer to the status of Green AI.

## 3.5 Attention variants for Sequence to Sequence learning

Not only architectural improvements can lead the Transformers to translations of higher quality, the attention mechanism has also received some relevant contributions. In the approach followed by FAN *et al.* [24], the multi-head attention is modified to a new mechanism called multi-branch attention, where each branch is an independent multi-head attention mechanism.

Despite the evidence that increasing the number of attention heads may lead to redundancy, this work outperformed the standard Transformer baseline, which goes against the analysis previously stated. In this paper, other techniques such as a drop branch mechanism (similar to Dropout [25], but applied to the Transformer's branches) and a specific initialization recipe for each branch may have impacted the final outcome. The composition of such techniques makes unclear whether the benefits come from scaling the attention complexity or from those architectural alternatives.

Going back to the literature, variants applied over the set of words that the attention blocks sees as well as over the weight distribution functions are among the most common variants explored. In the paper published by LUONG *et al.* [17], the attention mechanism is split into 2 classes: global and local attention. The former always attends to all source words of a given sentence, whilst the latter only looks at a subset of words at a given time.

Local attention is explored with different alignments: monotonic and predictive. In the first case, they just assume that the source and the target sequences are roughly monotonically aligned, and concatenate the target hidden state $\mathbf{h}_t$ with the equivalent source version $\mathbf{h}_s$. In the predictive variant, they use a Gaussian distribution centered around the word to be predicted and define alignment weights based on the values of the Gaussian, so nearby words are more important than distant words. Following this approach, the attention weights are biased to follow a normal distribution.

The main findings of this paper are: (1) attention-based models usually outperform non-attentional ones and (2) properly tuning the alignment model can yield to better results. A small increase in the performance was observed by adding a custom weight function (Gaussian-based in this case). They also concluded that using an ensemble involving some of the proposed attention architectures may be effective. However, further details of how this ensemble was implemented are not provided.

Incorporating different attention perspectives to the attention mechanism is another common exploited approach, which has been followed by CALIXTO *et al.* [26]

and CUI *et al.* [27]. In [26], a single layer feedforward network is used to compute the expected alignment between each annotation vector in both mechanisms, but this increases the number of parameters to be trained on the model. In [27], the concept of forward and backward attention is introduced, where specialized masks help the Transformer to model word order information, a slightly different schema than the positional encoder used in the original paper. Both mechanisms are concatenated with the standard global and local attention. Even though this approach leads to a better BLEU score, it also increases the number of model parameters.

There is no strong evidence that tuning the attention mechanism would provide relevant benefits in low-resource settings. Furthermore, all variations usually increase the quantity of model parameters, raising its complexity and amount of data required for learning, along with the amount of resources required to be trained (GPU memory). In a single GPU scenario, a larger and more robust GPU would be necessary in these cases. Increasing the number of GPUs unfortunately does not fit in the scope and budget of this work. However, data augmentation and embedding techniques have a higher potential to succeed in our case, hence the focus of this work was shifted away from a possible tuning of attention mechanisms.

### 3.5.1 Positional encoding

To insert information relative to the order of the words in a sequence without recurrence or convolutions, the encoder may augment the base layer of the network with positional encodings correspondent to the order of occurrence of each word in the source sentence. Considering for instance a sequence length of $n = 5$ and the use of 3-dimensional bit vectors to represent each word ($d = 3$), we could generate vectors to represent the location of the words like $\mathbf{l} \in 000, 001, ...110, 111$. In this case, a position matrix can be represented by $\mathbf{P} \in \mathbb{R}^{n \times d}$. However, a more compact representation of the word position can be obtained by using basis functions and real-valued weights. VASWANI *et al.* [3] proposes a sinusoidal basis to encode this position:

$$p_{(i,2d)} = \sin\left(\frac{i}{C^{\frac{2d}{d_{model}}}}\right), \tag{3.14}$$

and

$$p_{(i,2d+1)} = \cos\left(\frac{i}{C^{\frac{2d}{d_{model}}}}\right), \tag{3.15}$$

where $d$ is the dimension or level within the representation (the quantity of real numbers used to represent a word) and $C$ corresponds to a maximum sequence length, set to 10000 in the original paper. If a matrix $\mathbf{P}$ has dimensions $d = 4$, then it's $i$th row is given by

$$p_i = \left[\sin \frac{i}{C^{\frac{0}{4}}}, \cos \frac{i}{C^{\frac{0}{4}}}, \sin \frac{i}{C^{\frac{2}{4}}}, \cos \frac{i}{C^{\frac{2}{4}}}\right]. \tag{3.16}$$

Figure 3.8 provides an intuitive view of how the entries of the matrix $\mathbf{P} \in \mathbb{R}^{n \times d}$ for a sequence of length $n = 60$ and an embedding dimension of size $d = 32$ contributes in encoding the word position.

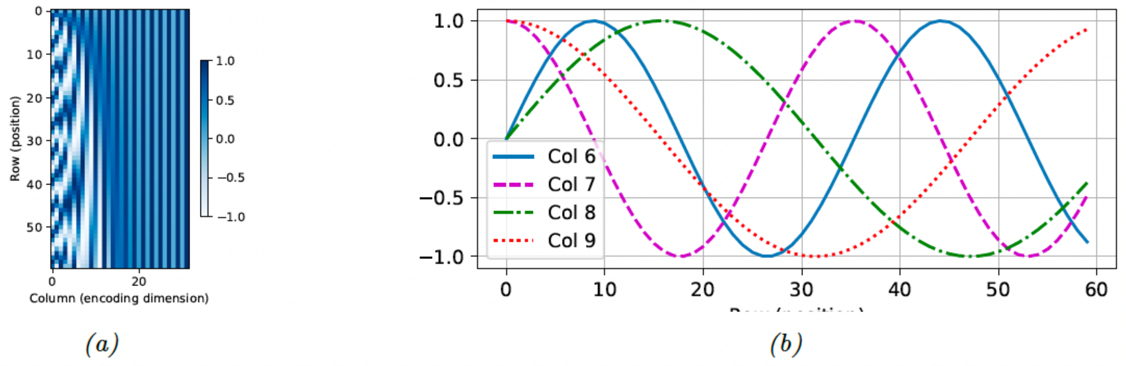


Figure 3.8: (a) Positional encoding matrix with dimensions $n = 60$ and $d = 32$. (b) Basis functions for columns 6 to 9. Generated by `http://code.probml.ai/book1/15.25` and adapted from MURPHY [21].

This representation brings a two-fold advantage: first, it can be computed for arbitrary length inputs (up to $L \leq C$), unlike mapping alternatives such as one-hot encoding [21]. Second, the representation of one location is linearly predictable from any other, given the knowledge of their relative distance [21]. Any word position can be modelled by $p_{t+\phi} = f(p_t)$, where $f$ is a linear transformation, and this property is inherited due to the use of trigonometric functions. Note this by the following decomposition

$$\begin{bmatrix} \sin(\omega_k(t+\phi)) \\ \cos(\omega_k(t+\phi)) \end{bmatrix} = \begin{bmatrix} \cos(\omega_k \phi) & \sin(\omega_k \phi) \\ -\sin(\omega_k \phi) & \cos(\omega_k \phi) \end{bmatrix} \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix}. \tag{3.17}$$

If $\phi$ is small, then $p_{t+\phi} \approx p_t$, correctly inducing the desired relative position correlation. Once the positional encodings $\mathbf{P}$ are computed, they are combined with the original Word Embeddings $\mathbf{X}$ through the following:

$$POS(\mathbf{X}) = \mathbf{X} + \mathbf{P}. \tag{3.18}$$

## 3.6 Relevant Transformer variations

Since their publication, Transformers were customized in many ways by researchers aiming to define new state of the art. Not all the variations are worth a story to tell, but some of them have exceeded expectations and deserve some space in this discussion. Beginning with BERT (Bidirectional Encoder Representations from Transformers), the Transformer idealized by DEVLIN *et al.* [28], designed to extract deep bidirectional representations from unlabeled text.

Roughly, two interesting tasks can be addressed with BERT: one is predicting masked tokens in a sentence, i.e., the BERT performs like a masked language model. In this case, during training some percentage of the input tokens are hidden at random, and the tokens masked are predicted by the model. The second aims to extend the model's functionalities to other NLP tasks, such as question answering and natural language inference, and this is accomplished via next sentence prediction. The intriguing aspect of BERT is that it can be easily fine-tuned to perform a wide range of tasks, whilst still keeping an architecture that isn't far from the original Transformer. In these cases, this model still beats the best results available for a range of tasks, such as natural language understanding, inference and question answering.

Some other variants of the traditional Transformer architecture were heavily focused on increasing complexity aiming to surpass the state of the art, and solving a wider number of tasks. The variant presented by BROWN *et al.* [29] have shown that it is possible to improve the performance of the model by simply increasing the model size, and, as a consequence, the dataset size and the computational power used as well.

A typical example is the GPT-3. This model comes in various sizes, with the smallest model having 125M parameters, 12 attention layers, each one with 12 heads of dimension 64. The largest version has 175B parameters, along with 96 attention layers, each one containing 96 heads with dimension 128. This model has raised an environmental concern, as it spent about 190,000 kWh to be trained [30]. Some GPT-3 outcomes are interesting to be highlighted: (1) in text generation, humans have failed in discriminating artificially generated summaries from genuinely human-made texts for 24% of the cases; (2) Machine Translation, surpassing the best available BLEU scores for many language pairs wherein the English idiom figures as the target, and (3) other less traditional tasks, performing surprisingly well in arithmetic and word scrambling.

Not all the Transformer variants focused on firepower, though. There is some research aiming to come up with lightweight alternatives to the Transformer. In this case, the main goal is increasing the model performance but still keeping the computational burden under satisfactory standards. Both models proposed on MEHTA *et al.* [31] and a year later on MEHTA *et al.* [32] are able to improve the efficiency of the original Transformer as a language model on a range of tasks. The definition of efficiency here is achieving a high BLEU score (for NMT) or another score (for other tasks), but considering the number of model parameters integrating the cost function.

The Delight model, one of the most recent works in this area, have changed the inner mechanism of the Transformer layer by using the delight transformation, which is composed by a group of linear transformation layers that are shallower and narrower near the input whilst deeper and wider near the output. The authors state that this transformation decouples the attention dimensions from the depth and width, allowing representations to be learned efficiently using block-wise scaling. Delight also replaces the multi-head attention mechanism by a single-head attention, reducing the number of parameters needed in training. Over the WikiText-103 dataset, the authors obtained an increase on BLEU score of 3.3%, whilst using only 80% of the standard Transformer parameters and requiring just 23 hours to train, as opposed to the 37 hours used by the original model.

## 3.7 Alternative training objectives

Many of the state-of-the-art models have reached their related scores using the traditional cross-entropy loss. Most language models have also focused on the maximum likelihood learning objective, as stated in TAN *et al.* [33].

Let $\mathbf{x}$ to be the real-valued representation of a token in the source language and, analogously, $\mathbf{y}$ to be the same for a token in the target language. The equation to obtain log-likelihood function in this case is given by:

$$L(\theta) = \sum_{s=1}^{S} \log P(\mathbf{y}^{(s)}|\mathbf{x}^{(s)};\theta), \tag{3.19}$$

where $\mathbf{x}$ and $\mathbf{y}$ are data and target vectors from the training set $D = \{(\mathbf{x}^{(s)}, \mathbf{y}^{(s)})\}_{s=1}^{S}$ whose size is $S$, and $\theta$ represent the model parameters. The standard goal of most NMT algorithms is maximizing the value of this function on the training set data, as follows

$$\hat{\theta}_{MLE} = argmax(L(\theta)). \tag{3.20}$$

Nevertheless, the work of RANZATO *et al.* [34] indicate two drawbacks of this approach. First, NMT models are usually trained on a specific dataset, but are not exposed to their own translations and consequently their own errors: this phenomenon is referred to as exposure bias. Finally, the MLE estimation is defined at the token-level rather than sentence-level. So, theoretically, the objetive settled by the optimization is not aligned with the final objective, which is generating a sentence that matches a correct human translation.

With this problem in mind, these authors introduced the Mixed Incremental

Cross Entropy Reinforce (MIXER), which switches the learning objective towards a sentence-level training. The proposed algorithm tries to handle the problem of backpropagating gradients from non-differentiable metrics like BLEU, exploiting some ideas borrowed from the Reinforcement Learning (RL) area.

Other works have also put efforts in this learning objective issue. For instance, the technique proposed by SHEN *et al.* [35], called Minimum Risk Training (MRT). In this learning objective, the model considers the discrepancy between its prediction and the equivalent gold standard translation. The model generates multiple samples from the search space provided by beam search to calculate the expected risk, and this risk is defined as the expected loss with respect to the posterior distribution. This loss function is not parameterized, thus, not differentiable. Moreover, it can also be negative when considering sentence-level evaluation metrics such as BLEU.

The work of SHEN *et al.* [35] reported better results than RANZATO *et al.* [34]. The authors' allegation for this is because the MIXER approach samples only one candidate for calculating the reward, whilst their approach, Minimum Risk Training (MRT), generates multiple samples. This potentially increases the MRT's discrimination capabilities, a pretty close effect than to increase the beam size in beam search in the standard NMT setup.

Apart from the efforts put into iterating on the learning objective and the benefits alleged by these papers, some side effects of switching this cost function are also reported in CHOSHEN *et al.* [36]. Some major pointed issues are the weakness of RL-based approaches for optimization, the fact that some gains can not be fully attributed to the techniques, and some convergence issues related to the training objectives proposed by SHEN *et al.* [35] and RANZATO *et al.* [34]. Therefore, based on this literature review, the potential of changing the learning objective to obtain better translations under low-resource is uncertain. As a result, this work opted for not considering any change in the learning objective.

## 3.8 Alternative positional encoding

The positional encoding performed by the traditional Transformer is called absolute positional encoding. As it simply adds the positional embeddings to the original Word Embeddings, it comes with the drawback that word position and word meaning are entangled, despite representing unrelated information. The work of KE *et al.* [37] warns for the risks brought by possible mixed correlations between the word semantics and the position information itself, which may add unwanted noise to the learning process and limit the expressiveness of the model.

These authors propose the use of the Transformer with Untied Positional Encoding (TUPE), which processes the word contextual and positional information separately, following different parametrizations, and adds them later. By following their approach, the resulting attention weights can be decomposed into terms that represent interactions between word-to-word, word-to-position, position-to-word and position-to-position, a factor that they claim to enrich the word representation [37]. Their model is compared with other literature proposals for positional encoding, like mixing relative positional encoding with absolute positional encoding, obtaining a superior performance over them with the GLUE dataset benchmark.

# Chapter 4

# The low-resource domain context

The term "low-resource" recalls situations where any of the resources needed to perform a task are scarse. In NMT, this phenomenon can manifest itself in a number of ways. Complex models that contain a high number of parameters to be learned also demand an abundant quantity of data, a phenomenon that is explained by the VC (Vapnik-Chervonenkis) dimension theory. If a supervised complex model has scarce data for training, its potential in achieving relevant generalization scores can be significantly hindered. This can even lead to simpler algorithms outperforming more complex models in a given task. There are some situations to which phrase-based Machine Translation models can surpass neural-based models, especially when the dataset is small, such as the case study reported in SENNRICH and ZHANG [38].

Scarce data also often strongly contributes to increasing the open vocabulary challenge, mentioned in Section 2.3.7. Solving the translation task with reduced data also means solving the open vocabulary challenge under adverse conditions. Increasing the complexity of the corpus content somewhat turns this challenge more difficult, so the technique adopted must be resource efficient to excel under such conditions.

The second most important issue that is fundamental for the success of a translation model is the infrastructure used for training. NLP models usually need one or more GPUs or TPUs (Tensor Processing Units). Besides, models that require a high batch size such as Transformers cannot be trained in practical terms with a CPU. Switching from CPU to GPU may accelerate the training process hundreds of times, reducing to hours what would require days to be completed.

The memory footprint demanded to tackle with a dataset in some infrastructure is directly related to its size. Two dataset dimensions that directly influence on the memory footprint are the number of sentences and the number of words per sentence. Such numbers are relevant to determine the ideal batch size.

Another factor that contribute on the choice of the batch size is the number of trainable parameters of the model, which linearly increase with the depth of a neural network. A common practice is to increasing the batch size proportionally to the network depth. This approach may be observed for the GPT-3 variants described in the work of BROWN *et al.* [29]. There is no theory establishing a deterministic rule relative to how the batch size must be increased as a function of the number of parameters. However, it seems to be a good practice to consider them directly proportional, to achieve a better generalization during training.

Another factors that must be also considered when one addresses an NMT task, but rarely represent a bottleneck: disk storage, internet connection, and a correctly configured Python environment (or another language of choice). These factors do not usually represent the biggest challenge because such resources are more accessible than cutting edge GPUs. Therefore, when referring to low-resource domains in the following sections, only the dataset size and the GPU power constraints will be considered.

Fortunately, techniques that turn the NMT models more resilient to open vocabulary as well as some of the previously mentioned issues are available in the literature. Some of these strategies will be addressed in the following sections, right after a brief overview of some challenges faced whilst tackling Portuguese NMT under low-resource settings.

## 4.1 Intrinsic challenges of this work

Before focusing our attention to low-resource approaches, our initial objective was creating a model that would surpass the state of the art. However, as soon as the computational infrastructure at our reach has shown its intrinsic limitations (*e.g.* the GPU out of memory error), we realized that we could not compete with the literature at such a level. Experiments involving datasets of different lengths and the process of tuning hyperparameters for many Transformer variants can be quite challenging when one uses a 12GB GPU, which is the standard low-cost or free cloud hardware solution available in 2021.

Transformer variants have been showing an astonishing performance in a variety of practical applications, attracting multiple companies and dedicated research groups that seek to reach the state-of-the-art status. Researchers from more resourceful countries or universities usually have multiple GPUs in the same cluster, enabling them to evaluate solutions with a higher GPU memory usage. However, this can be quite expensive without sponsorship to get on your own. Given such limitations, our decision was to contribute to the low-resource branch of NMT given the overall practical limitations.

The process of understanding the constraints of our environment (specially GPU memory related) also involved a non-friendly trial-and-error approach. Even when setting our batches to begin higher than usual and dwindle in size, the out of memory error does not usually appear in the beginning of the first epoch or even in the first few epochs. Frequently, we had to conduct a number of experiments that took some time until fail aiming to understand what infrastructural factors are constraining the experiment to behave as desired. This trial-and-error process was repeated whenever the dataset used or the number of parameters (complexity) of the model under training has changed, significantly increasing the time spent when performing the experiments with different datasets and techniques.

The following sections provide a brief coverage over the techniques considered in this work to deal with low-resource issues and some technical trade-offs necessary to be settled when using them in the NMT context.

## 4.2 Subword Embeddings

In Chapter 2, we only mentioned some interesting aspects of switching to the subword level. In this section, we provide a more detailed explanation of how the subword models operate. The process is quite similar to producing vector embeddings for the words in a sentence (word embedding), differing in the sense that each word is broken into several tokens, and to each token is produced an embedding vector. The methodology that dictates how words are split into subparts has an important influence in the explainability (quality) of the final subword representations.

This type of embedding is more robust relatively to the out of vocabulary (OOV) occurrence as compared to the traditional word-level approach [22]. When one breaks a word into smaller tokens, the smaller resulting tokens are often more likely to appear in both training and test sets. Considering the translation of rare tokens, the use of Subword Embeddings relieves part of the burden of the model having to be trained in a dataset with a statistically relevant occurrence of that token, enabling it to reach an adequate performance even with small datasets. Subword models can be designed to act in a character level or in an intermediate level between word and character (like phonemes, syllables and others).

There is a number of Subword Embeddings available in the literature that could be exploited in low-resource settings. This work concentrates on evaluating a token embedding approach known as BPE, due to its success in a wide number of applications.

### 4.2.1 BPE (Byte Pair Encoding)

The pioneer work introducing Subword Embeddings is due to SENNRICH *et al.* [39]. According to it, the segmentation is performed based on BPE, an algorithm introduced to break the raw text into sub-parts. Roughly, BPE breaks the words from a corpus into smaller parts (the smallest BPE unit is a character). Some of these parts are subsequently merged, and the number of merge operations is a hyperparameter to be tuned. All the steps involved in BPE can be found at Algorithm 1.

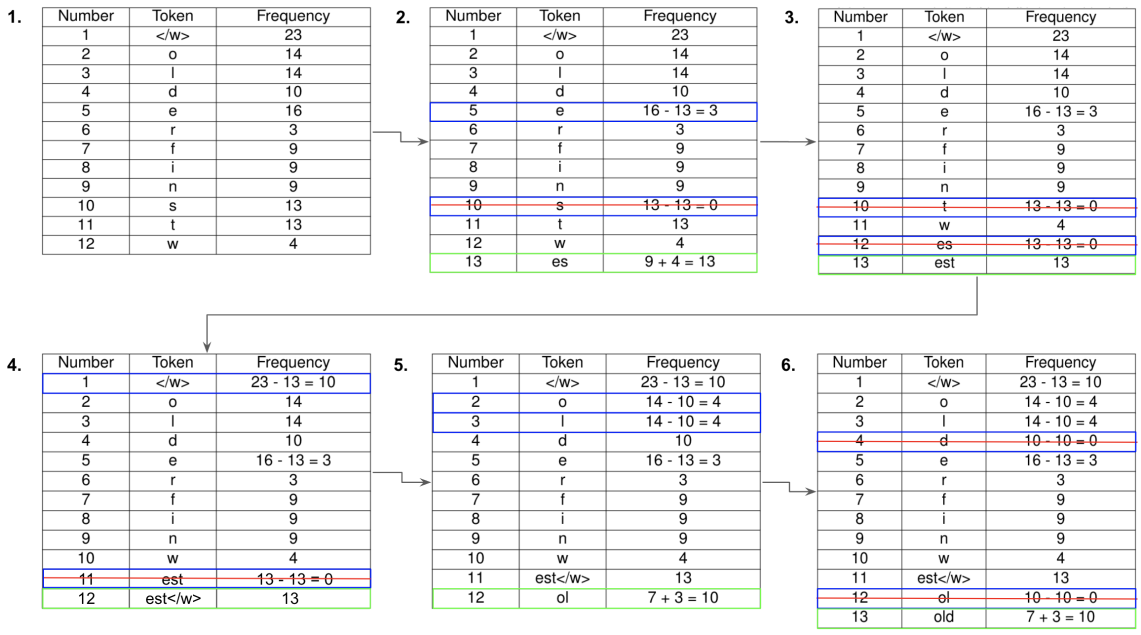
The original BPE implementation has only one mandatory hyperparameter: the number of merges, which is not much intuitive since the vocabulary size is what most concerns an NLP researcher. Thus, one of the original implementation drawbacks is that the practioner cannot clearly define the maximum vocabulary size a priori, relying solely on the number of merge operations for that. This opens a margin for creating a huge vocabulary with a low frequency for some Subwords, possibly hindering model performance. Despite the previously mentioned drawbacks and clear room for improvement, this method became widespreadly used in many applications because it is simple, cheap to run, easy to understand, and effective.

**Algorithm 1** Pseudocode for BPE, taken from BOSTROM and DURRETT [40]

```
1: Input: set of $D$ tokens, target vocabulary size $k$
2: procedure BPE($D, k$)
3:     $V \leftarrow$ all unique characters in $D$ (about 4,000 in English Wikipedia) |V| < k
   ▷ Process of merging tokens
4:     $t_L, t_R \leftarrow$ The most frequent bigram in $D$
5:     $t_{NEW} \leftarrow t_L + t_R$                                   ▷ Make a new token
6:     $V \leftarrow V + [t_{NEW}]$
7:     Replace each occurrence of $t_L, t_R$ in $D$ with $t_{NEW}$
8:     return $V$
9:     end procedure
```

The process of breaking words into smaller units inherently increases the number of samples per token. This grants to the algorithm the ability to cope better with the OOV issue. The original paper [39] compares BPE against segmentation techniques commonly used in Statistical Machine Translation (SMT), for instance the bi-gram segmentation. Bi-gram stands for the combination of 2 tokens into a single word. For more details, please check the original paper. The tests performed against the other segmentation techniques focused on evaluating translation scores and OOV coverage, showing that BPE outperforms most of them.

To illustrate better how BPE works, consider a hypothetical corpus with 4 distinct words, each one having a particular number of occurrences: $\{"old </w> " : 7, "older </w> " : 3, "finest </w> " : 9, "lowest </w> " : 4\}$, where



1.

| Number | Token | Frequency |
|---|---|---|
| 1 | </w> | 23 |
| 2 | o | 14 |
| 3 | l | 14 |
| 4 | d | 10 |
| 5 | e | 16 |
| 6 | r | 3 |
| 7 | f | 9 |
| 8 | i | 9 |
| 9 | n | 9 |
| 10 | s | 13 |
| 11 | t | 13 |
| 12 | w | 4 |

2.

| Number | Token | Frequency |
|---|---|---|
| 1 | </w> | 23 |
| 2 | o | 14 |
| 3 | l | 14 |
| 4 | d | 10 |
| 5 | e | 16 - 13 = 3 |
| 6 | r | 3 |
| 7 | f | 9 |
| 8 | i | 9 |
| 9 | n | 9 |
| ~~10~~ | ~~s~~ | ~~13 - 13 = 0~~ |
| 11 | t | 13 |
| 12 | w | 4 |
| 13 | es | 9 + 4 = 13 |

3.

| Number | Token | Frequency |
|---|---|---|
| 1 | </w> | 23 |
| 2 | o | 14 |
| 3 | l | 14 |
| 4 | d | 10 |
| 5 | e | 16 - 13 = 3 |
| 6 | r | 3 |
| 7 | f | 9 |
| 8 | i | 9 |
| 9 | n | 9 |
| ~~10~~ | ~~t~~ | ~~13 - 13 = 0~~ |
| 11 | w | 4 |
| ~~12~~ | ~~es~~ | ~~13 - 13 = 0~~ |
| 13 | est | 13 |

4.

| Number | Token | Frequency |
|---|---|---|
| 1 | </w> | 23 - 13 = 10 |
| 2 | o | 14 |
| 3 | l | 14 |
| 4 | d | 10 |
| 5 | e | 16 - 13 = 3 |
| 6 | r | 3 |
| 7 | f | 9 |
| 8 | i | 9 |
| 9 | n | 9 |
| 10 | w | 4 |
| ~~11~~ | ~~est~~ | ~~13 - 13 = 0~~ |
| 12 | est</w> | 13 |

5.

| Number | Token | Frequency |
|---|---|---|
| 1 | </w> | 23 - 13 = 10 |
| 2 | o | 14 - 10 = 4 |
| 3 | l | 14 - 10 = 4 |
| 4 | d | 10 |
| 5 | e | 16 - 13 = 3 |
| 6 | r | 3 |
| 7 | f | 9 |
| 8 | i | 9 |
| 9 | n | 9 |
| 10 | w | 4 |
| 11 | est</w> | 13 |
| 12 | ol | 7 + 3 = 10 |

6.

| Number | Token | Frequency |
|---|---|---|
| 1 | </w> | 23 - 13 = 10 |
| 2 | o | 14 - 10 = 4 |
| 3 | l | 14 - 10 = 4 |
| ~~4~~ | ~~d~~ | ~~10 - 10 = 0~~ |
| 5 | e | 16 - 13 = 3 |
| 6 | r | 3 |
| 7 | f | 9 |
| 8 | i | 9 |
| 9 | n | 9 |
| 10 | w | 4 |
| 11 | est</w> | 13 |
| ~~12~~ | ~~ol~~ | ~~10 - 10 = 0~~ |
| 13 | old | 7 + 3 = 10 |

Figure 4.1: Steps of the BPE algorithm applied to an example corpus.

the token $< /w >$ marks the final word boundary. The whole process of breaking words into sub-parts is presented in Figure 4.1 and explained below, where each enumerated bullet represents an equivalent step in the figure:

1. The first step of the algorithm simply counts the number of characters in the corpus, assuming that each character defines a candidate token. After that, the frequencies of occurrence of token pairs are inferred, and the most frequent token pair is identified and added to the list of token candidates, as indicated in green at the bottom of table in the algorithm step 2.

2. The most common token pair identified in the previous stage is "$es$", occurring $9 + 4 = 13$ times. Thus, the tables entries related to the occurrence of these characters in a stand-alone fashion must be updated, simply by subtracting the number of common occurrences. Naturally, this happens to distinguish the occurrences of the single character tokens $e$ and $s$ from those to which the token $es$ happens, which are now separated. In this case, the row associated with $s$ is eliminated (represented by the red line), as it only spawns within the $es$ token.

3. In the step 3, a new token pair that shows the same frequency than $es$ is $est$, which by coincidence also contemplates all the occurrences of $es$ and $t$, leading to their elimination at this step.

4. In the next step, the merge of $est$ with $< /w >$ results in the most frequent new

candidate token having (13) occurrences. This merge results in a reduction of the frequency associated with $< /w >$ to 10.

5. Another common token pair is *ol*, which has a frequency of 10 in the given vocabulary. To represent this new token, the single characters *o* and *l* are subtracted by this number of occurrences (10), reducing their frequency to 4.

6. In this step the token *old* is formed, incorporating all the occurrences of *ol* and removing the token *d*. Then, the process is interrupted due to the impossibility of fusing any of the candidate tokens.

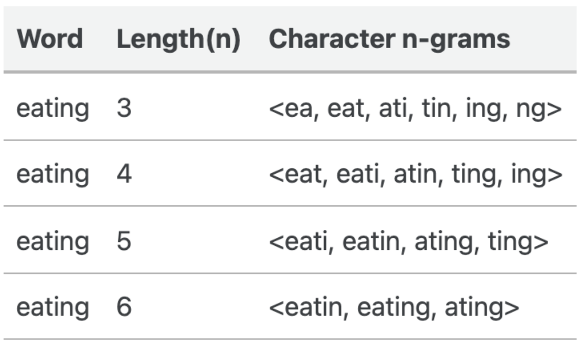
7. Table 4.1 summarizes the final tokens integrating the vocabulary. Note that the final vocabulary size (11) is inferior to the one considered at the start of this process (12).

The main difference between this method and the one derived from bi-gram segmentation is that the latter is capable of breaking words into interpretable units. Finding a clever way to split the words was one of the main challenges at that time. On the other hand, BPE is a greedy algorithm inspired in a compression algorithm with the same name, leading to some occurrences to which the subword tokens may not represent a morpheme with an enclosed meaning. However, despite having some meaningless tokens, the percentage of the ones that are meaningful are enough for it to stand out among other techniques.

Table 4.1: Final distribution of tokens (step 7).

| Number | Token | Frequency |
|---|---|---|
| 1 | </w> | 10 |
| 2 | o | 4 |
| 3 | l | 4 |
| 4 | e | 3 |
| 5 | r | 3 |
| 6 | f | 9 |
| 7 | i | 9 |
| 8 | n | 9 |
| 9 | w | 4 |
| 10 | est</w> | 13 |
| 11 | old | 10 |

The effectiveness of this approach is evaluated in SENNRICH and ZHANG [38], which concentrates on evaluating the practical effects of changing word-level NMT to subword NMT using BPE. An ultra-low resource setting experiment was the one that benefited the most, raising a BLEU of 7.2 to 16.6 using the model of BAHDANAU *et al.* [15].



| Word | Length(n) | Character n-grams |
|---|---|---|
| eating | 3 | <ea, eat, ati, tin, ing, ng> |
| eating | 4 | <eat, eati, atin, ting, ing> |
| eating | 5 | <eati, eatin, ating, ting> |
| eating | 6 | <eatin, eating, ating> |

Figure 4.2: $N$-grams generated by Fast Text for the word "eating".

### 4.2.2 Subword variations

Some research has also dedicated attention to using character-level RNNs only when the process of word-level embeddings generate an OOV word, as in LUONG and MANNING [41]. Switching to character-level has also its drawbacks since character level spliting result in longer sequences of tokens wherein each one contains less information, yielding to computational and modelling challenges. The most basic one relates to the total of memory required to represent a set of words in a character-level word representation, as it is substantially higher than subword or word-level solutions.

Sticking to an unique level of embedding is a paradigm broken in the work of CHEN *et al.* [42], wherein different levels of granularity (word, subword and character) are combined to augment the token representations. These authors extended the encoder with a character-attention mechanism, aiming to better learn source-side representations and incorporate information into the decoder with a multi-scale attention, allowing the character-level conjugated with word-level information to improve the translation. Results show that such models when compared to the single granularity alternatives achieved higher scores, and the use of multiple levels also benefited the model to address the OOV issue.

Pre-trained Word Embeddings were also explored in the work published by BOJANOWSKI *et al.* [43], which became popular under the name of Fast Text. This algorithm was inspired on the Skipgram Word2Vec variant. It treats words as a bag of character $n$-grams and adds the special symbols "$>$" and "$<$" at the beginning and end of each word, respectively, to allow the algorithm to distinguish prefixes and suffixes from other character sequences. A visual distribution of the $n$-grams derived from the word "eating" is available at Table 4.2, where $n = 6$.

Basically, to each word is assigned a number of $n$-grams, and in practice only $n$-grams for $3 \leq n \leq 6$ are produced to each word. When the word representation is generated, it is represented by an index and a respective set of $n$-grams, which are called subword tokens. Finally, the word is represented by the sum of the vector representations correspondent to this set of $n$-grams.

## 4.3 Transfer Learning

Transfer Learning is a common practice in the Machine Learning area, roughly consisting of reusing a model developed for a specific task as a starting point of a model dedicated to another task, typically reducing the complexity and time of the training process. This technique was exploited in ZOPH *et al.* [44], where weights associated to NMT models developed with a high-resource language pair are transferred to models targeting a low-resource language pair, both having English as the target language. By following this approach, the NMT model turns able to outperform the previous state-of-the-art Syntax-Based Machine Translation (SBMT) model. The metric used for evaluation was BLEU, and this happened with one of the four low-resource language tasks: the Hausa language.

Another transfer learning approach that may optimize the process of learning vector word representations is the use of Pre-trained Word Embeddings. This may be achieved by adopting unsupervised language models to adjust their previously learned vectors and improve them within a $N$-dimensional space, where $N$ denotes the word-embedding size. The learned representations (weights) are then reused by the NMT model as starting values of its embedding layer, instead of proceeding with the typical random initialization adopted in models trained from scratch.

The Fast Text algorithm presented in Section 4.2.1 was the transfer learning strategy chosen for evaluation in this work, due to disposing of the benefits from the Subword Embeddings, analogous to the BPE. The representations loaded in this case have the advantage of carrying semantical and syntactical relations previously stored in the Pre-trained language model.

An interesting work that experimentally evaluates the effect of Pre-trained Word Embeddings in low-resource scenarios is QI *et al.* [45]. This analysis makes use of the same dataset considered in this work, TED Talks, which will be further described later. However, the model explored is based on BAHDANAU *et al.* [15], trained in a different framework, and initialized differently than in our work.

One of the experiments described in this paper analysed the effect of loading Pre-trained Word Embeddings on reduced datasets, considering three distinct languages. The authors report that there is a "sweet spot" relative to the quantity of data available used for training that results in more efficient models. When loaded to a model trained with only 10% of the TED Talks dataset, the Pre-trained Word Embeddings have yielded small gains as compared to a random initialization. However, when trained over 30% of the dataset, the BLEU gains suffer a peak, which monotonically decreases with the increase on this proportion of dataset instances used for training.

Another experiment reported aims to enforce consistent embedding spaces across both languages, a technique called word embedding alignment. The authors exploit an approach that learns orthogonal transformations that convert the Word Embeddings of multiple languages to a common space, and use these transformed embeddings instead of the traditional learning process. These generated embeddings will be used as a starting point for the NMT model as usual. They achieve this by using linear geometrical transformations that normalize the vectors to the same range. Unfortunately, mixed results are reported, turning it hard to derive conclusions regarding its effectiveness. Some experiments report that Pre-trained Word Embeddings are more effective when applied to more related language pairs (e.g. Portuguese-Spanish).

Aiming to allow transfering the knowledge from language models trained on monolingual data to NMT, GULCEHRE *et al.* [46] proposes 2 solutions to fulfill this task: shallow fusion and deep fusion. The first uses a language model during decoding to rescore the candidate words that the translation model is considering as the next word to be predicted. The other one combines the decoder and the language model, which are coordinated with a controller mechanism. Unfortunately, the improvements reported are limited to the range of 0.5 up to 2 BLEU points in all experiments.

The results reported in the work [45] for several language pairs in a range of scenarios seem to be more consistent, whilst the technique presented at [44] is highly language and dataset dependent, as well as subjected to a high deviation. Therefore, there is some evidence that employing Pre-trained Word Embeddings tend to be more effective than using transfer learning with a model trained on a different dataset.

## 4.4 Data Augmentation

The quantity of parallel sentences is a key factor to the success of NMT, and data-hungry complex models may impose difficulties to the researchers aiming to contribute to this field. Unfortunately, large-scale parallel corpora is not available for the majority of the existing language pairs. On the other hand, it is much easier to obtain monolingual corpora. The Internet is a good example as disposing of a range of websites in multiple languages. This scenario motivated us to study data augmentation strategies to support the NMT task.

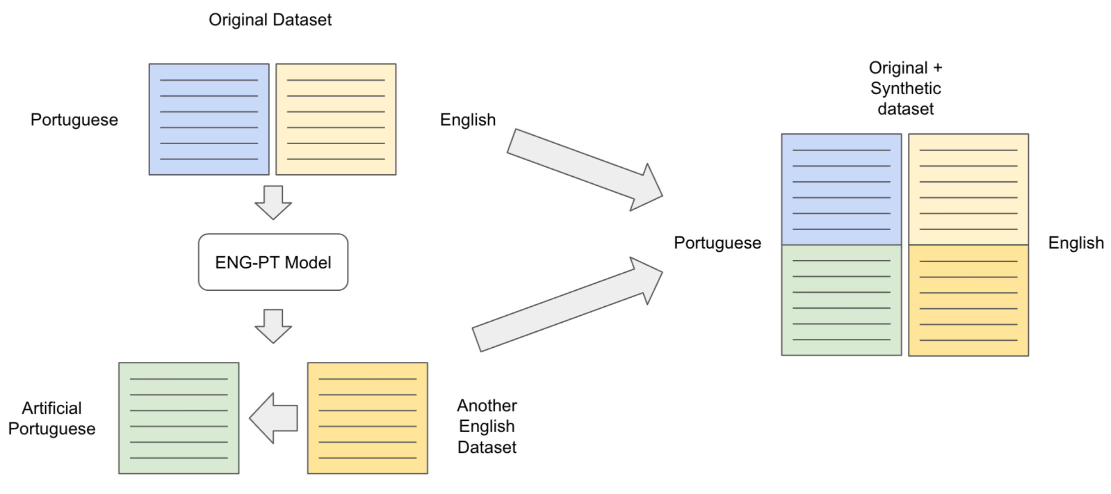


Figure 4.3: The Augmentation of a dataset with Back Translation

### 4.4.1 Back Translation

A branch of research in Data Augmentation is dedicated to the study of variations of the technique referred to as Back Translation (BT). In SENNRICH *et al.* [47], the authors extract monolingual data corresponding to the target-side language, train a model to translate back to the source language and use it to build synthetical translations. Figure 4.3 represents the augmentation process with a simplified representation. A NMT model is used to translate in the reverse direction (target to source language), generating artificial sentence pairs that are appended to the original dataset when training the main NMT model.

BT has shown to be a simple, yet effective method to address low-data availability in many domains, as shown in PONCELAS *et al.* [48]. This study concludes that the translation performance (measured with BLEU) is increased when a certain quantity of synthetic translations are added, but this gain seems to tail off when the dataset balance is tipped too far in favour of the synthetic data.

In theory, the reason why the performance increases with the addition of artificial sentence pairs generated by another model is that when generating new synthetic pairs, "noise features" are added, which may be beneficial to learning. This sort of feature engineering technique in NMT is only beneficial in some cases. This strategy is only effective when the share of synthetic sentences is limited to a small portion of the whole dataset, thus keeping the "natural" sentences as the majority of instances. It is hard to define a rule to determine this ideal share, since it varies accordingly to the languages and text characteristics. Adding too much noise may compromise the models' performance, so this technique must be used with caution.

Different strategies for generating synthetic data are presented and compared in XIA *et al.* [49]. This study evaluates several augmentation methods in scenarios wherein English (ENG) is paired with high resource languages (HRL) and low resource languages (LRL). The strategies include techniques such as pivoting, word substitution (with the use of a dictionary), and an unsupervised NMT approach. A two-step pivoting method is also introduced: it uses a ENG-HRL model to create artificial HRL examples, and then use a HRL-LRL to create artificial sentences to the LRL. The authors propose methods to perform HRL-LRL translations which benefit from language characteristics such as word order. They claim that this ends up approximating better the true LRL sentences than using a ENG-LRL model. This technique has shown to be particularly useful for languages with even more restricted resources available than Portuguese.

In some scenarios, this work reported that methods such as two-step pivoting may outperform Back Translation. However, this study is limited to a few datasets and the gains reported in BLEU seem to not be expressive. Considering that consistent results are reported when other works apply the traditional Back Translation in a wider range of datasets, we decided to use this technique rather than two-step pivoting for addressing low-resource issues in this work.

# Chapter 5

# Experiments on Neural Machine Translation

The experimental investigation conducted in this work aim not only to report a possible score increase when applying a technique $X$ or $Y$, but also to address qualitative issues about the translations generated. The correlation between errors patterns and a specific technique is a process often overlooked in the NLP field, typically focused in optimizing one or more performance scores. Questions such as the following are not commonly addressed:

1. How switching to a Subword Embedding may influence on the patterns of errors that a model makes?
2. Is the sentence complexity or the incorporation of external knowledge a determining factor for model's performance? Do they induce some qualitative bias?
3. Is it beneficial to augment the dataset with synthetic data (Back Translation)? How may the noise introduced by this procedure influence on the patterns of the errors produced by a model?

These questions require a deeper analysis that goes over just matching words between the reference and the translations generated by the model, raising a demand for the definition of a categorization scale of the model's error with a linguistic focus. A contribution of this work is proposing a bidimensional criterion to assess the translation quality. The hypothesis raised here is that by analysing such patterns, we may enrich the insights that could be extracted from the experiments.

Another goal of this analysis is trying to understand the effects on the translation quality of the level of complexity of the sentence, as from the dataset nature. In this direction, in our experiments, datasets from different domains were exploited to validate the hypothesis regarding the effectiveness of using datasets from distinct domains on BT. Another relevant factor that influenced on the choice of the datasets considered by this work are the limited GPU memory and data availability. In the following sections these experiments and their corresponding outcomes are described in greater details.

## 5.1 Methodology

Datasets with low to medium complexity levels were carefully selected for the proposed analysis: Tatoeba [50] and TED Talks [51]. Both represent a low-resource scenario due to the scarce number of sentences, in alignment with some references [38] [44]. Tatoeba is mainly constituted of basic to intermediate level English sentences, which can be interpreted as a corpus from a "school domain", ranging from elementary to middle school education constructions. The Tatoeba dataset contains 143.8k small sentences with basic to intermediate English level, posing a low complexity challenge for the NMT task. It includes 26.3k unique words in PT and 15.3k in EN.

TED Talks is a medium-size dataset covering a range of subjects, including from low to highly complex sentences. It represents a mixed domain dataset, since it contains talks from experts on distinct areas. News Commentary v16 [52] is a news focused domain monolingual dataset, used for BT in this work, and includes a rich range of sentences in terms of content and complexity.

In the experiments, 10% of Tatoeba was held out for testing, whilst the remaining data was split into 10% for validation and 90% for training, using a seed equal 0. Despite TED disposing predefined training, test, and validation sets, the original validation set is too small (906 sentences), leading us to move the last 20 talks (2081 sentences) from the training set to this set. As a result, the training set contains 236.1k (1918 talks) sentences, randomly sampled to define training batches using a seed equal to 157, and the test set includes 11.4k sentences. Additionally, all text was pre-processed to eliminate all XML enclosed sentences and tags, except for the ones related to title and description.

The experiments were conducted using Python. Gensim was the library used to deal with Word Embeddings, and Spacy's Portuguese and English tokenizers were applied to the previously mentioned datasets. All experiments were performed on a single GPU, using the Google Colaboratory and Kaggle infrastructure. Typically such environments dispose of NVIDIA GPUs like Tesla P100, Tesla K80 or Tesla T4, with a GPU memory size ranging from 12GB to 16GB.

Transformers is the architecture adopted in all experiments, and the parameters tuned for them are $d_{model} = 256$, $d_{ff} = 256$, 8 attention heads, and the $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$

are matrices with dimension $64 \times 64$. The Pre-trained Fast Text-based models, which employed the embeddings described in [53] for Portuguese and [54] for English, are the only exceptions. In this case, both embeddings had $d_{model} = 300$ and 6 attention heads. The weights downloaded from the site published in the work of FARES *et al.* [54] were Pre-trained on the Gigaword [55] corpus, considering the ID 16 option on the download list available at their website, after clicking on the repository tab. We chose this option as it was produced without lemmatization, since this technique is not applied when processing our datasets. All variants adopted the Adam optimizer with $\beta \in (0.9, 0.98)$ and $\epsilon = 10^{-8}$, a learning rate of $10^{-4}$ and the beam search considered a beam with size equal to 3. The early stopping criterion was based on the validation perplexity behaviour for ten epochs, halting the training in case of performance stagnation. During training, a new model was saved each time the best validation perplexity was achieved.

Regarding the BPE implementation, a toolkit developed by Google was preferred due to an interesting feature added over the original algorithm: it allows a priori vocabulary size definition instead of using the traditional number of merges. Sentencepiece [56] is a language agnostic fast and lightweight implementation made in C++, widely used for subword segmentation. It was settled considering a maximum vocabulary size of 32k tokens.

To assess quantitative performance, Sacrebleu [57] and NLTK [58] were the BLEU variants exploited in our experiments. The major difference between them resides in a stronger Sacrebleu's penalization over cases to which the translated and reference sentences differ in length.

## 5.2 Quantitative study

To understand and evaluate the translation outcomes derived from an NMT model, both quantitative and qualitative aspects must be taken into account. To evaluate the former, scores such as Sacrebleu and NLTK BLEU provide distinct optics that allow to isolate the sentence length penalization variable. For the latter, the knowledge of an expert may help to break the problem into smaller parts, categorizing quality issues with the goal of capturing error patterns.

The main goal of the experiments described in this section is to evaluate how the candidate techniques react to low-resource conditions quantitatively. The central idea resides in inferring which of them may help in circumventing issues that might arise in such cases and how to tune the auxiliary techniques based on a set of hyperparameters and setup conditions. To aid the reader on the task of following the rationale, it follows a brief summary describing all the experiments performed when conducting this evaluation process:

1. The impact of restricted dataset content
    - The experiment described in Section 5.2.1 evaluates quantitatively the effect of several adverse limited data conditions on the NMT task performance.
2. Effects of switching to subword level and ways of incorporating external knowledge
    - In Section 5.2.2, a discussion regarding ways of loading Pre-trained Word Embeddings and the effectiveness of switching to subword level (BPE) standalone is made. The experiments consider Fast Text and BPE strategies added to the Transformer.
3. Effectiveness of Back Translation when addressing different low-resource settings
    - The experiment shown in Section 5.2.3 evaluates the performance impact of using BT to extend a dataset, considering different synthetic shares for the dataset and exploiting data from the same as from a different domain.
4. Comparison of Transformer candidates against the Google Translate benchmark
    - The last experiment from Section 5.2.4 compares the BLEU score of the aforementioned low-resource Transformer candidates with the output of the Google Translate service taken at the date of 11/01/2022.

### 5.2.1 The impact of restricted dataset content

The process of training a model with constrained data may affect its performance in different ways. Therefore, different low-resource scenarios were created to try to unveal such effects. To shed light on such limitations, we considered a hypothetical experimental scenario, where only a fraction of TED and Tatoeba training sets were used in training. The following percentages were considered: 33.3%, 50%, 66.6%, 83.3%. According to these percentages, in Tatoeba the quantity of sentences used for training ranges from 47.9k to over 143k and in TED from 78k to the totality of 236k.

This experiment had the purpose of inferring if there is a minimum number of sentences that a parallel dataset must have for the model to reach a good performance. It should become clear that after a certain size (or fraction, in this case) the additional performance obtained with the increase on the number of training instances is minimal, so alternatives must be explored to maximize model's performance. Table 5.1 summarizes the outcomes of this experiment.

Table 5.1: Performance obtained with models developed under restricted data (see text)

| Fraction of the Dataset | Tatoeba | | | | TED | | | |
|---|---|---|---|---|---|---|---|---|
| | Sacre-bleu | NLTK BLEU | Batch Size | Epochs | Sacre-bleu | NLTK BLEU | Batch Size | Epochs |
| 33.3% | 48.64 | 67.09 | 512 | 76 | 24.7 | 57.65 | 30 | 40 |
| 50% | 52.53 | 70.12 | 512 | 65 | 25.18 | 56.46 | 30 | 40 |
| 66.6% | 55.3 | 72.12 | 512 | 58 | 26.22 | **56.81** | 29 | 36 |
| 83.3% | 56.24 | 73.18 | 512 | 58 | **26.74** | 56.57 | 28 | 30 |
| 100% | **57.99** | **74.07** | 512 | 58 | 25.24 | 55.36 | 28 | 30 |

The results show that the Sacrebleu scores for the Tatoeba dataset were about twice the same as those achieved with TED, corroborating with the much higher complexity of the latter. The BLEU metrics for both datasets have shown a monotonic behaviour, with exceptions to TED in two cases: Sacrebleu (100% $\times$ 83.3%) and NLTK (100% $\times$ 83.3 and 83.3% $\times$ 66.6 %). The reasons for such findings may include: (1) the possible use of synonyms in the translations, an aspect ignored by any BLEU metric; (2) a higher incidence of *Repetition* errors due to data quality issues (to be discussed further in Section 5.3); (3) the more complex and richer TED content, which might have led to a wider subject coverage in the training set, reducing model accuracy, a hypothesis requiring a future investigation.

Models developed with a fraction of the original training datasets (66.6%) performed surprisingly well, with the highest performance occurring at 100% and 83.3% for Tatoeba and TED, respectively. Besides, the competitive results in these cases as compared to 66.6% indicate that the latter percentual may be sufficient for the model not suffering from low-resource data availability issues, but a more accurate percentage in each case requires a further investigation.

### 5.2.2 Effects of switching to subword level and ways of incorporating external knowledge

Aiming to evaluate the leveraging effects of Pre-trained Fast Text and BPE [39] strategies in low-resource NMT tasks, BPE models were implemented exploiting the Texar framework [59] (PyTorch version). In contrast, the alternative models considered customized PyTorch [60] solutions.

In practical terms, there are a range of alternatives when applying Pre-trained Word Embeddings for NMT, thus identifying the most cost-effective approach can only be achieved experimentally. The main processing mechanism of the Transform model includes a embedding layer at the encoder and decoder blocks. The translation task is typically applied to a pair of distinct languages, hence one embedding will be used as the source language weights associated to this layer in the encoder, whilst the embedding from another language will perform similarly in case of the decoder. Another option is to replace the randomly initialized weights in either the encoder or the decoder, inducing the model to be trained from a "partial warm-start".

Apart from the embedding strategy adopted, when loading the weights using the Gensim library, some optional parameters need to be specified in the *load_word2vec_format* function: *binary* and *unicode_errors*. These parameters are set to *False* and "*strict*" by default, but the reference [54] recommends to load the embeddings using *False* and "*replace*". Both combinations were experimented to evaluate a possible impact over the performance scores, restricting only to the TED Talks dataset. Table 5.2 exhibits the corresponding results, including in its first line the last one from Table 5.1 to ease the comparison. All the Fast Text experiments used the default parameters, except for the 4th row of the table that used Fares' recommendations [54].

Table 5.2: Transfer learning and Subword Embeddings translation results

| Technique applied | Tatoeba | | | | TED | | | |
|---|---|---|---|---|---|---|---|---|
| | Sacre-bleu | NLTK BLEU | Batch Size | Epochs | Sacre-bleu | NLTK BLEU | Batch Size | Epochs |
| None | 57.99 | 74.07 | 512 | 58 | 25.24 | 55.36 | 28 | 30 |
| Fast Text (encoder of [53], randomly initialized decoder) with default package parameters | 56.96 | 69.91 | 512 | 50 | 24.07 | 61.69 | 30 | 45 |
| Fast Text (encoder of [53] + Gigaword decoder) with default package parameters | N/A | N/A | N/A | N/A | 24.54 | 61.76 | 32 | 89 |
| Fast Text (encoder of [53] + Gigaword decoder) with Fares' parameters [54] | N/A | N/A | N/A | N/A | 21.26 | 52.31 | 32 | 80 |
| Subword BPE | **66.63** | **83.02** | 512 | 40 | **40.26** | **72.20** | 32 | 40 |

Curiously, the adoption of Fast Text embeddings is related with an unexpected performance drop for both datasets. Conversely, the gains observed with BPE, which also exploits Subword Embeddings, were impressive. The variants following the recommendations [54] achieved worse performance than those related to the default parameters.

One possible hypothesis for Fast Text's bad performance is a possible overspecialization to other corpora domains, since it was produced with Internet data corpus mined by a crawler [53]. The higher relative BPE gain with TED (15.02) as compared to Tatoeba (8.64) signalizes the effectiveness of BPE in dealing with more complex NMT scenarios, especially those involving a more diverse vocabulary, avoiding the OOV occurrences.

### 5.2.3 Effectiveness of Back Translation when addressing different low-resource settings

One hypothesis still unanswered is whether the noise inserted by a model when generating synthetic pairs may be beneficial to the performance of another model or not, and in what extent. Addressing this problem requires an experimental approach, since different domains, synthetic text ratios, natural languages and model characteristics constitute unique scenarios. The experiments covered in this section aim to answer questions like: What may be considered as an ideal synthetic ratio when applying BT? How the domain of the augmentation dataset impacts on the performance scores?

The experiments reported here are restricted to the TED dataset, since the low complexity of Tatoeba poses an easy challenge which could obfuscate possible higher gains of model performance. Data augmentation was performed with synthetic sentences produced with the own TED (using its left out sentences) and with the News dataset. These experiments aim to verify if data augmentation can help to reach higher BLEU scores under different low-data availability conditions.

Some scenarios of comparable models were created to evaluate the use of different domains versus the original (non-synthetic) dataset in the BT process. The motivation lies in understanding if the addition of extra data brings a boost in models' performance or even if it could compromise BLEU score. Datasets with different ratios of synthetic sentences were chosen, considering samples that contain 50%, 33.3%, 16.6% and 8.33% in terms of representivity. The artificial samples were generated by a single Transformer operating with the reverse language pair, i.e., EN-PT, reaching 27.73 and 63.8 points for the Sacrebleu and NLTK scores, respectively.

In addition, one extra BT variant was produced assuming a higher number of sentences than the original dataset size (120%). This variant included 20% of synthetic TED sentences, which somewhat makes it comparable to the 83.3% + 16.6% of TED dataset version, since both have the same synthetic ratio (20%). The idea here was checking if adding more synthetic data on the top of the original dataset might enhance the previously reported scores. One last detail of this experiment is that the subset of back-translated sequences appended to the training sets was randomly sampled using the following seeds: 157 (TED) and 0 (News). Table 5.3 exhibits the results.

Table 5.3: TED Talks Back Translation results

| Technique applied | Batch size | Epochs Trained | Sacrebleu | NLTK BLEU | Synthetic Ratio |
|---|---|---|---|---|---|
| None (Original TED) | 30 | 27 | 25.24 | 53.26 | N/A |
| Reduction of TED to 50% | 40 | 30 | 25.18 | **56.46** | N/A |
| BT (50% of News synthetic examples) | 34 | 33 | 21.80 | 51.34 | 50% |
| BT (50% of TED synthetic examples) | 34 | 28 | **25.95** | 56.42 | 50% |
| Reduction of TED to 66.6% | 36 | 29 | 26.22 | 56.81 | N/A |
| BT (33.3% of News synthetic examples) | 34 | 27 | 24.12 | 53.77 | 33.3% |
| BT (33.3% of TED synthetic examples) | 34 | 27 | **27.54** | **58.95** | 33.3% |
| Reduction of TED to 83.3% | 28 | 30 | 26.74 | 56.57 | N/A |
| BT (16.6% of News synthetic examples) | 34 | 29 | 31.28 | 63.30 | 16.6% |
| BT (16.6% of TED synthetic examples) | 34 | 27 | **34.62** | **64.61** | 16.6% |
| Reduction of TED to 91.67% | 34 | 24 | 26.74 | 56.57 | N/A |
| BT (8.33% of News synthetic examples) | 36 | 23 | 29.66 | 60.4 | 8.33% |
| BT (8.33% of TED synthetic examples) | 34 | 24 | **32.27** | **65.13** | 8.33% |
| Augmented TED (100% of TED + 20% of News) | 28 | 24 | 29.31 | 60.80 | 16.6% |

One may readily infer that the increasing score trend switches its behaviour, depending on the size of the synthetic portion with respect to the whole dataset. For a more severe restriction on the dataset size (50%), using other-domain synthesized sentences is harmful to model performance, whilst own-domain synthesis resulted in a marginally better BLEU score. However, for a lower percentage of synthetic data, positive effects started to appear. Considering a less significant restriction ($\approx$ 33%), using the same domain sentences in Back Translation led to a mild increase in both BLEU values when as to the Original TED, signalizing that such "noisy" sentences may contribute to increasing the translation quality. When applying an intermediate restriction ($\approx$ 16.6%), the approaches from both domains are quite effective, resulting in models that largely surpass the model developed over the original data. Finally, when considering a small restriction ($\approx$ 8.3%), the monotonic behaviour of scores related to same data domain augmentation switches its trend and starts decreasing, but still representing an improvement over the benchmark. Regarding augmentation with synthesized data from other domain, the score also decreases similarly. It is important to emphasize that the scores of all variants were generated using the same test data, otherwise the comparison would not be fair.

One intriguing outcome of this experiment was that the Transformer variant trained on a larger dataset behaved unexpectedly worse than variants considering the same synthetic ratio on smaller datasets. The decrease in BLEU was significant: more than 5 points (34.62 vs 29.31) when compared to the 83.3% TED version. On the other hand, when considering the Transformer trained on the original dataset, the addition of 20% of News data enhanced the original models' performance. One possible hypothesis to explain this, but that deserves further investigation, is that since the TED dataset includes mixed domain content, the last piece of the dataset (about 16%) may contemplate texts with a vocabulary pretty decorrelated to the remaining dataset parts. The addition of more Back Translation data may have also led to an increase on the model's performance, but exploring this hypothesis further will remain as a proposal for future work.

### 5.2.4 Comparison of Transformer candidates against the Google Translate benchmark

To evaluate how efficient were the previous stategies in tackling low-resource issues, we compare these models with a benchmark that isolates this limitation. The comparison is crucial to understand how well the model has mitigated issues related to such restricted domains. Therefore, Google Translate was used as a reference benchmark. Table 5.4 shows the results for BLEU considering the best performing model candidates among the previous experiments, and for this benchmark.

Table 5.4: Performance comparison between our best performing models and Google Translate

| Model | Batch size | Epochs Trained | Sacre-bleu | NLTK BLEU |
|---|---|---|---|---|
| Transformer (Original TED) | 30 | 27 | 25.24 | 55.36 |
| Subword BPE | 32 | 40 | **40.26** | **72.20** |
| Reduction of TED to 83.3% | 28 | 30 | 26.74 | 56.57 |
| BT (16.6% of News synthetic examples) | 34 | 29 | 31.28 | 63.30 |
| BT (16.6% of TED synthetic examples) | 34 | 27 | **34.62** | **64.61** |
| Google Translate (used on 11/01/2022) | Unknown | Unknown | **52.19** | **78.74** |

Considering the Sacrebleu BLEU metric that includes sentence length penalization, our best model reached 77.1% of Google's performance, making use of a model and computational infrastructure many orders of magnitude inferior in terms of power consumption and usage cost, than the one associated with Google's model. When switching to the NLTK metric, which ignores the sentence length, the gap between them significantly reduces, as this ratio goes up to 91.69%. We must stress that the current Google model is able to identify even disease acronyms, such as ACS (Acute Coronary Syndrome), one achievement almost impossible to be reached for models developed over only one low-resource dataset. This model is even able to predict translations having just part of the word as input.

Unfortunately, there is not much to be said regarding the underlying techniques that empower Google's model. The last papers over this topic that were made available suggest the use of a RNN-based encoder and a decoder with attention, but the type of attention mechanism is not mentioned. When looking at more recent information, based on a blog post of CASWELL and LIANG [61], they grant a relevant increase in BLEU to the use of Back Translation techniques, specially for low-resource languages. The latest papers available are heavily outdated [62] [63] and the techniques have probably been deprecated or updated somehow. Recent updates are mostly reported in blogs, with the latest one summarizing their Artificial Intelligence achievements in 2021 [64], with some mentions to the new Google Translate features, such as the efforts towards reducing the gender bias.

The Google Translate model is probably built upon a carefully curated set of techniques and parameters. One possibility to bridge the gap between it and other models is to try a combination of the discussed low-resource strategies, although there are no guarantees that their positive benefits would necessarily be constructive when combined. As the main goal from this work is more focused on qualitative outcomes, and not in boosting some performance score as far as possible, we focused our energy in quantifying and interpreting patterns of translation errors. The next section provides more detail of such attempts.

## 5.3 Qualitative study

Understanding a problem through the lens of a quantitative methodology can already yield a range of findings, but engineers often underestimate the potential of a qualitative approach to complement such analysis. In modern tech companies, both methodologies are commonly used together to provide a richer view of the problem, which would be too restricted by solely considering a qualitative focus. When speaking of product management, for instance, designers and analysts are the left and right arms of the manager in the task of evaluating a scenario of interest in a qualitative and quantitative way, respectively. The same analogy can be applied here, where computational researchers working on the Natural Language domain often refer to linguistics experts to better address and solve a language related problem.

### 5.3.1 Challenges of a qualitative analysis

When performing Machine Translation, a number of language, model specific and data availability nuances might pose a range of challenges with a high potential of hindering qualitative analysis results. Most of them are fairly widespread and well-known by most researchers in this area, but depending on whether the dataset has been curated, how much of its biases have been identified and addresses, and also how well dimensioned it is; the struggle to create high quality translations can increase significantly.

The first challenge are data quality issues, such as references that do not match the exact expected translations, i.e., contain some imprecisions or errors. Data quality issues may have a significant impact on the task of evaluating correctly the translations from a model, and of course, they are also scattered throughout the TED Talks dataset [51]. Such data quality issues can also intensify some known flaws of using BLEU for evaluation, especially considering its lack of ability to catch synonyms, resulting in an overestimated loss score due to human translation biases in the references. Table 5.5 shows some sentences arbitrary selected from the TED Talks database that illustrates these issues.

Table 5.5: Selected references with errors

| Original Sentence (PT) | Reference | Suggested Correction |
|---|---|---|
| A medida que fazíamos nossas futuras expedições Eu estava vendo criaturas em fontes hidrotermais e às vezes coisas que eu nunca tinha visto antes, às vezes coisas que ninguém tinha visto antes, que realmente não haviam sido descritas pela ciência no momento em as vimos e as **imaginamos**. | As we did some of our subsequent expeditions, I was seeing creatures at hydrothermal vents and sometimes things that I had never seen before, sometimes things that no one had seen before, that actually were not described by science at the time that we saw them and **imaged** them. | Switch **imaged** to **imagined**. |
| Fico feliz em dizer que não temos este tipo de situação – um cirurgião com quem conversei alguns anos atrás que tinha trazido para aqui gêmeos xipófagos a fim de separá-los, em parte para ficar famoso. | I'm glad to say we don't have the kind of situation **with** - a surgeon I talked to a few years ago who had brought over a set of conjoined twins in order to separate them, partly to make a name for himself. | Remove **with** from the reference. |
| Fomos para **São Paulo** onde os outdoors foram banidos. | We went to **San Paulo** where they have banned outdoor advertising. | The city must be correctly represented without the tilde: Sao Paulo. |

All models show some sort of error patterns after being evaluated with a number of curated sentences, which are not usually trivial to explain. Besides, splitting a qualitative analysis into smaller and focused aspects that can be addressed and assessed by a bidimensional criterion is not also straightforward. However, the better the answer found to this problem, the better the proposed analysis can provide insights about models' performance, especially in a practical sense. This scenario raises a second challenge, regarding how to set the breadth and depth of the qualitative criterion used. To address the complexity of defining such a criteria, the help of an expert on the field is mandatory. These issues motivated us to hire someone with 9 years of teaching and translating experience in English, that describes himself as a C1 level english speaker, according to the CEFR scale [4].

The participation of a translator in this work was a rewarded interaction. In other words, he was called on demand to help, and the whole process was constrained by my personal budget available for it. Since all the interactions were conducted autonomously, without any type of sponsoring, the number of evaluated sentences and thus the generalization power of the analysis is somewhat restricted despite highly valuable in terms of innovation in this field.

Part of the qualitative work conducted here was dedicated to understanding whether text complexity does influence the error patterns when generating translations or not. This led the translator to categorize Ted Talks database instances in different levels according to the CEFR scale [4], which will be better detailed in a specific section ahead.

The process of creating a qualitative bidimensional criterion turned out to be the third challenge, especially regarding how to avoid a stronger inductive bias, i.e., biases not inherently related with the reference, but with the qualitative criterion itself. The consulted translator defined some categorization ideas, but understanding his rationale and giving feedbacks whilst avoiding to not bias the final decision has demonstrated to be a very subtle process.

The bias question is also strongly linked to the process of sampling database instances: samples had to be curated to match the expected complexities and reach a certain number of excerpts per complexity. Random choices of excerpts were picked when performing this task to avoid bias, demanding a lot of manual work. However, when getting near the pre-established goal of 15 excerpts, the approach demanded a more flexible conduction. Whenever a pair of excerpts was near the threshold between two adjacent complexity levels, we had to discuss whether it could be switched to the level requiring more samples or if it should remain in the original level.

It is worth to emphasize that 100 distinct pairs of sentences were considered in this analysis, but the total number of evaluations is quite higher, as the candidates have changed in the first steps taken. Also, a higher number of models than what is actually reported was selected for the qualitative experiments, because we had to evaluate their pattern of errors and decide upon keeping them or keep exploring. Increasing the number of evaluated sentences was also necessary to cover all the complexity levels. Ensuring that all the main types of errors are covered in our sample of the dataset was the ultimate goal. The clusters defined by each qualitative criterion should be well-scoped and avoid intersections to bypass a potential interpretability loss. The discussion for both complexities and qualitative criteria was fruitful as it helped to better grasp the consistency of the criteria chosen as the samples were evaluated.

### 5.3.2 The qualitative criteria

As previously mentioned, two qualitative dimensions were created to validate the hypothesis stated in the beginning of this chapter: sentence complexity and error patterns. One relevant hypothesis raised before that is still to be addressed is that the patterns of errors related to the same model but considering different text domains (or subjects) and text complexities may very. We have chosen to complement the observed error patterns with sentence complexity level labels aiming to isolate the effects of both and to be better prepared to validate our initial hypothesis. Another intent behind this segmentation in complexity levels is that it also should help to better understand the associations between text complexity and the error patterns of the techniques exploited.

#### Setting the classes of errors

To analyse the patterns of errors generated by the model, a multicategorical criterion was created along with a human translator to cluster the types of errors. Eight categories were considered: *Reference Matching*, *Omission*, *Out of Context*, *Verb Tense*, *Meaning Deviation*, *Insertion*, *Repetition* and <*unk*> errors. A detailed error description of each category can be found at Table 5.6.

Table 5.6: Description of the error classes defined in the process of error analysis

| Error class | Description |
|---|---|
| *Reference Matching* | A similar, but different word from the reference is used in a context to which the sentence can still be understood, or the order of words are switched. |
| *Omission* | Corresponds to ocassions wherein a part of the source sentence is ignored by the model. Consequently, the translated sentence does not integrate the corresponding content. |
| *Out of Context* | The model uses a word that doesn't fit in the surrounding context, and the translation loses cohesion. This error also contemplates nonexistent words created by BPE. |
| *Verb Tense* | When conjugating a verb, the model misses to capture the right tense and changes it on its guess, generating an innaccurate translation. |
| *Meaning Deviation* | The model produces a sequence of words that is way different from the expected translation. This usually happens when several words are changed and the sentence is rephrased to preserve cohesion between these new words, resulting in a loss of meaning when compared to the source sentence. |
| *Insertion* | This category indicates that the algorithm has inserted words that are unnecessary given the context. |
| *Repetition* | Happens when the likelihood estimate of the next translation becomes biased, inducing the model to keep translating the same set of words more than once. |
| <*unk*> | The traditional token that a model outputs whenever none of the words known by it would fit in a given context. |

We must recall that the <*unk*> error class is inside the list but it naturally happens when the model sees a word during the inference process that was not explored in training. This category is directly related to the cohesion and coherence of the model when processing a sentence. All the sentences evaluated in the qualitative analysis were classified under these 8 classes of errors. For the sake of illustration, Table 5.7 shows some cherry-picked sentences and their correspondent evaluation regarding error classes.

Table 5.7: Examples of wrong translations classified according to the proposed error criterion

| Original sentence | Reference | Model outcome | English error diagnostic |
|---|---|---|---|
| E porque isso é verdade? | And why is it true? | And why is this true? | ***Reference Matching* error**: switched "it" for "this", without losing meaning. |
| Michael Specter: O perigo da negação da ciência. | Michael Specter: The danger of science denial. | Michael <unk>: the danger of science. | 1) *Omission* error: denial; 2) <*unk*> error: Specter was interpreted as an unknown token. |
| Mas, na verdade, nós somos mudados. Somos marcados, sem dúvida, por um desafio, quer físicamente, emocionalmente, ou ambos. | But, in fact, we are changed. We are marked, of course, by a challenge, whether physically, emotionally or both. | But in fact, we're changed. We're <unk>, of course, by a challenge, whether we can physically, emotionally, or both. | 1) ***Omission* error**: the model omitted "marked"; 2) ***Out of Context* error**: the model added "can", completely changing the original meaning of the sentence; 3) **<*unk*> error**: "marked" wasn't successfully translated. |
| Ganhamos um monte de prêmios. | We've won a bunch of awards. | We won a bunch of awards. We won a lot of awards. | 1) ***Verb Tense* error**: switched the Present Perfect form in "we've won" for Simple Past in "we won"; 2) ***Repetition* error**: "We won a lot of awards" was inserted, repeating the original reference's meaning. |
| E exigiu muita preparação, tivemos que construir câmeras e luzes e todo tipo de coisas. | And it took a lot of preparation, we had to build cameras and lights and all kinds of things. | And it was a lot of please, we had to build cameras and lights and all kinds of things. | ***Meaning Deviation* error**: the model switched "it took a lot of preparation" for "it was a lot of please". |
| A mesma coisa, só que maior. | Same thing, only bigger. | The same thing, only bigger than that. | ***Insertion* error**: the model inserted "the" and "than that" in the translation. |

The selected examples in Table 5.7 low in complexity and focused on being didactic. They contain simpler mistakes than the average, as the goal is just to illustrate the descriptions of error categories. A small piece of text containing many errors makes the process of distinguishing between these error categories even harder, which would compromise the didactic purpose of Table 5.7 and might increase the chance of mislabelling. Another factor that contributes to the same issue is the sentence complexity, which is covered as one of the qualitative pillars of the analysis.

### Setting the complexity categories

To define the dataset instances included in this qualitative analysis, random samples were selected from TED, analysed by the human translator, and stratified according to the CEFR scale [4]. Due to dataset characteristics, this study restricted its coverage to sentences classified as A1, A2, B1 and B2. A very few sentences were classified as from the C1 and C2 levels in this set: less than 5 samples from a total of 100 analysed sentences. As time and budget were natural limitations, we decided to remove them from the experiments. Thus, from the 100 distinct sentences evaluated, 60 were used for the experiments performed in this work, with 15 belonging to each category of complexity.

Categorizing sentences solely based on the general descriptions available for the CEFR levels is a hard task. Table 5.8 illustrates cases found in this analysis, with the corresponding complexity assigned by the translator and the rationale behind this classification according to him. It helps in the task of absorbing the scope of the qualitative criteria, since it has one sample per complexity level and most of the errors are classified and the reasons for that are exposed.

We must stress that there are no clear hard rules to describe the thought process behind choosing a given complexity category or error class. The existence of some error patterns tends to favour the evaluation towards a label, but a broader evaluation and richer exploratory analysis would be necessary to define more embracing and generic rules. Some other sentences are available in a separate repository for a more interested reader[1].

### Design of experiments

The classification in error classes and complexity levels for all the 60 sentences culminated in a dataset with a lot of binary columns labelling each sentence regarding the occurrence or not of each one of the qualitative error categories. It isn't trivial to extract insights from these outcomes by simply inspecting the ordinary frequencies observed for this dataset in its raw form, as there are bidimensional associations that couldn't be easily identified or described by this way. This issue motivated us to use dimensionality reduction and clustering techniques to unveil bidimensional patterns.

The set of experiments dedicated to clarify the translation error patterns was carefully thought to evidence possible associations between the model candidate, sentence complexity and their respective class of error patterns. To guide the reader over the qualitative analysis, below there is a summary of the experiments performed:

[1]Detailed error descriptions and some evaluation samples can be found at `https://github.com/Art31/pt-nmt-low-resource.git`.

Table 5.8: Evaluation of some translated sentences

| Original sentence | Reference | Model guess | Model Type | Complexity and reason | English error classification |
|---|---|---|---|---|---|
| Você não muda nunca, né? | You never change, do you? | You don't change it, do you? | Transformer with BPE | A1 - Simple present | 1) *Insertion*: it was added. 2) *Reference Matching*: Never/Don't are close in meaning |
| Tente perguntar coisas assim: "Como é que foi aquilo" ? | Try asking them things like, "what was that like?" | Try to ask things like this: "how is it that?" | Transformer with BPE | A2 - ING used as infinitive and intermediate level question in Simple Past | 1) *Verb Tense*: switched "try asking" (continuous) for "try to ask" (infinitive). 2) *Omission*: "them". 2) *Insertion*: "this". 3) *Meaning Deviation*: "what was that like?" was changed to "how is it that?" |
| E vocês sabem, uma epifania é normalmente algo que encontramos que tínhamos deixado cair em algum lugar. | And you know, an epiphany is usually something you find that you dropped someplace. | And you know, an epiphany is normally something that we've found that we'd left somewhere. | Transformer 66% of TED | B1 - Simple present and simple past, with uncommon vocabulary | 1) *Reference Matching*: usually/normally. 2) *Meaning Deviation* and *Verb Tense*: "we've found that we'd left somewhere" lost both its original meaning and tense |
| Ele tinha acabado de ouvir a primeira e a quarta sinfonia de Beethoven, e veio até o backstage se apresentar. | He had just heard a performance of Beethoven's first and fourth symphonies, and came backstage and introduced himself. | He'd just heard the first one and the fourth symphony of Beethoven, and came to the <*unk*> to introduce him. | Transformer 33% of TED | B2 - Past perfect and past simple | 1) *Omission*: The model omitted "a performance of" that appears in the original sentence. 2) Unk error. 3) *Insertion*: "first **one**". |

1. Statistical hypothesis test over the proportion of errors committed by each model

   - Multiple Fisher exact tests were used to infer whether a particular model and technique combination indeed contributes to mitigate some modalities of errors, or maybe if a given model is more prone to producing an error than others or not.

2. Hierarchical clustering analysis

   - Bidimensional associations between the errors, models and sentence complexity were evaluated in this experiment aiming to shed light on possible patterns of errors associated to each technique.

3. Correspondence analysis

    - This algorithm also allows the analysis of bidimensional associations, exploring a visual 2-D representation that enables interesting visual interpretations, complementing the previous analysis

In the following experiments, only one of the Back Translation Transformers generated in this work is considered. The model there referred to as BT is the Transformer trained on a dataset composed 83.3% of the original TED and 16.6% of the other part of the same dataset subjected to BT.

### 5.3.3 Error proportion analysis with the Fisher Exact test

The purpose behind this experiment is to understand if solely based on the error prevalence, we can assume that one variant (model submitted to sentences of some complexity) is biased to a specific error class, with statistical significance. An appropriate hypothesis test was chosen for this goal, given the limited sample size: the Fisher Exact Test. Fifteen sentences from each complexity level were presented to all the four candidate models: (1) the best Back Translation variant, trained on 83.3% of TED augmented with 16.6% of TED back-translated data, (2) the Transformer trained over a fraction of 33% of TED, (3) the Transformer trained on 66.6% of TED, and (4) the BPE variant, trained over the entire dataset.

The rationale for using such candidates is based on their performance on the previous experiments and the structural differences between them. We must recall that one of the hypothesis is whether switching to the subword level and using data augmentation leads to some impact in our qualitative indexes. This analysis included the reduced TED Transformers to evaluate possible effects of restricting the dataset size over the error patterns.

Table 5.9 depicts the number of errors committed by each model, stratified by sentence complexity and error category. The cells inside the complexity category rows indicate in bold which is the highest prevalence. The cells belonging to the rows with an average label show numbers in bold if the hypothesis test involving that error and that model had a statistically significant result, which will be further detailed in Section 5.3.3. Considering the limitations of this analysis, such as the reduced sample size and the availability of only one translator, all models performed quite similarly regarding the *Omission* and *Verb Tense* prevalence. Nonetheless, BPE stands out by having the lowest average prevalence of *Insertion*, *Repetition* and *Omission* errors. Curiously, the Back Translation variant has the highest prevalence of *Out of Context* and *Reference Matching* errors, which may be an influence of using of artificial data. Both Transformers trained on reduced TED have a very similar distribution of errors considering the error types as their average prevalences are usually very close, indicating that this modality of low-resource constraint may have a significant impact on their errors patterns.

Table 5.9: Ratios of class errors per dataset and sentence complexity (see text)

| Model Name | Complexity | *Reference Matching* | *Omission* | *Out of Context* | *Verb Tense* | *Meaning Deviation* | *Insertion* | *Repetition* | <*unk*> error |
|---|---|---|---|---|---|---|---|---|---|
| BT | A1 | 6/15 | 3/15 | 1/15 | 1/15 | 0/15 | 1/15 | 0/15 | 1/15 |
| | A2 | 13/15 | 7/15 | 2/15 | 4/15 | 3/15 | 6/15 | **1/15** | 3/15 |
| | B1 | 13/15 | 5/15 | **3/15** | 5/15 | 5/15 | 5/15 | 0/15 | 3/15 |
| | B2 | **15/15** | **11/15** | 0/15 | **11/15** | **7/15** | **10/15** | **1/15** | **8/15** |
| | Average | 78.3% | 43.3% | 40.0% | 35.0% | **25.0%** | 36.6% | **3.3%** | **25.0%** |
| 33% TED | A1 | 5/15 | 3/15 | 0/15 | 2/15 | 3/15 | 7/15 | 4/15 | 4/15 |
| | A2 | 11/15 | 7/15 | **5/15** | 4/15 | 4/15 | 6/15 | 2/15 | 5/15 |
| | B1 | 13/15 | 9/15 | **5/15** | 6/15 | 8/15 | 3/15 | 2/15 | 5/15 |
| | B2 | **14/15** | **11/15** | 4/15 | **10/15** | **13/15** | **7/15** | **5/15** | **9/15** |
| | Average | 71.6% | 50.0% | 23.3% | 36.6% | **46.6%** | 38.3% | **21.6%** | **38.3%** |
| 66% TED | A1 | 4/15 | 3/15 | 1/15 | 0/15 | 4/15 | **7/15** | 5/15 | 3/15 |
| | A2 | **12/15** | 8/15 | **3/15** | 3/15 | 3/15 | 6/15 | 2/15 | 4/15 |
| | B1 | **12/15** | 8/15 | **3/15** | 6/15 | 5/15 | 4/15 | 4/15 | 5/15 |
| | B2 | **12/15** | **11/15** | **3/15** | **10/15** | **9/15** | **7/15** | **6/15** | **7/15** |
| | Average | 66.6% | 50.0% | 16.6% | 31.6% | 35.0% | 40.0% | **28.3%** | **31.6%** |
| BPE | A1 | 5/15 | 0/15 | 0/15 | 1/15 | 2/15 | 4/15 | 0/15 | 0/15 |
| | A2 | 10/15 | **9/15** | **6/15** | 4/15 | 4/15 | 1/15 | 0/15 | 0/15 |
| | B1 | 13/15 | 8/15 | 1/15 | 8/15 | 4/15 | 3/15 | 0/15 | 0/15 |
| | B2 | **14/15** | **9/15** | 5/15 | **9/15** | **9/15** | **8/15** | **1/15** | 0/15 |
| | Average | 70.0% | 43.3% | 20.0% | 36.6% | 31.6% | 26.6% | **1.6%** | **0.0%** |

Results from Table 5.9 underwent multiple Fisher Exact tests to evaluate if the differences observed between the error ratios from all different pairs of models are statistically significant. This analysis considered multiple 2x2 tables (one to each class of error), with rows defining the model and columns associated with the occurrence or not of some class of error. The significance level was set to 5%; thus, the null hypothesis was rejected whenever the $p$-value was lower than 0.05, representing a statistically significant difference between the pair of ratios under comparison.

The experiments considering the Fisher Exact Test will evaluate the effects of associating 2 different pairs of dimensions: first, the classes of errors and model variants, and later the classes of errors along with the categories of sentence complexity. Following this approach, at first all the models were compared with each other considering the percentage of errors among different classes of errors, to check if the observed differences are statistically significant. The same test was also conducted to compare the percentages of errors between the different classes of sentence complexity. Each experiment has a different purpose: the first experiment analyses mainly the hypothesis that different strategies adopted for producing a same model may influence on the error patterns, and the second experiment validates whether a change in the complexity of a sentence results in a different percentage of errors per class. The results for each case are discussed below.

### Error classes and model variants

This experiment considers 8 error classes and 4 variants, hence a total of 32 test case scenarios were created. The findings of this analysis are displayed in Table 5.10, considering only the cases to which a statistically significant difference was observed.

Table 5.10: Summary of the multiple Fisher Exact test outcomes for the error-model dimension pair, restricted to statistically significant pairs (see text)

| Error class | Benchmark model | Behaviour | Challenger model | $p$-value | Odds Ratio |
|---|---|---|---|---|---|
| *Meaning Deviation* | BT Transformer | it produces significantly **less** errors than | 33% Transformer | $22 \times 10^{-3}$ | 0.381 |
| *Repetition* | BT Transformer | it produces significantly **less** errors than | 33% Transformer | $4 \times 10^{-3}$ | 0.125 |
| *Repetition* | BT Transformer | it produces significantly **less** errors than | 66% Transformer | $3 \times 10^{-4}$ | 0.087 |
| *Repetition* | 33% Transformer | it produces significantly **more** errors than | BPE Transformer | $98 \times 10^{-5}$ | 16.319 |
| *Repetition* | 66% Transformer | it produces significantly **more** errors than | BPE Transformer | $4 \times 10^{-5}$ | 23.325 |
| <*unk*> | BT Transformer | it produces significantly **less** errors than | 33% Transformer | $17 \times 10^{-5}$ | 0.146 |
| <*unk*> | BT Transformer | it produces significantly **less** errors than | 66% Transformer | $249 \times 10^{-5}$ | 0.196 |
| <*unk*> | BT Transformer | it produces errors, which is not the case of | BPE Transformer | $57 \times 10^{-3}$ | $\infty$ |
| <*unk*> | 33% Transformer | it produces errors, which is not the case of | BPE Transformer | $174 \times 10^{-10}$ | $\infty$ |
| <*unk*> | 66% Transformer | it produces errors, which is not the case of | BPE Transformer | $701 \times 10^{-9}$ | $\infty$ |

The results indicate that models trained on reduced versions of the TED dataset are prone to reproducing the *Repetition* error more than the other techniques, as the related odds ratio are much higher than one in these cases. Also, the *Meaning Deviation* error seems to be better addressed by the Back Translation model, as compared to the 33% Transformer, which has the poorest BLEU score of them.

Regarding the <*unk*> error, naturally the BPE Transformer outperforms any other variant, as it replaces <*unk*> tokens for customized words. On top of that, the Back Translation Transformer also had a statistically significant lower percentage of <*unk*>, *Repetition*, and *Meaning Deviation* errors as compared to models trained on reduced versions of the TED dataset.

### Error classes and sentence complexity

This experiment has applied the hypothesis test to a combination of 8 error classes and 6 complexity pairs (considering A1, A2, B1 and B2), resulting in a total of 48 test cases. Speaking of the possible associations between the error patterns and the categories of sentence complexity, for seven out of eight error classes it was identified at least one statistically significant difference between two different classes of sentence complexity. This result corroborates with the hypothesis that a change in the sentence complexity may affect the type of error. In most error classes, except for the *Repetition* error, the complexities that are spaced more than 1 position further away (e.g. A1 and B1 or A2 and B2), the hypothesis test have rejected the null hypothesis. In the following bullets, the corresponding results are listed per error class:

1. *Reference Matching* error: Meaningful differences were reported by all tests, except for the pairs $A2|B1$ and $B1|B2$;
2. *Omission* error: Meaningful differences were reported by all tests, except for the pairs $A2|B1$ and $A2|B2$;
3. *Out of Context* error: Meaningful differences were reported by all tests, except for the pairs $A2|B1$, $A2|B2$ and $B1|B2$;
4. *Verb Tense* error: Meaningful differences were reported by all tests, except for the pair $A2|B1$;
5. *Meaning Deviation* error: Meaningful differences were reported by all tests, except for the pairs $A1|A2$ and $A2|B1$;
6. *Insertion* error: Meaningful differences were reported by all tests, except for the pairs $A2|B1$, $A1|B1$ and $A1|A2$;
7. *Repetition* error: No relevant difference is reported by the tests;
8. <*unk*> error: Meaningful differences were reported by all tests, except for the pairs $A2|B1$, $A1|A2$ and $A1|B1$.

In summary, these results signalize that our criterion for segmenting sentences in levels of complexity seems to be relevant for understanding error patterns, otherwise most of the previously discussed tests wouldn't point out any relevant difference. Also, some techniques have exhibited a very similar prevalence for some classes of errors, which may be interpreted as some degree of similarity. However, other multidimensional associations might be hidden in data in the previous analysis. In order to get further insights over the outlined hypothesis at the beginning of this chapter, we have explored some multivariate analysis tools as described in the following.

### 5.3.4 Hierarchical Clustering (HC) evaluation

Reputed as one of the most classical unsupervised clustering techniques, the HC algorithm is widely used for segmentation, recommendation, and even for exploratory data analysis. This algorithm can be roughly summarized as follows: it starts assigning each data point to a separate cluster, calculates the dissimilarity between each pair of them, and merges the pair which is most similar. This process is repeated until no further merges can be made. This algorithm also has a few hyperparameters to be settled by the practioner: the distance metric used to the proximity matrix calculations (which must be carefully chosen according to the type of data) and the linkage method, which defines how these clusters must be merged. Practical implementations often allow setting the number of clusters or the maximum dissimilarity (typically distance) measure. All these aspects will be discussed in the implementation section in the following.

#### Why use HC for a qualitative analysis?

Our main goal with using clustering was to spot patterns in the models' outcomes. As we aim for interpretability, *k-means* stands out as a nice competitor to HC. Table 5.11 shows a brief summary the major *k-means* and HC characteristics to support our decision making.

When compared to *k-means* Clustering, HC does not require one to define the number of clusters to partition data in advance, providing a family of nested clustering alternatives which can be freely chosen by the user. HC also disposes of intuitive visualization techniques, such as the Dendrogram, that may be helpful when choosing an appropriate number of clusters. Also, *k-means* may not fit adequately categorical data, as it considers the Euclidian distance between data and cluster centers. As a result, HC tends to be less biased than its competitor, despite the linkage parameter inserting some bias on cluster formation. Conversely, HC also

Table 5.11: Comparison of *k-means* and HC techniques

| Criteria | Hierarchical Clustering | *k-means* | Outcome |
|---|---|---|---|
| Bias | The linkage algorithm choice inserts a particular bias during merge operations. | It uses the Sum of Squares Error (SSE) and assumes normally distributed clusters. | HC has a higher flexibility regarding bias as compared to *k-means*, hence HC wins. |
| Scalability | $O(n^2)$ complexity[65] | $O(n)$ complexity | *k-means* wins |
| Methodology to find an adequate number of clusters | The final clustering can be found by setting a maximum dissimilarity threshold or by visually inspecting the Dendrogram. | It requires several experiments and setting a figure of merit for evaluation. | HC characteristics easy the task of finding an adequate number. |

has a higher computational cost $O(n^2)$[65] than *k-means* ($O(n)$).

In summary, the HC algorithm provides a set of linkage approaches that could be selected to better match the intrinsic structure of data. This turns the algorithm less susceptible to the choice of hiperparameters and more intuitive to interpret based on the Dendrogram visualization. It enables HC to be more flexible to match different kinds of data such as the categorical type, which is the case here. The flexibility in terms of metrics used for the merging criteria comes with the drawback of the cost of poor scalability. Since our dataset is quite small, this issue is irrelevant.

### The Hierarchical Clustering algorithm

Hierarchical clustering is a broad term that refers to 2 categories of algorithms: those that work bottom up (agglomerative) and the ones that work top-down (divisive), with the former being more commonly used. The main difference between both is that the agglomerative variant sequentially merges small clusters into bigger ones, whilst the divisive approach does the opposite, starting with all data combined in a single cluster and then splitting it according to some similarity metric.

To better understand the agglomerative variant, let us assume that the distance between 2 clusters $i$ and $j$ is defined by $d_{ij}$, with the cluster $i$ containing $N_i$ objects. Consider also a matrix $\mathbf{D}$ representing the set of all remaining distances between the clusters, and the set of $l$-dimensional vectors $\mathcal{X} = \{\mathbf{x}_i, i = 1, \ldots, N\}$ that are to be clustered. This matrix is updated in each iteration, and its elements are defined by a similarity or dissimilarity metric. The initial clustering $\mathbb{R}_0$ consists of $N$ clusters, each one containing a single element of $\mathcal{X}$. At the first step, the clustering $\mathbb{R}_1$ is produced. It contains $N-1$ sets, such that $\mathbb{R}_0 \subset \mathbb{R}_1$. This procedure keeps iterating until the final clustering $\mathbb{R}_{N-1}$ is obtained, which contains a single set ($\mathcal{X}$). The pseudocode for the HC algorithm is as expressed in Algorithm 2.

The distance used to calculate the matrix $\mathbf{D}$ is usually of free choice, but de-

**Algorithm 2** Pseudocode for Hierarchical Clustering (according to the linkage criterion), taken from THEODORIDIS and KOUTROUMBAS [66] (chapter 13)

```
• Initialization:
    – Choose $\mathbb{R}_0 = \{\mathbf{C}_i = \{\mathbf{x_i}\}, i = 1, \ldots, N\}$ as the initial clustering
    – $t = 0$
• Repeat:
    – $t = t + 1$
    – Among all possible pairs of clusters $(C_r, C_s)$ in $\mathbb{R}_{t-1}$ find the one, say $(C_i, C_j)$, such that
          $g(C_i, C_j) = \begin{cases} min_{r,s} g(C_r, C_s), & \text{if } g \text{ is a dissimilarity function} \\ min_{r,s} g(C_r, C_s), & \text{if } g \text{ is a similarity function} \end{cases}$   (5.1)
    – Define $C_q = C_i \cup C_j$ and produce the new clustering $\mathbb{R}_t = (\mathbb{R}_{t-1} - \{C_i, C_j\}) \cup \{C_q\}$
    – Until all vectors lie in a single cluster
```

pending on the linkage approach there might be constraints in terms of the options available (e.g. in the Ward linkage). Some possible linkage criteria are:

1. Single Linkage: It can be seen as a nearest neighbor modality of clustering, where the distance between two groups is defined as the distance between their two closest instances. Illustratively speaking, this algorithm has a tendency of sequentially adding instances to a single group, which may lead to a premature merging of groups when compared to other linkage approaches.

2. Complete Linkage: Here, the similarity between two groups is defined by the similarity value between the pair of instances that are the farthest from each other, also known as the furthest neighbor method. This method usually yields to clusters that are well-separated, but outliers can cause the merging of groups later than what might be considered as adequate.

3. Simple Average: It is also called as the weighted pair-group method, this algorithm defines the distance between two groups as the average distance between each pair of their members. This distance is weighted by the number of instances integrating each group.

4. Group Average: This linkage algorithm is also known as the unweighted pair-group method. It is similar to the simple average, differing by not assuming weighting constants.

5. Ward's Minimum Variance: This method assumes the Euclidean distance as the dissimilarity measure, forming groups so that the pooled within-group sum

of squares is minimized. At each step, the two clusters selected for fusion are those which result in the least increase in the pooled within-group sum of squares.

### Interpreting HC outcomes

Before starting to interpret the results derived with Hierarchical Clustering, one must guarantee that the resulting cluster appropriately reflects the intrinsic structure of data. This is usually performed by setting a maximum similarity threshold or using a metric to validate which Dendrogram cut would allow a better data segmentation.

The Dendrogram is a tree diagram that allows one to visualize each group merging operation performed by the algorithm, as well as the resulting dissimilarity of the new clusters formed. The Dendrogram is usually truncated depending on the number of data points each identified group has, and when that happens, the corresponding numbers in the $x$ axis come between brackets, otherwise they represent the identifier for that data point. Also, when truncated, in the $y$ axis the visualization will start with the dissimilarity threshold used for that operation, not 0. An untruncated version of the diagram is depicted in Figure 5.1, and it works by stretching lines from the $x$ axis for distinct heights (the $y$ axis). The height that they form a right angle represents the distance threshold where two cluster groups gets merged into other groups. Each vertical line may represent a single data point (which is the case) or a set of data points.

The number of clusters that would be formed at a particular cutoff can be easily determined in Figure 5.1 by drawing a horizontal line at that value in the $y$ axis and counting the number of horizontal lines intersected by it. For instance, one line drawn at a dissimilarity value of 1.5 would split the data into 4 different clusters. An adequate number of clusters can be found by visual inspection, where the cutoff line can be drawn at any line segment that represent a dissimilarity hop between group merges.

Regarding the use of similarity metrics for evaluating clustering, the Silhouette coefficient is one that is most commonly used. It consists in measuring how similar a data point is to its own cluster (cohesion) when compared to other clusters (separation). The Silhouette value $s(i)$ is defined for a data point $\mathbf{x}_i$ as

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}, \tag{5.2}$$

where $a(i)$ is the average distance between $\mathbf{x}_i$ and all the other data points from the cluster to which $\mathbf{x}_i$ belongs, and $b(i)$ is the minimum average distance from $i$ to the data points integrating other clusters. Whenever $a(i) = b(i)$ or the cluster related to

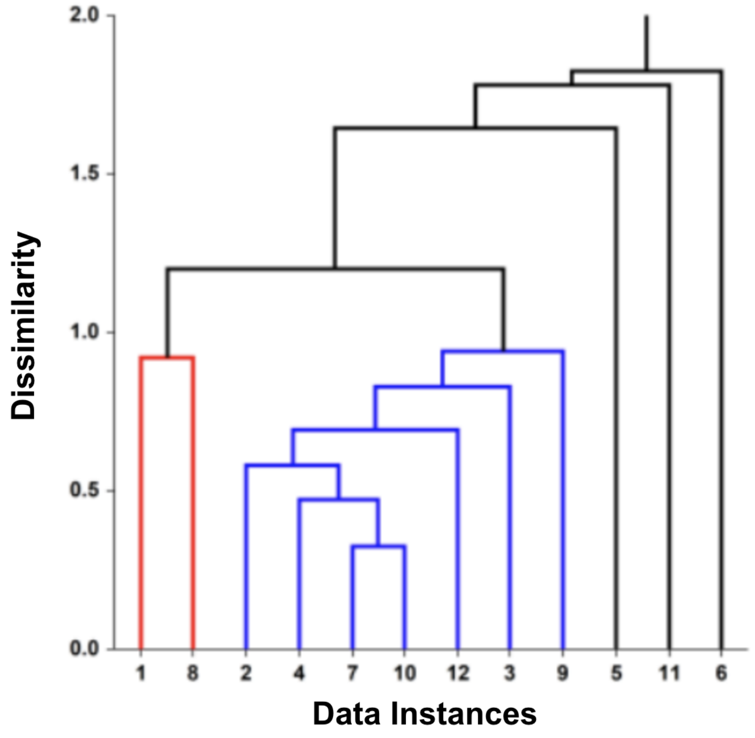


Figure 5.1: Dendrogram plot for an illustrative example.

$\mathbf{x}_i$ contains only a single data point, the Silhouette coefficient is set to 0, following the orientations of ROUSSEEUW [67], the creator of this metric.

The range of $s(i)$ is $[-1, 1]$. If it is near its maximum, close to 1, this means that $a(i)$ is much smaller than the smallest dissimilarity $b(i)$. Therefore, we can assume that $\mathbf{x}_i$ is well-clustered, as the distance between it and the other cluster's data points is very small. If $s(i)$ is about zero, this means that $a(i)$ and $b(i)$ are approximately equal, hence it is not clear whether $\mathbf{x}_i$ is better assigned to either cluster $A$ or $B$. The worst scenario takes place when $s(i)$ is close to $-1$, meaning that $a(i)$ is much larger than $b(i)$, so $\mathbf{x}_i$ is on the average closer to the cluster $B$ than to $A$. In this case, it makes more sense that $\mathbf{x}_i$ should have been assigned to $B$, an evidence that $\mathbf{x}_i$ is misplaced. Usually, when analyzing a cluster with the Silhouette coefficient, a single value is generated for the whole dataset, defined as the average of the Silhouette values computed for all data points.

The use of both approaches (Silhouette Coefficient and Dendrogram inspection) or even other literature alternatives is strongly recommended to find an appropriate number of clusters. It is also recommended to conduct an exploratory analysis over the clusters identified, firstly to check if all of them contain a reasonable number of data points, i.e., if they are representative. Then, descriptive metrics can be derived to evaluate if the cluster unveils multidimensional similarities between its points, or if it is unclear that they are grouped. If interpreting the cluster outcomes is hard, then maybe the number of clusters or the algorithm chosen is not the most adequate.

## Analysis of models and classes of errors

To perform this experiment, all the 15 sentences extracted to match each one of the 4 language complexity classes were labelled with a binary variable indicating the absence or presence of one or more classes of error described in Table 5.6. Therefore, each of the 4 models under analysis (described in Section 5.3.3) involves 60 sentences, totalling 240. In practical terms, we are speaking of a dataset that contains one binary column for each type of error (hence, 8 columns), along with other binary columns that represent the models involved in the translation and columns that represent the complexity to which each sentence was classified. To avoid the evaluation of 3 dimensions simultaneously, we only analysed two possible iteractions between the pairs of interest: models versus classes of errors and classes of sentence complexity versus classes of errors. The former one will be presented in this subsection, and the latter in the sequence.

Hierarchical Clustering considered the Manhattan distance, appropriate for categorical data, which is the case here since the columns representing the error classes are binary. The Linkage criterion was chosen after running some experiments with different linkage modalities and numbers of clusters, with the goal of better understanding how the clusters are composed. The complete linkage approach was chosen since the other alternatives resulted in many clusters representing few data points, an issue that could potentially hinder our analysis.

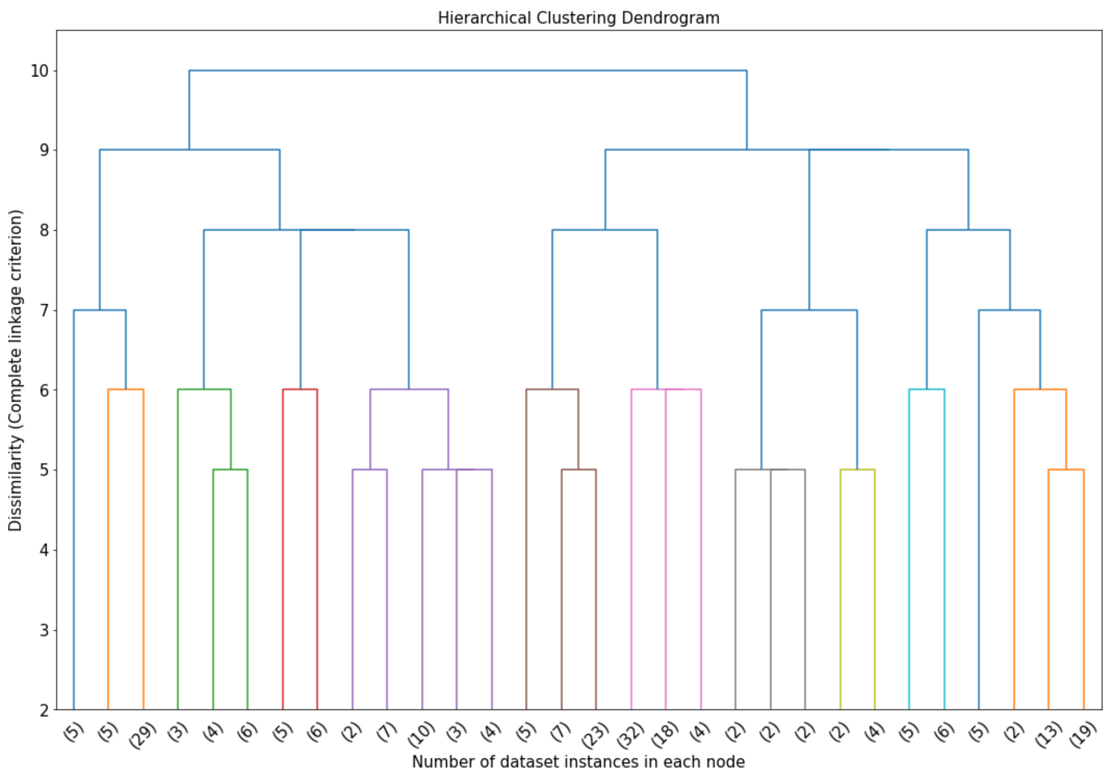


Figure 5.2: Dendrogram related to both HC experiments.

Recall that in this case the HC algorithm has fit a dataset with 240 distinct sentences, which represent its rows with an unique identifier, and 8 binary columns representing the presence of an error class in the translated sentence. The models responsible for producing the translations were kept in a separate data structure, but with the same identifier, so after the algorithm produces its result, we could easily join and understand how the models were clustered. The Dendrogram corresponding to the algorithm applied with complete linkage can be seen in Figure 5.2, with a dissimilarity range starting from 2 since it was truncated.

Inspecting the graph induces the reader to a doubt regarding where to put the cutoff, since the groups' dissimilarities are somewhat similar at lower cluster levels. This is a consequence of using categorical data. It looks like the distances are binned and the number of clusters could not be inferred by graphical inspection in this case. Despite such limitations, there seem to have some options such as 2, 5 and 9 clusters. Additionally, we computed the Silhouette coefficient for some linkage approaches and a number of cluster groups ranging from 3 to 10. The resulting coefficients are arranged in Table 5.12.

Table 5.12: Silhouette coefficients per linkage approach and number of clusters.

| Linkage | Number of clusters | Silhouette coefficient |
|---|---|---|
| Complete | 3 | 0.1415 |
| Complete | 4 | 0.1442 |
| Complete | 5 | **0.2320** |
| Complete | 6 | 0.2199 |
| Complete | 7 | 0.2368 |
| Complete | 8 | 0.2665 |
| Complete | 9 | 0.2931 |
| Complete | 10 | 0.2930 |

Considering the coefficients summarized in Table 5.12 and also identifying the number of sentences per cluster, we concluded that the most appropriate cluster involves 5 groups. Note that if the number of clusters is increased, the complexity of interpreting the results also increases significantly, without a meaningful compensation from the coefficient side. For instance, assuming 10 groups would lead to an increase of only 0.061 points (about 26%), whilst adopting 7 groups would lead to an increase of 0.048 (about 2%). On the other hand, when choosing a number of clusters as small as 3, the decrease in the coefficient would be of 0.0905 (about 39%), which is too high.

After setting the number of groups to five, some descriptive metrics were generated for each cluster to help in understanding how the dataset was segmented: the percentual incidence of each class of error per cluster, the overall percentage of errors per cluster, and the number of sentences belonging to each cluster. All these metrics are exposed in Table 5.13.

Table 5.13: Prevalence of the classes of errors and the models involved per cluster.

| Dimension Name | Dimension Value | Cluster 0 | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|---|---|
| Error class | *Reference Matching* | 86.0% | 74.0% | 97.8% | 41.7% | 0.0% |
| | *Omission* | 46.0% | 58.0% | 48.3% | 75.0% | 20.5% |
| | *Out of Context* | 16.0% | 2.0% | 18.0% | 100.0% | 12.8% |
| | *Verb Tense* | 98.0% | 52.0% | 1.0% | 58.3% | 2.6% |
| | *Meaning Deviation* | 18.0% | 62.0% | 29.2% | 91.7% | 15.4% |
| | *Insertion* | 26.0% | 72.0% | 23.6% | 50.0% | 23.1% |
| | *Repetition* | 12.0% | 40.0% | 2.2% | 8.3% | 10.3% |
| | *<unk>* | 2.0% | 80.0% | 14.6% | 16.7% | 2.6% |
| Model Name | 33% TED Transformer | 22.0% | 30.0% | 23.6% | 33.3% | 23.1% |
| | 66% TED Transformer | 22.0% | 38.0% | 20.2% | 16.7% | 25.6% |
| | BPE Transformer | 34.0% | 4.0% | 27.0% | 41.7% | 30.8% |
| | BT Transformer | 22.0% | 28.0% | 29.2% | 8.3% | 20.5% |
| Overall error prevalence per cluster | | 38.0% | 55.0% | 29.4% | 55.2% | 10.9% |
| Total of sentences per cluster | | 50 | 50 | 89 | 12 | 39 |

Table 5.13 shows that the least representative cluster has the highest incidence of errors, whilst still there is a cluster with 50 sentences that has an incidence which is high and very close to the previous number. Some errors like *Omission* are more evenly spread among all clusters, whilst others like *Reference Matching* tend to vary a lot. These are the first impressions of the data, a more detailed analysis combining everything will be given after this table is complemented with more results.

Another goal of the cluster analysis is to understand what are the underlying patterns that motivated the algorithm to group sentences together, specially in terms of the types of errors that the model variants produce. Aiming to better derive similarities, we have performed an experiment summarizing the classes of errors observed in each cluster, considering only pairs of cluster and errors with an overall prevalence higher than 50%. The expectation here is to list the strongest associations between the classes of errors and the models, reduce the subjectivity when interpreting the results, and derive practical conclusions about the error weaknesses or strengths of each model. To help in performing such analysis, the error prevalence for each model was ranked per cluster and the results can be found in Table 5.14, where if two or more models reach the same error prevalence in one cluster, they will be tied at the same ranking position. In this table, each row represents the group of sentences of one cluster that were generated by one type of model.

Now each cluster involved in the model-error experiments will be commented, considering the previous Tables 5.13 and 5.14. Although the cluster highlights and the ranking of error class prevalences per model are interesting, we still have to interpret qualitatively the bidimensional relationships by analysing this data to be able to understand the whole picture. In the bullet points below, statements and conclusions about the data previously presented in the tables are made. They aim to clarify some of the relationships that the HC algorithm helped to spot:

Table 5.14: Rank of the most frequent errors per model in each cluster

| Row | Error class | Prevalence | Sentences | Model Name | Rank | Cluster |
|---|---|---|---|---|---|---|
| 1 | *Verb Tense* | 100% | 11 | 33% TED Transformer | 1 | 0 |
| 2 | *Verb Tense* | 100% | 11 | 66% TED Transformer | 1 | 0 |
| 3 | *Verb Tense* | 100% | 17 | BPE Transformer | 1 | 0 |
| 4 | *Verb Tense* | 91% | 11 | BT Transformer | 2 | 0 |
| 5 | *Reference Matching* | 91% | 11 | 33% TED Transformer | 1 | 0 |
| 6 | *Reference Matching* | 82% | 11 | 66% TED Transformer | 3 | 0 |
| 7 | *Reference Matching* | 88% | 17 | BPE Transformer | 2 | 0 |
| 8 | *Reference Matching* | 82% | 11 | BT Transformer | 3 | 0 |
| 9 | *Insertion* | 87% | 15 | 33% TED Transformer | 2 | 1 |
| 10 | *Insertion* | 68% | 19 | 66% TED Transformer | 3 | 1 |
| 11 | *Insertion* | 100% | 2 | BPE Transformer | 1 | 1 |
| 12 | *Insertion* | 57% | 14 | BT Transformer | 4 | 1 |
| 13 | *Verb Tense* | 40% | 15 | 33% TED Transformer | 3 | 1 |
| 14 | *Verb Tense* | 37% | 19 | 66% TED Transformer | 4 | 1 |
| 15 | *Verb Tense* | 100% | 2 | BPE Transformer | 1 | 1 |
| 16 | *Verb Tense* | 79% | 14 | BT Transformer | 2 | 1 |
| 17 | *Meaning Deviation* | 87% | 15 | 33% TED Transformer | 2 | 1 |
| 18 | *Meaning Deviation* | 53% | 19 | 66% TED Transformer | 3 | 1 |
| 19 | *Meaning Deviation* | 100% | 2 | BPE Transformer | 1 | 1 |
| 20 | *Meaning Deviation* | 43% | 14 | BT Transformer | 4 | 1 |
| 21 | <unk> | 87% | 15 | 33% TED Transformer | 1 | 1 |
| 22 | <unk> | 79% | 19 | 66% TED Transformer | 3 | 1 |
| 23 | <unk> | 0% | 2 | BPE Transformer | 4 | 1 |
| 24 | <unk> | 86% | 14 | BT Transformer | 2 | 1 |
| 25 | *Omission* | 60% | 15 | 33% TED Transformer | 2 | 1 |
| 26 | *Omission* | 53% | 19 | 66% TED Transformer | 3 | 1 |
| 27 | *Omission* | 50% | 2 | BPE Transformer | 4 | 1 |
| 28 | *Omission* | 64% | 14 | BT Transformer | 1 | 1 |
| 29 | *Reference Matching* | 60% | 15 | 33% TED Transformer | 4 | 1 |
| 30 | *Reference Matching* | 68% | 19 | 66% TED Transformer | 3 | 1 |
| 31 | *Reference Matching* | 100% | 2 | BPE Transformer | 1 | 1 |
| 32 | *Reference Matching* | 93% | 14 | BT Transformer | 2 | 1 |
| 33 | *Reference Matching* | 95% | 21 | 33% TED Transformer | 3 | 2 |
| 34 | *Reference Matching* | 100% | 18 | 66% TED Transformer | 1 | 2 |
| 35 | *Reference Matching* | 100% | 24 | BPE Transformer | 1 | 2 |
| 36 | *Reference Matching* | 96% | 26 | BT Transformer | 2 | 2 |
| 37 | *Meaning Deviation* | 100% | 4 | 33% TED Transformer | 1 | 3 |
| 38 | *Meaning Deviation* | 100% | 2 | 66% TED Transformer | 1 | 3 |
| 39 | *Meaning Deviation* | 100% | 5 | BPE Transformer | 1 | 3 |
| 40 | *Meaning Deviation* | 0% | 1 | BT Transformer | 2 | 3 |
| 41 | *Omission* | 100% | 4 | 33% TED Transformer | 1 | 3 |
| 42 | *Omission* | 50% | 2 | 66% TED Transformer | 3 | 3 |
| 43 | *Omission* | 60% | 5 | BPE Transformer | 2 | 3 |
| 44 | *Omission* | 100% | 1 | BT Transformer | 1 | 3 |
| 45 | *Verb Tense* | 100% | 4 | 33% TED Transformer | 1 | 3 |
| 46 | *Verb Tense* | 50% | 2 | 66% TED Transformer | 2 | 3 |
| 47 | *Verb Tense* | 40% | 5 | BPE Transformer | 3 | 3 |
| 48 | *Verb Tense* | 0% | 1 | BT Transformer | 4 | 3 |
| 49 | *Out of Context* | 100% | 4 | 33% TED Transformer | 1 | 3 |
| 50 | *Out of Context* | 100% | 2 | 66% TED Transformer | 1 | 3 |
| 51 | *Out of Context* | 100% | 5 | BPE Transformer | 1 | 3 |
| 52 | *Out of Context* | 100% | 1 | BT Transformer | 1 | 3 |

1. Cluster 0: This cluster contains 50 sentences (20.8% of the total). BPE is associated with the biggest share of sentences (34%), with the other models having 22%, each. The overall percentage of errors is 38%, ranking it in the 3rd place (exactly in the middle) when compared to the others. *Verb Tense* and *Reference Matching* stand out with a percentual incidence of 98% and 86% respectively, with the least frequent errors being <*unk*> and *Repetition* with 2% and 12%, each.

   It seems that the cluster 0 is mostly represented by BPE due to its higher sentence share. It also concentrates the sentences with the highest prevalence of *Verb Tense* and *Reference Matching* errors, since all models made these errors in more than 80% of their own sentences, as shown by the Prevalence column in the first 8 rows. There are signals in this cluster that these errors are quite common to all models, specially to the dominant model (BPE). This cluster also has the lowest prevalence of the <*unk*> error as shown in Table 5.13, which is somewhat coherent with the fact that BPE slightly dominates this cluster, since the model is immune to this error.

2. Cluster 1: It contains 50 sentences (20.8% of the total). The 66% Transformer has the highest share of sentences (38%), followed by the 33% Transformer with 30%, BT with 28% and BPE with 4% (2 sentences). The overall percentage of prevalence of errors is 55%, ranking it in the 2nd place. The <*unk*> (80%), *Reference Matching* (74%) and *Insertion* (72%) errors stand out as the most frequent, whilst the *Out of Context* error almost didn't happen (2%) and also the low profile *Repetition* error (40%) are the least occurring ones.

   Although this is the cluster with the second highest prevalence of errors, the gap is narrow (0.21%) when compared to the first place. It has a strong participation (68%) of the Transformers trained on a reduced dataset (33% and 66%), with BPE and BT models associated with less than a third of the sentence share, hence being mostly represented by the former models. These 2 dominating models demonstrate a high prevalence of errors in Table 5.14, such as *Insertion* (rows $9-12$), *Meaning Deviation* (rows $17-20$) and <*unk*> errors (between rows $9$ – $12$), losing to BT in the ranking when talking of *Omission* and *Verb Tense*, in the remaining rows related to the same cluster. The difference on the cluster pattern to the first one is clear: it has the highest prevalence of the *Repetition* error, and the prevalence of the <*unk*> error is much more expressive. The main characteristic that sets them apart is that in the first cluster BPE predominates. The differences on the error patterns between clusters corroborates with the hypothesis that the model plus low-resource technique choice indeed directly impacts on the translation behaviour.

3. Cluster 2: It is the biggest cluster with 89 sentences (37% of the total). The models share almost the same quantity of sentences, with BT and BPE Transformers being the most occurring (29% and 27%). The overall percentage of errors is 29%, being the second with less mistakes. *Reference Matching* is almost ubiquitous (98%), whereas curiously the other errors have all less than 50% of prevalence. *Verb Tense* and *Repetition* errors almost don't happen, with 2% and 1%, respectively.

   The most representative cluster is also fairly equally distributed, and the high prevalence of *Reference Matching* in this sample, ranging from 95% to 100% in the ranking (rows $33-36$), indicates that this is an error that these models can hardly avoid. In fact, this error is the most common among all the clusters. It is a well-known flaw on the standard NMT models' evaluation process, and it could be probably addressed by modifying the methodology (such as using multiple references, provided by several human translators). Moreover, for this cluster, the models pratically do not perform the *Verb Tense* or the *Repetition* errors.

4. Cluster 3: It is a small cluster with 12 errors (5% of the total), with BPE responsible for most of the sentences (42%), followed by the 33% Transformer (33%), and a very small contribution from BT (8%). The overall percentage of errors is the highest, with 55.2%, where the *Out of Context* error is unanimous (100%), followed by *Meaning Deviation* (92%) and *Omission* (75%). The least frequent errors are *Repetition* (8%) and *<unk>* (16%).

   This cluster has a small share of the total sentences, but it concentrates the most difficult ones for all models, curiously with a higher share related to BPE. Noteworthy, the percentages related to the most frequent errors of this cluster strongly differentiates from those observed with other clusters: from the *Meaning Deviation* error (91.67% vs 42.85% on average), *Out of Context* (100% vs 29.76% on average) and *Omission* (75% vs 49.56%). Anticipating one result from the subsequent analysis of the sentence complexity effects, which will be reported in the next section, this cluster has about 75% of its sentences belonging to B2 and B1, whilst the others posess a lower share (from 66% to 7.6%) of sentences of these higher complexities, which explains the higher percentage of errors observed here. However, since this cluster contains only 5% of all the sentences, the extension and relevance of the conclusions derived here are somewhat limited.

5. Cluster 4: It corresponds to a medium-sized cluster including 39 sentences and encompassing the rest of the dataset, representing 16% of all translations. The BPE Transformer is the most influential model contributing with 30% of

the sentences, whilst the remaining ones are approximately equally distributed among alternative models, with BT being the least frequent (20.5%). The percentage of overall errors in this cluster is low (10.9%), with no errors occurring more than 50% of the sentences. The *Insertion* error is the most frequent one (23%), whilst *Reference Matching* does not happen at all, followed by <*unk*> and *Verb Tense*, both with 2%.

This cluster contrasts with the last one since the low prevalence of errors observed seems to be the lower complexity of the involved sentences, from which 84.6% are A1. This may explain why no errors happen in more than 50% of sentences. Regarding the scarcity of errors, the error classes that drive more attention are *Reference Matching* (0% vs 59.88% on average), *Verb Tense* (2.56% vs 42.38 on average) and <*unk*> (2.56% vs 23.17% on average).

The results of this experiment, despite the sample limitations and constraints established by the classes of errors created, sheds some light on the relations between models and patterns of errors. There is some evidence that the reduced models tend to perform the *Repetition* and <*unk*> errors, which may be associated with a lack of vocabulary. Errors like *Reference Matching* seem to happen independently of the technique, except for the case of Cluster 4 in Table 5.13, whilst *Verb Tense* is also quite common, but it is usually directly correlated with a higher representivity of BPE in a given cluster, as in Clusters 0 and 3. Whenever the Transformers trained on reduced TED 33% and 66% appear, their share of sentences in a cluster tends to be the same (as in clusters 1, 3, 5 and somewhat 1), which can be an evidence that the algorithm sees some correlation between their error patterns.

### Analysis of sentence complexity and classes of errors

The rationale from the last section was applied for experimentally inferring possible relations between sentence complexity and classes of error. Therefore, the qualitative results considered the same cluster but visited with a different label, the sentence complexity instead of the model variant. Table 5.15 depicts the same metrics adopted in Table 5.13, repeating the error class incidences per cluster, but now comparing with sentence complexities.

Looking how the sentences of different complexities are split among the clusters, a first look indicates that the overall percentage of errors is somewhat correlated with the complexity that dominates in that cluster. If we create a matrix whose rows are the percentage of sentences belonging to a specific cluster and columns represent the complexities, the Pearson correlation between each of the adjacent complexity pairs (A1|A2, A2|B1, B1|B2) will indicate how similar their distribution is. The module of that correlation will also outline if the information of complexity is relevant to

Table 5.15: Error class and complexity prevalence per cluster.

| Dimension Name | Dimension Value | Cluster 0 | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|---|---|
| Error class | *Reference Matching* | 86% | 74% | 97.75% | 41.67% | 0% |
| | *Omission* | 46.0% | 58.0% | 48.3% | 75.0% | 20.5% |
| | *Out of Context* | 16.0% | 2.0% | 18.0% | 100.0% | 12.8% |
| | *Verb Tense* | 98.0% | 52.0% | 1.0% | 58.3% | 2.6% |
| | *Meaning Deviation* | 18.0% | 62.0% | 29.2% | 91.7% | 15.4% |
| | *Insertion* | 26.0% | 72.0% | 23.6% | 50.0% | 23.1% |
| | *Repetition* | 12.0% | 40.0% | 2.2% | 8.3% | 10.3% |
| | *<unk>* | 2.0% | 80.0% | 14.6% | 16.7% | 2.6% |
| CEFR Complexity | A1 | 4.0% | 16.0% | 19.1% | 0.0% | 84.6% |
| | A2 | 32.0% | 20.0% | 31.5% | 25.0% | 7.7% |
| | B1 | 42.0% | 8.0% | 32.6% | 25.0% | 7.7% |
| | B2 | 22.0% | 56.0% | 16.9% | 50.0% | 0.0% |
| Overall error prevalence per cluster | | 38.0% | 55.0% | 29.4% | 55.2% | 10.9% |
| Total sentences per cluster | | 50 | 50 | 89 | 12 | 39 |

separate the sentences. The value of the correlation between A1 and A2 is 0.85 in module, for A2 and B1 it is 0.88; and for B1 and B2, 0.12. Most of the combinations show a high correlation, hence we could consider that the segmentation performed is somewhat accurate. B2 is an exception, but it is also where the biggest complexity gap lies, as we had to remove the C1 and C2 levels due to sample constraints, thus we will consider that this issue is acceptable for our analysis purposes.

Analogously to the analysis performed in Section 5.3.4, a ranking of the complexities per cluster that had at least one class of error with more than 50% of prevalence has also been generated and is reported in Table 5.16. The data depicted in this table will be analysed for each cluster as follows:

1. Cluster 0: Most of its sentences are of higher complexities (66% of B1 and B2), with only 4% belonging to A1. This higher sentence complexity is in line with the percentage of errors, about 38%. Curiously, for A1 sentences, there is no *Reference Matching* errors, whilst for the remaining complexities the observed percentages are all greater than 85%. Regarding the *Verb Tense* error, which shows a high prevalence in this cluster, it seems that its appearance is uncorrelated to the sentence complexities.

2. Cluster 1: This cluster has a bias on the direction of high complexity sentences (64% of B1 and B2), a bit similar to the previous cluster, but with a higher share related to A1 (16%). It possesses the second highest percentage of errors per sentence (55%), much higher than Cluster 1, despite having a similar complexity distribution. A look in the cluster rank tells that error classes do not change their percentage of errors proportionally to the complexity of sentences, for instance with the *<unk>* (rows 9−12) and *Insertion* errors (rows 13−16). As one can observe, the percentages does not follow a monotonic trend

Table 5.16: Rank of most frequent errors per complexity in each cluster

| Row | Error class | Prevalence | Sentences | Complexity | Rank | Cluster |
|---|---|---|---|---|---|---|
| 1 | *Reference Matching* | 100% | 11 | B2 | 1 | 0 |
| 2 | *Reference Matching* | 88% | 16 | A2 | 2 | 0 |
| 3 | *Reference Matching* | 86% | 21 | B1 | 3 | 0 |
| 4 | *Reference Matching* | 0% | 2 | A1 | 4 | 0 |
| 5 | *Verb Tense* | 100% | 2 | A1 | 1 | 0 |
| 6 | *Verb Tense* | 100% | 21 | B1 | 1 | 0 |
| 7 | *Verb Tense* | 100% | 11 | B2 | 1 | 0 |
| 8 | *Verb Tense* | 94% | 16 | A2 | 2 | 0 |
| 9 | *<unk>* | 100% | 4 | B1 | 1 | 1 |
| 10 | *<unk>* | 90% | 10 | A2 | 2 | 1 |
| 11 | *<unk>* | 75% | 8 | A1 | 3 | 1 |
| 12 | *<unk>* | 75% | 28 | B2 | 3 | 1 |
| 13 | *Insertion* | 80% | 10 | A2 | 1 | 1 |
| 14 | *Insertion* | 75% | 4 | B1 | 2 | 1 |
| 15 | *Insertion* | 71% | 28 | B2 | 3 | 1 |
| 16 | *Insertion* | 63% | 8 | A1 | 4 | 1 |
| 17 | *Meaning Deviation* | 68% | 28 | B2 | 1 | 1 |
| 18 | *Meaning Deviation* | 60% | 10 | A2 | 2 | 1 |
| 19 | *Meaning Deviation* | 50% | 8 | A1 | 3 | 1 |
| 20 | *Meaning Deviation* | 50% | 4 | B1 | 3 | 1 |
| 21 | *Omission* | 82% | 28 | B2 | 1 | 1 |
| 22 | *Omission* | 50% | 4 | B1 | 2 | 1 |
| 23 | *Omission* | 30% | 10 | A2 | 3 | 1 |
| 24 | *Omission* | 13% | 8 | A1 | 4 | 1 |
| 25 | *Reference Matching* | 100% | 4 | B1 | 1 | 1 |
| 26 | *Reference Matching* | 89% | 28 | B2 | 2 | 1 |
| 27 | *Reference Matching* | 50% | 8 | A1 | 3 | 1 |
| 28 | *Reference Matching* | 40% | 10 | A2 | 4 | 1 |
| 29 | *Verb Tense* | 82% | 28 | B2 | 1 | 1 |
| 30 | *Verb Tense* | 50% | 4 | B1 | 2 | 1 |
| 31 | *Verb Tense* | 13% | 8 | A1 | 3 | 1 |
| 32 | *Verb Tense* | 0% | 10 | A2 | 4 | 1 |
| 33 | *Reference Matching* | 100% | 28 | A2 | 1 | 2 |
| 34 | *Reference Matching* | 100% | 15 | B2 | 1 | 2 |
| 35 | *Reference Matching* | 97% | 29 | B1 | 2 | 2 |
| 36 | *Reference Matching* | 94% | 17 | A1 | 3 | 2 |
| 37 | *Meaning Deviation* | 100% | 3 | A2 | 1 | 3 |
| 38 | *Meaning Deviation* | 100% | 6 | B2 | 1 | 3 |
| 39 | *Meaning Deviation* | 67% | 3 | B1 | 2 | 3 |
| 40 | *Omission* | 100% | 3 | B1 | 1 | 3 |
| 41 | *Omission* | 83% | 6 | B2 | 2 | 3 |
| 42 | *Omission* | 33% | 3 | A2 | 3 | 3 |
| 43 | *Out of Context* | 100% | 3 | A2 | 1 | 3 |
| 44 | *Out of Context* | 100% | 3 | B1 | 1 | 3 |
| 45 | *Out of Context* | 100% | 6 | B2 | 1 | 3 |
| 46 | *Verb Tense* | 83% | 6 | B2 | 1 | 3 |
| 47 | *Verb Tense* | 67% | 3 | B1 | 2 | 3 |
| 48 | *Verb Tense* | 0% | 3 | A2 | 3 | 3 |

when the complexity gradually increases, with some of those errors even having a smaller prevalence among B2 sentences, as per the aforementioned rows $9-16$. Conversely, some errors do attend the approximately monotonic trend, namely *Reference Matching*, *Omission* and *Meaning Deviation*. In *Reference Matching*, curiously the complexities A2 and B1 have the lowest and highest prevalence, respectively.

3. Cluster 2: It's the largest cluster with 89 sentences, wherein 49% of them are of the challenging complexities (B1 and B2). It has the second smallest overall error percentage (29.35%), in coherence with the smaller number of higher complexity sentences. Only the *Reference Matching* error shows more than 50% of prevalence, it seems to be to be uncorrelated with sentence complexity, since both A2 and B2 have the same percentage of errors.

4. Cluster 3: This tiny cluster includes 12 sentences, with 75% of them belonging to B2 and B1 complexities, the highest among all, which is potentially explained by its highest overall error percentage (55.2%). Curiously, this cluster did not include A1 sentences, which makes sense if the goal was to concentrate on more complicated sentences. Some error classes in the rank appear to not increase their prevalences with an increase in the CEFR complexity, namely *Meaning Deviation* (rows $37-39$) and *Out of Context* (rows $43-45$). On the other hand, *Verb Tense* and *Omission* errors show a relevant difference for A2 and B2 sentences, as expected.

5. Cluster 4: The last cluster contains 39 sentences and shows the lowest overall percentage of errors, about 10.9%. This is somewhat compatible to the fact that 84.6% of the sentences in this cluster are A1 level, whilst none belong to B2. No class of error has shown a prevalence higher than 50%. It seems that in this case the algorithm preferred to agglomerate easier sentences that are less prone to translation errors.

A summary of the previous findings are as follows. The algorithm seems to have agglomerated sentences with similar complexity in some clusters, specially the ones enumerated as 3 and 4, despite not disposing of this information beforehand. Also, for a significant share of sentences the percentage of errors also increases with the sentence complexity, which is an evidence in favour of the criteria considered when ranking the sentences according to the CEFR levels, which seems to be appropriated.

The diversity and share of errors also differs a lot from cluster to cluster, depending on the sentence complexity share. For instance the *Verb Tense* errors seem to be influenced by the level of complexity of the sentences when looking to Table 5.16, except for the cluster 0. The same happens with the *Omission* error, for all clusters. Nonetheless, this analysis confirmed that the sentence complexity or model chosen are not isolated factors contributing to the error patterns, acting simultaneously in a bidimensional manner. This fact, not evidenced in the preceeding analysis (Section 5.3.3), was hopefully captured by HC.

### 5.3.5 Correspondence Analysis (CA) experiment

CA is an exploratory multivariate analysis technique that is able to process a categorical $N$-dimensional table, summarizing its content in a $N$-dimensional coordinate system. These coordinates can be plotted into a $N$-dimensional figure, which can help one to infer relations between different variables from the table, expressed in its corresponding rows and columns. The most common use case involves only 2 dimensions, wherein a co-occurrence matrix is converted to a 2-D mapping, and its rows and columns are depicted as points. The table containing the number of common occurrences of a given pair of nominal variables is named contingency matrix, and may have 2 dimensions or $N$ dimensions, the first called Simple Correspondence Analysis; and the latter, Multiple Correspondence Analysis.

#### Why use CA for a qualitative analysis?

CA is popularly considered to be as a Principal Components Analysis (PCA) variant suitable for categorical data, as mentioned in GREENACRE and HASTIE [68]. The coordinates generated by the algorithm are analogous to the PCA factors (used for continuous data), but the key technical difference is that CA assesses the Chi-square distances between two or more dimensions when testing their independence, instead of the total variance, as usually performed by PCA [69].

This algorithm also benefits from some properties of Singular Value Decomposition (SVD), such as the capability of generating linear transformations to extract meaningful features of multidimensional data, similar to PCA. Assuming that the algorithm is able to represent the table data with a sufficient explained variance, it is possible to generate a plot that enables a geometric interpretation of the existing relations between the table dimensions, and use the Chi-square statistic to determine the distances between the data points. The potential to extract relevant relations depends on whether the explained variance is considered to be significant or not. There is no hard rule to describe what must be considered as a good explained variance, it depends on the rigor of the analysis. Values above 80% considering the principal components are considered satisfactory and above 90% are assumed to be an almost ideal fit.

The CA algorithm was applied to the same qualitative dimensions exploited in the previous experiments (models and classes of errors or sentence complexities and classes of errors). Literature shows that the conclusions derived with the simple

CA are straightforward to interpret, and assessing its effectiveness and reliability is possible through explained variance. Hence, we believe that this tool has potential to corroborate, complement or even extend the findings derived from the previous analysis.

**The Correspondence Analysis algorithm**

The step by step process to implement CA will be mentioned here, but a further theoretical description and motivation of this algorithm goes beyond the scope of this work. For more details the reader is referred to IZENMAN [69]. To allow a better understanding of the *modus operandi* of this algorithm, a pseudocode is provided here, heavily inspired in [70] [69].

A co-occurrence matrix $\mathbf{K}$, also referred to as two-way contingency matrix in the literature, is a matrix whose cells enumerate the co-occurrences of specific row-column combinations. Consider the matrix $\mathbf{K}$ with dimensions $r \times s$, where $r$ is the number of rows and $s$ is the number of columns. Each row contains one category of some arbitrary dimension $a$ (the class of error, for instance), from a total of $r$ categories. The same must be assumed for the columns, assuming a total of $s$ categories from some dimension $b$. This matrix constitutes the input of the CA algorithm, which involves the following subsequent processing steps:

1. Assure that $\mathbf{K} \in \mathbb{Z}_+^{*(r \times s)}$, i.e., that all elements of $\mathbf{K}$ must be non-negative and none of the row or column entries sum up zero.

2. Compute the profile matrix $\mathbf{P}$ whose entries are given by

$$p_{ij} = \frac{k_{ij}}{\sum_i \sum_j k_{ij}}. \tag{5.3}$$

   The expected format for the matrix $\mathbf{P}$ is given by Table 5.17:

Table 5.17: General matrix $\mathbf{P}$ format, after normalizing $\mathbf{K}$

| Row Variable | $B_1$ | $B_2$ | $\dots$ | $B_j$ | $\dots$ | $B_s$ | Row Total |
|---|---|---|---|---|---|---|---|
| $A_1$ | $p_{11}$ | $p_{12}$ | $\dots$ | $p_{1j}$ | $\dots$ | $p_{1s}$ | $p_{1+}$ |
| $A_2$ | $p_{21}$ | $p_{22}$ | $\dots$ | $p_{2j}$ | $\dots$ | $p_{2s}$ | $p_{2+}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $A_i$ | $p_{i1}$ | $p_{i2}$ | $\dots$ | $p_{ij}$ | $\dots$ | $p_{is}$ | $p_{i+}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $A_r$ | $p_{r1}$ | $p_{r2}$ | $\dots$ | $p_{rj}$ | $\dots$ | $p_{rs}$ | $p_{r+}$ |
| Column Total | $p_{+1}$ | $p_{+2}$ | $\dots$ | $p_{+j}$ | $\dots$ | $p_{+s}$ | 1 |

3. The totals sums of both the rows and columns of $\mathbf{P}$ can also be embedded in the vectors $\mathbf{r}$ and $\mathbf{c}$, respectively, defined as

$$\mathbf{P1}_s = \begin{bmatrix} p_{1+} \\ \vdots \\ p_{r+} \end{bmatrix} = \mathbf{r}, \tag{5.4}$$

and

$$\mathbf{P}^T\mathbf{1}_r = \begin{bmatrix} p_{+1} \\ \vdots \\ p_{+s} \end{bmatrix} = \mathbf{c}, \tag{5.5}$$

which are also known as the average row and column profiles. Consider now the generation of the diagonal matrices $\mathbf{D}_r$ and $\mathbf{D}_c$, whose entries are the vectors of row totals and columns totals from the previous table, respectively. The vectors $\mathbf{r}$ and $\mathbf{c}$ can be manipulated to generate the diagonal elements of these square matrices as

$$\mathbf{D}_r = \begin{bmatrix} p_{1+} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & p_{r+} \end{bmatrix}, \tag{5.6}$$

and

$$\mathbf{D}_c = \begin{bmatrix} p_{+1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & p_{+s} \end{bmatrix}, \tag{5.7}$$

that contain the row and column totals $\mathbf{r}$ and $\mathbf{c}$ in their diagonal.

4. After obtaining the matrices $\mathbf{D}_r$ and $\mathbf{D}_c$, one can calculate $\mathbf{P}_r$ and $\mathbf{P}_c$, which represent the row and column profiles of $\mathbf{P}$

$$\mathbf{P}_r = \mathbf{D}_r^{-1}\mathbf{P} \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_r^T \end{bmatrix}, \tag{5.8}$$

where

$$\mathbf{a}_i^T = \left[\tfrac{k_{i1}}{k_{i+}}, \ldots, \tfrac{k_{is}}{k_{i+}}\right] \tag{5.9}$$

is defined based on the elements of the previous matrix $\mathbf{K}$, with $k_{i+}$ representing the sum of the elements over its rows, and analogously $k_{+j}$ over its columns, whilst

$$\mathbf{P}_c = \mathbf{D}_c^{-1}\mathbf{P}^T \begin{bmatrix} \mathbf{b}_1^T \\ \vdots \\ \mathbf{b}_s^T \end{bmatrix}, \tag{5.10}$$

where

$$\mathbf{b}_j^T = \left[\tfrac{k_{ij}}{k_{+j}}, \ldots, \tfrac{k_{rj}}{k_{+j}}\right]. \tag{5.11}$$

5. At this point, a fundamental step of the algorithm enables its output to become an interpretable graph, wherein the distances would resemble relationships between the variables in columns or rows. Here, the Chi-square distribution is used as a measure of distance to compute the degree of relationship between the rows and columns table dimensions. Please note that the steps below have been summarized, restricting only to the fundamental manipulations. The Chi-square distribution is calculated as follows:

   - Row profile distances: the squared $\chi^2$ distance between the entries of the two row profies $a_i$ and $a_{i'}$ is given by

     $$d^2(\mathbf{a}_i, \mathbf{a}_{i'}) = (\mathbf{a}_i - \mathbf{a}_{i'})^T \mathbf{D}_c^{-1}(\mathbf{a}_i - \mathbf{a}_{i'}) = \sum_{j=1}^{s} \frac{k}{k_{+j}}(\frac{k_{ij}}{k_{i+}} - \frac{k_{i'j}}{k_{i'+}}). \tag{5.12}$$

     However, the distance accounting a dimension category regarding its centroid is sometimes of more interest to be represented in the plot. Considering row profiles, their centroids are represented by $\mathbf{c}$. The value of the distance from $\mathbf{a}_i$ to their centroid can be obtained by

     $$d^2(\mathbf{a}_i, \mathbf{c}) = (\mathbf{a}_i - \mathbf{c})^T \mathbf{D}_c^{-1}(\mathbf{a}_i - \mathbf{c}) = \frac{1}{k_{i+}} \sum_{j=1}^{s} \frac{k}{k_{i+}k_{+j}}(k_{ij} - \frac{k_{i+}k_{+j}}{k}), \tag{5.13}$$

     which, when summed over all the row profiles, results in

     $$k\sum_{i=1}^{r} p_{i+} + d^2(a_i, c) = \sum_{i=1}^{r}\sum_{j=1}^{s} \frac{(k_{ij} - \frac{k_{i+}k_{+j}}{k})^2}{\frac{k_{i+}k_{+j}}{k}}. \tag{5.14}$$

     Taking Eq. 5.14, and matching $O_{ij}$ with $k_{ij}$ and $E_{ij}$ with $\frac{k_{i+}k_{+j}}{k}$, we find out that it can be approximated by the Pearson's chi-squared statistic as follows

     $$X^2 = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}}. \tag{5.15}$$

     In Eq. 5.15, $O_{ij}$ represents the observed cell frequency while $E_{ij}$ is the expected cell frequency. The calculation of these terms will be crucial for the next steps.

- Column profile distances: the steps involved in the case of columns are the same adopted with the rows, changing only the axis regarding the algebraic manipulations. Therefore, the column profiles distances can be calculated as

$$d^2(\mathbf{b}_j, \mathbf{b}_{j'}) = (\mathbf{b}_j - \mathbf{b}_{j'})^T \mathbf{D}_r^{-1} (\mathbf{b}_j - \mathbf{b}_{j'}) = \sum_{i=1}^{r} \frac{k}{k_{i+}} (\frac{k_{ij}}{k_{+j}} - \frac{k_{ij'}}{k_{+j'}}). \quad (5.16)$$

  Following the same rationale for the row profiles, the distance from $\mathbf{b}_j$ to its centroid is

$$d^2(\mathbf{b}_j, \mathbf{r}) = (\mathbf{b}_j - \mathbf{r})^T \mathbf{D}_r^{-1} (\mathbf{b}_j - \mathbf{r}) = \frac{1}{k_{+j}} \sum_{i=1}^{r} \frac{k}{k_{i+}k_{+j}} (k_{ij} - \frac{k_{i+}k_{+j}}{k}), \quad (5.17)$$

  that can be analogously summed over all the column profiles, reaching an equation similar to Eq. 5.14. Naturally, this equation related to the Pearson's Chi-squared statistic described by Eq. 5.15.

6. Now we must derive the total inertia equation and proceed with its decomposition. In our experiments that implement CA, we consider the one-hot-encoding modality of our categorical variables in the matrix dimensions, analogously to the examples provided in the book [69]. This allows us to proceed with the same standard assumption of contingency table analysis followed there, according to which the row and column totals are assumed to be fixed, and the cell frequencies in $\mathbf{K}$ are allowed to vary given these constraints. The relative frequency matrix that incorporates such restrictions is given by

$$\tilde{\mathbf{P}} = \mathbf{P} - \mathbf{r}\mathbf{c}^T. \quad (5.18)$$

   Using $\tilde{\mathbf{P}}$, we can define the matrix $\tilde{\mathbf{K}} = k\tilde{\mathbf{P}}$, also known as the matrix of residuals. This name is given because its $ij$th entry shows the difference between the observed cell frequency $O_{ij} = k_{ij}$ and the expected cell frequency $E_{ij} = \frac{k_{i+}k_{+j}}{k}$, resulting in $\tilde{k}_{ij} = O_{ij} - E_{ij}$. The last step to reach the inertia calculation is to calculate the $(s \times s)$-matrix $\mathbf{R}_0$

$$\mathbf{R}_0 = \mathbf{D}_c^{-\frac{1}{2}} \tilde{\mathbf{P}}^T \mathbf{D}_r^{-1} \tilde{\mathbf{P}} \mathbf{D}_c^{-\frac{1}{2}}, \quad (5.19)$$

   where $\mathbf{D}_r^{-1} = diag\{\mathbf{r}^{-1}\}$ and $\mathbf{D}_c^{-\frac{1}{2}} = diag\{\mathbf{c}^{-\frac{1}{2}}\}$. The entry in the $j$th row and

$j'$th column of $\mathbf{R}_0$ is given by

$$R_{0_{jj'}} = (k_{+j}k_{+j'})^{-\frac{1}{2}} \sum_{i=1}^{r} \frac{1}{k_{i+}} (k_{ij} - \frac{k_{i+}k_{+j}}{k})(k_{ij'} - \frac{k_{i+}k_{+j'}}{k}), \tag{5.20}$$

and the $j$th diagonal entry of $\mathbf{R}_0$ is obtained by setting $j = j'$

$$R_{0_{j'j'}} = \frac{1}{k_{+j}} \sum_{i=1}^{r} \frac{1}{k_{i+}} (k_{ij} - \frac{k_{i+}k_{+j}}{k})^2. \tag{5.21}$$

The trace of $\mathbf{R}_0$, which corresponds to the sum of its eigenvalues, is given by

$$\sum_{j=1}^{s} \lambda_j^2 = tr\{\mathbf{R}_0\} = \sum_{i=1}^{r} \sum_{j=1}^{s} \frac{1}{k_{i+}k_{+j}} (k_{ij} - \frac{k_{i+}k_{+j}}{k})^2 = \frac{X^2}{k}, \tag{5.22}$$

where $X^2$ is given by Eq. 5.15. Finally, the accumulated contribution of the first $t$ principal components (or inertias) can by calculated using

$$C_t = \frac{\lambda_1^2 + \ldots + \lambda_t^2}{\sum_{j=1}^{s} \lambda_j^2}. \tag{5.23}$$

7. Compute the scaled matrix $\mathbf{M}$ where $\mathbf{M} = \mathbf{D}_r^{-\frac{1}{2}} \tilde{\mathbf{P}} \mathbf{D}_c^{-\frac{1}{2}}$, where $\tilde{\mathbf{P}}$ is the same as calculated in Eq. 5.18.

8. Compute the SVD of the matrix $\mathbf{M} = \mathbf{U}\mathbf{D}_\lambda \mathbf{V}^T$, where $\mathbf{U}$ is an $(r \times s)$ unitary matrix, i.e., $\mathbf{U}^T\mathbf{U} = \mathbf{I}_s$; $\mathbf{D}_\lambda = diag\{\lambda_1, \ldots, \lambda_s\}$ is a $(s \times s)$ diagonal matrix defined by the singular values, and $\mathbf{V}$ is an $(s \times s)$ unitary matrix, therefore $\mathbf{V}^T\mathbf{V} = \mathbf{I}_s$

9. Calculate the matrices that represent the principal axes of the row and column profiles, namely $\mathbf{A} = \mathbf{D}_r^{-\frac{1}{2}}\mathbf{U}$ and $\mathbf{B} = \mathbf{D}_c^{-\frac{1}{2}}\mathbf{V}$

10. After obtaining $\mathbf{A}$ and $\mathbf{B}$, a series of manipulations are performed to reach the coordinate matrices, which will not be shown here but are available in the reference book [69]. The standard principal coordinates matrix can be obtained by calculating $\mathbf{G}_S = \mathbf{U}\mathbf{D}_r^{-\frac{1}{2}}$ and $\mathbf{H}_S = \mathbf{D}_c^{-\frac{1}{2}}\mathbf{V}$. The columns of $\mathbf{G}_S^T$ and $\mathbf{H}_S^T$ represent the standard coordinates of the row and column profiles, respectively.

11. To be able to exploit the conventional interpretation criteria over the generated plot, the coordinates must be scaled, otherwise interpretation guidelines often explored turn invalid, hindering the extraction of correct insights. The scaling techniques and their appropriate use cases deserve a deeper explication

that is available in [69]. To obtain the scaled coordinates, we compute the matrices containing the principal coordinates of the row and column profiles as $\mathbf{G}_P = \mathbf{D}_\lambda \mathbf{U} \mathbf{D}_r^{-\frac{1}{2}}$ and $\mathbf{H}_P = \mathbf{D}_\lambda \mathbf{D}_c^{-\frac{1}{2}} \mathbf{V}$, where $\mathbf{D}_\lambda$ is as defined in the step 8. These matrices contain the data that is necessary to extract associations and properties from both dimensions, and they mark the end of the algorithm.

By following all the previous steps, the $(x, y)$ coordinates will be available for all categories in both dimensions through the $\mathbf{G}_P$ and $\mathbf{H}_P$ matrices, respectively. Such rows and columns when drawn as a scatter plot in a two-dimensional cartesian plane can be used to indicate how each category in their dimension is positioned and how to correlate them. Depending on whether the relation between two points is intra or interdimensional, there are different methodologies to interpret, which will be briefly discussed in the next section.

### Interpreting CA outcomes

Considering our analysis requirements, we will restrict the following interpretation guideline to only two-way contingency tables (associations between 2 variables). The final visualization has 3 graphical features that should be considered for interpretation.

1. The distance of the categorical variable (row or column) with respect to the origin.
2. The proximity of data points that belong to the same dimension (comparison of rows with themselves or columns with themselves). This is the recommended measure of intradimensional similarity .
3. The angle that connects a row and a column variable to the origin. This is the recommended measure of interdimensional similarity.

Regarding the first item, whenever a category is far from the origin, it can be inferred that it is more dissimilar than other categories that are close to the origin. The category which is furthest from the origin is less correlated with the others and usually grants characteristics that distinguish it from the other categories within the same dimension. The opposite is also valid, if a given category is near the origin, this means that it probably shares characteristics with other nearby categories.

The second item refers to categories of the same dimension that usually share similar characteristics, thus showing small distances between them. Conversely, if a pair of categories are far from each other in the cartesian plane, they are somewhat uncorrelated and they probably vary in different degrees with respect to the categories of the other dimension.

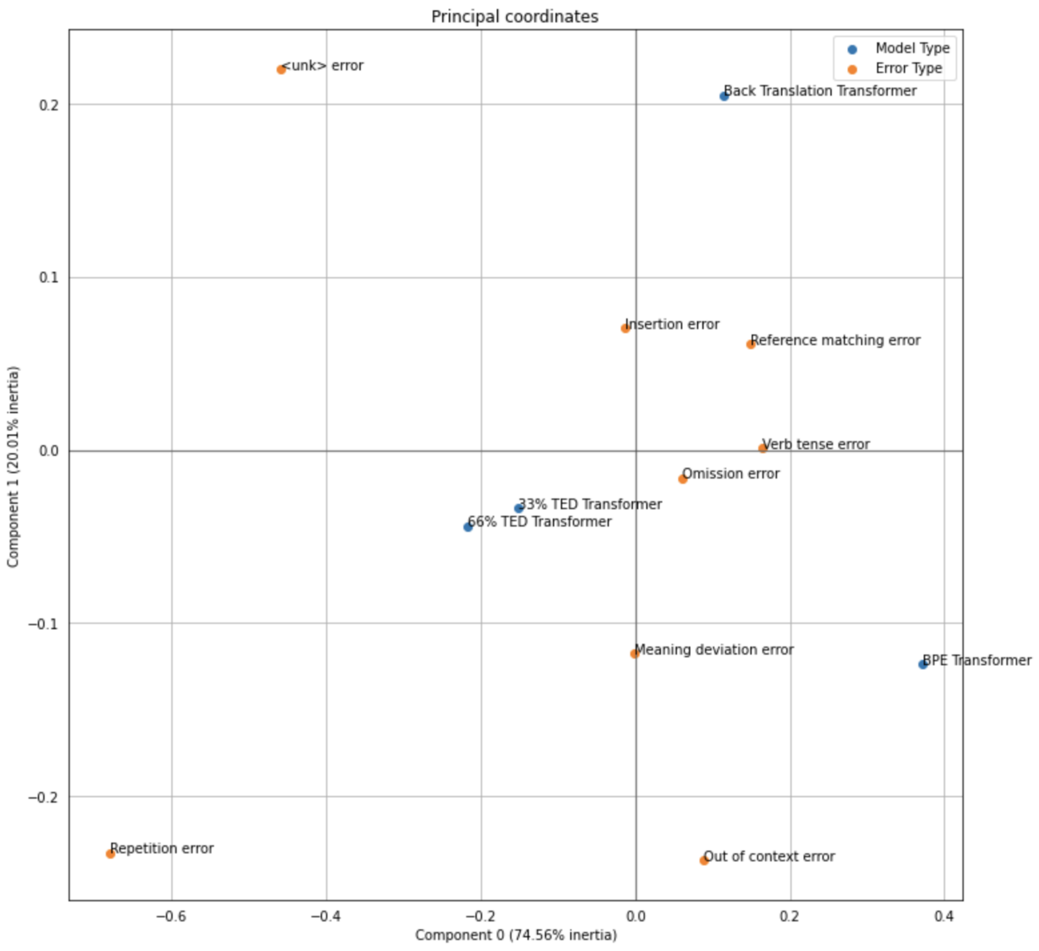


Figure 5.3: Correspondence analysis visualization comparing models and error classes

Lastly, the third interpretation point is related to comparisons of categories belonging to distinct dimensions. The practical procedure is as follows: draw a line between the origin and a row category that you wish to compare, and repeat the same process for the desired column category. If the angle formed between those straight lines is very acute (close to 0), such categories are probably highly associated. If this angle is near 90 degrees, then these categorical variables do not possess any type of association at all; however, if the angle is close to 180 degrees, such categories are probably negatively associated.

### Analysis of models and classes of errors

We must recall that the dataset explored in this experiment is the same as used in Section 5.3.4. This dataset was reorganized in a 2-way contingency table, such that the rows correspond to the model types and the columns represent each one of the 8 error classes. The algorithm had this table as input.

The algorithm has successfully fitted the dataset with an explained variance of 94.6%, wherein 74.6% is related to the first component and 20.01% to the second component. This score indicates a very good representation of data. Figure 5.3 shows the resulting plot of this experiment.

Some errors have stood out from the rest. Particularly, *Repetition*, *<unk>* and *Out of Context* are all far from the origin, whilst the others are more concentrated into a smaller area. *Reference Matching* is located very close to *Insertion*, whilst *Verb Tense* is also close to *Omission*, indicating that these error patterns share similarities over the models. All these errors are located in an area with approximately the size of a box in the current graph scale, which means that despite the pairwise similarity between them seems to be higher, there is also some level of similarity between them.

Regarding the models, Back Translation stands out from the others, whilst 33% and 66% are close, indicating that their error patterns share similarities, as expected. BPE seems to be isolated, so it has a weak relationship with the rest, but not as far from the origin as Back Translation. BPE error patterns are probably low-correlated with those from the Back Translation model as they are the furthest models in the cartesian plane.

Below we report the main findings of this experiment, following a model-centered perspective:

1. BPE shows a positive correlation with the *Omission* error, being the model with the strongest relationship with this error. There is also some positive correlation with *Verb Tense*, *Out of Context* and *Meaning Deviation*. It is negatively correlated with the *Insertion* error and also with the *<unk>* error, as its placed in about 180 degrees from them (which makes sense even more for *<unk>* since BPE does not produce this error at all). At the same time, it seems not to correlate with the *Repetition* and *Reference Matching* errors, exhibiting an angle near 90 degrees with them relatively to the origin

2. Both reduced models 33% and 66% exhibit a similar pattern of relationship with the errors. They have shown to be highly correlated with the *Repetition* error, positively correlated with *Meaning Deviation* and *<unk>* errors, and negatively correlated with the *Reference Matching* and *Verb Tense* errors. Besides, these models seem to have almost no correlation with the *Omission*, *Insertion* and *Out of Context* errors

3. Finally, the Back Translation model has a high positive correlation with the *Reference Matching* and *Insertion* errors, and also some positive correlation with *<unk>*, whilst having almost no correlation with *Verb Tense* and *Omission* errors. It also has a strong negative correlation with the *Repetition* and *Meaning Deviation* errors, and in a smaller degree with *Out of Context*

Some interesting hypothesis can be outlined based on this graphical inspection.

Augmenting with data from a different domain induces the model to create translations that don't match the reference, potentially indicating that it finds synonyms for words of the same sentence. BPE seems to frequently omit words although it is also efficient against the *Insertion* of unnecessary words. Regarding the models trained on the reduced TED, they seem to repeat the same words when they do not possess vocabulary knowledge to finish a sentence with certainty. Also, the reduced vocabulary seems to prevent the occurrence of synonyms, probably because of the limited number of words for the algorithm choice.

### Analysis of sentence complexity and classes of errors

In this second round of experiments, the table of error patterns per model has been stratified by sentence complexity, generating a 2-way contingency table with CEFR complexity levels in the rows and the columns representing the classes of errors. The explained variance indicates again a successful representation of data: with two components it reaches 96.29%, with a contribution of 64.28% from the first component and 32.01% from the second component. The resulting visualization can be seen in Figure 5.4.

Considering the categorical associations along with both dimensions, all the positive, negative and absence of correlations derived from graphical inspection that were observed in Figure 5.4 are highlighted in the following:

1. A1 class exhibits a positive correlation with the *Insertion*, <*unk*> and *Repetition* errors; a negative correlation with *Verb Tense* and *Omission* errors, and a potential absence of correlation with the *Reference Matching*, *Out of Context* and *Meaning Deviation* errors

2. The figure indicates that A2 can be positively correlated with *Reference Matching*, *Out of Context* and *Omission* errors, probably negatively correlated with <*unk*>, *Meaning Deviation* and *Verb Tense* errors and lacks correlation with the *Insertion* and *Repetition* errors

3. B1 has a positive correlation with *Omission*, *Verb Tense* and *Out of Context* errors. A negative correlation is observed for *Insertion*, *Repetition* and <*unk*> errors, and also apparently there is no correlation with the *Reference Matching* and *Meaning Deviation* errors

4. Lastly, B2 seems to have a positive correlation with *Verb Tense*, *Meaning Deviation* and <*unk*> errors; a negative correlation with *Out of Context* and *Reference Matching* errors, and no correlation with the *Repetition*, *Omission* and *Insertion* errors

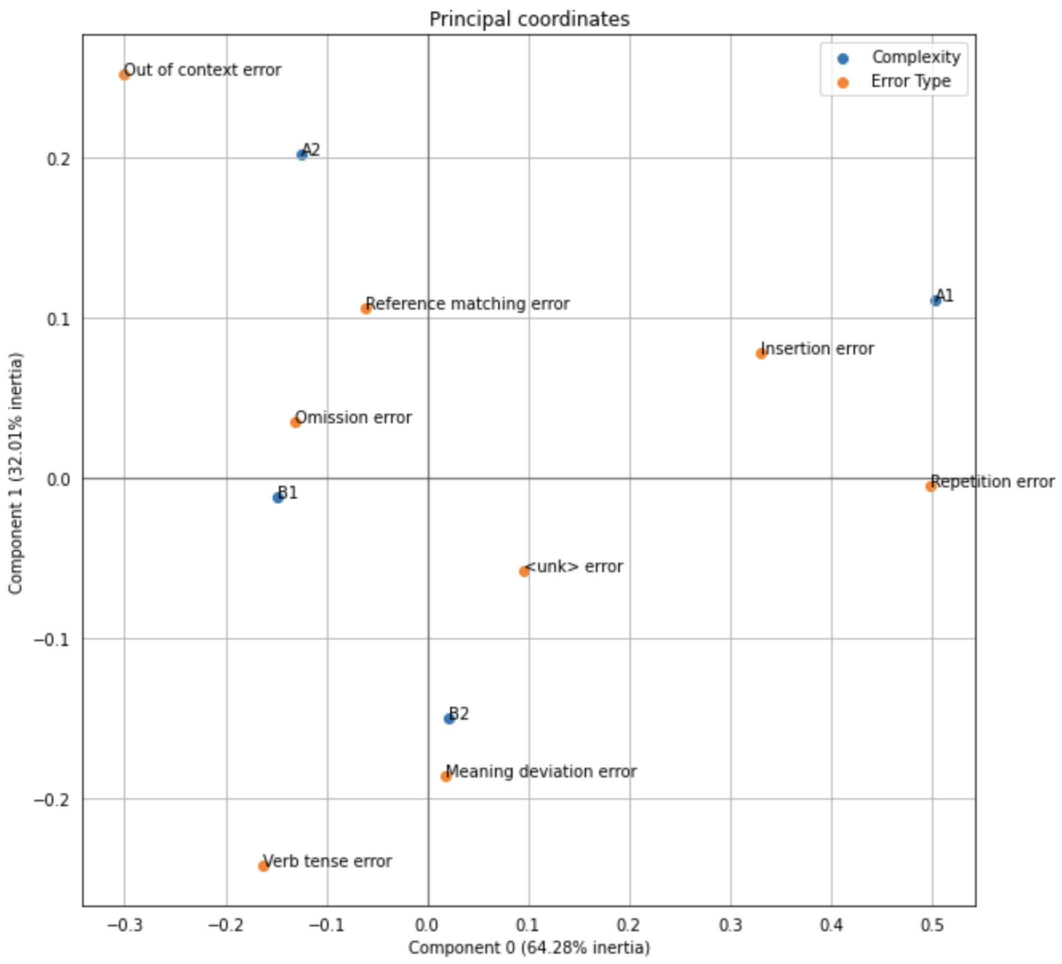


Figure 5.4: Correspondence analysis visualization comparing complexities and error classes

The most isolated, thus distinguishing categories reported by this analysis are *Repetition*, *Out of Context* and *Verb Tense* errors as well as A1 complexity. The complexity labels are well distributed along the graph area, leading us to conclude that the process of classifying sentences in complexity levels is somewhat consistent. The pairs B1 and B2, A2 and B1 are closer to each other, which makes sense as they represent consecutive complexities. The spread observed between the locations of the classes of errors has also improved when compared to the last experiment, which means that the association between sentence complexity and this dimension is clearer than the one considered in the last section. The proximity of the *Insertion* with the *Repetition* errors indicates some similarity between these errors, which was also observed in the previous experiment. *Omission* and *Reference Matching* errors are also close to each other, but this relation has only come out in this experiment.

The relations extracted through graphical interpretation have shown some interesting trends. The high explained variance and well distributed data points for both dimensions indicates that the categories considered in this analysis (sentence complexity and error classes) segmented very well the patterns of errors observed with these models. Qualitatively "simpler" errors such as *Insertion* and *Repetition* are usually correlated with sentences of lower complexity such as A1 and uncorrelated or negatively correlated with those of higher complexities, such as B1. Conversely, <*unk*> and *Meaning Deviation* errors, which are often more related to complex translations, are usually positively associated with sentences of higher complexities, such as from the B2 level, and negatively associated with those from lower complexities, such as A1. The other errors, namely *Omission*, *Reference Matching*, *Out of Context* and *Verb Tense* when compared to the complexity points indicate that they are situated in some sort of "medium level" complexity, as they create acute angles with A2 and B1. Ideally, to confirm the assertions made here with high confidence, similar patterns should be spotted in an equivalent experiment with an increased sample size, but we believe the current scenario brings solid evidences.

### 5.3.6 Consolidating the qualitative study

Each experiment performed in the qualitative domain allowed us to derive a number of insights, but being able to visualize the whole picture based on the outcomes of the isolated experiments is a hard task. In this section we have condensed the most relevant findings and will try to establish a connection between them, aiming to provide a clear and concise view about the major outcomes of the qualitative analysis.

As our goal here is to combine the results previously reported, it is important to recall why and how the techniques chosen to evaluate the translations qualitatively complement each other. The Fisher Exact test presented in Section 5.3.3 indicated relevant differences in the prevalence of error classes when comparing models. The HC algorithm in Section 5.3.4 helped to trace an unique profile of error statistics for the prevailing models in a cluster, whilst the CA in Section 5.3.5 experiments provided an easier interpretation along with richer and clearer associations between the dimensions of interest. In this case, CA help us to identify the presence of positive or negative correlation between 2 dimensions, or even the absence of it.

The same approach performed for the experiments will be followed in this section, analysing separately the qualitative influences of the types of models and sentence complexity with respect to the classes of errors.

#### Models and classes of errors

The main challenge in providing a clear view of how the findings connect with each other is that each experiment is performed with a different methodology. We believe that a table is the best visualization given this complexity, but before the results are presented, this section starts with a brief explanation of how to interpret the outcomes that each algorithm provide to a specific model. Independently of the algorithm, there are 3 potential relationships that the model can have with an error class:

1. The model has a high prevalence for the class of error $X$, so an increase in the percentage of sentences translated by that model in a text will increase the probability of that error to happen, which can be seen as a positive correlation
2. The model has a low prevalence for the class of error $X$, so an increase in the percentage of sentences translated by that model in a text will decrease the probability of that error to happen, which can be seen as a negative correlation
3. The relationship of the model with the class of error $X$ is inconclusive, being analogous to having no correlation

Now, to understand the results that will be later discussed in Table 5.18, the correct way to interpret the results for each model-experiment pair will be explained here. The interpretation of the results of a given model considering a specific algorithm are complemented by the information between brackets. This is how the analogy is made, considering each algorithm:

1. Fisher Exact test: Consider the relationship 1, which indicates that the model frequently produces a specific class of error. In the results table, the model being analysed will be indicated in the column, and the cell belonging to the algorithm's row will enumerate the error class(es) to which a statistically significant difference was found, considering the error prevalence of the analysed model and the model(s) reported in brackets. As the relationship in discussion is 1, then according to the test, the column model has a higher prevalence than the model in brackets. As an example, consider the *Repetition* error in Table 5.18 for both 33% and 66% TED models: the brackets show that Fisher Exact test indicated with 95% confidence that these models showed a higher prevalence than both BPE and BT models. The distinction between higher or lower prevalence depends on whether the odds ratio number is either positive or negative. The same analogy can be applied to interpret the relationship 2, associated to a negative odds ratio, but for the third relationship, we have positioned in this table all the error classes we could not refuse the hypothesis of similar proportions (null hypothesis) with 95% of confidence.
2. Hierarchical Clustering: the results extracted from this algorithm depend on the relationship being discussed, and for them we have two main sources of data: Table 5.14 and the enumerated descriptions by cluster from Section 5.3.4. For the relationship 1, we took the highest prevalence considering one error

class and cluster pair from Table 5.14, in some cases adding more models to the list depending on the gap between their prevalence and the one from the other models. Some scenarios can be illustrated to help translate that definition into practice: if 3 models have 100% and one model has 20%, only the latter will be highlighted with negative tendency, whilst in another experiment if 2 models have 98% and 95% of one error class and the others have <20%, then we consider the first and second with a positive tendency. However, if all four models have 98%, 30%, 20% and 8%, then the first model (98%) has a positive tendency and the last model (8%) has a negative tendency in that cluster. The goal is to measure how much that prevalence deviates from the mean and is closer to be classified as an outlier in that cluster, which is an indicator if that model possesses a different relationship with that specific error class when compared with others. Note that it is hard to be inconclusive in the case of HC, but cluster 3 has an example in Table 5.14, where all the four models have produced the *Out of Context* error in 100% of the sentences. We consider this to be inconclusive, since it shows no difference between models, but this result was registered only for the 33% TED and BPE models, since the others have a very low sample size of 2 or less sentences in that cluster.

3. Correspondence Analysis: For this algorithm, the methodology to analyse is easier since it translates directly to its visual interpretation given in Section 5.3.5. There are no brackets for the classes of errors enumerated in this model's row. Illustratively, for the relationship 1, the model has an acute angle with the origin and the error class; for the relationship 2, the closer it gets to 180 degrees, the stronger is the relationship; and for the relationship 3 that angle is around 90 degrees. Consider for instance the first CA row for BPE, which shows that this model tends to produce several *Verb Tense* errors, whilst the second CA row for the same model indicates that it produces few *Insertion* and none <*unk*> errors.

There are some relevant comments to make before exploring the results. The findings involving HC for the cluster 3 were discarded since the cluster only represents 12 sentences, with 1 belonging to the BT model and 2 to 66% TED, thus the sample size is not enough to be conclusive. Also, the categories of correlation with no findings were filled with "N/A". The row containing results with "almost zero correlation" had a different motivation for each experiment: in Fisher it basically represents the classes to which we could not reject the null hypothesis (neither positive nor negative correlation), whilst for HC if the prevalence of an error class does not vary between models, it falls under this category.

When analysing the results, it is important to keep in mind that there is some

Table 5.18: Main model-error conclusions extracted from the qualitative experiments

| Relationship | Experiment | Models | | | |
|---|---|---|---|---|---|
| | | Back Translation | 33% TED | 66% TED | BPE Transformer |
| The model has a **positive** tendency related to these classes of errors | Fisher Exact Test | N/A | *Meaning Deviation* (vs BT), *Repetition* (vs BT, BPE), <*unk*> (vs BT, BPE) | *Meaning Deviation* (vs BT), *Repetition* (vs BT, BPE), <*unk*> (vs BT, BPE) | N/A |
| | HC | *Omission* (cl1), *Reference Matching* (cl1) | <*unk*> (cl1), *Verb Tense* (cl3), *Reference Matching* (cl0), *Omission* (cl3), *Out of Context* (cl3) | *Reference Matching* (cl2), *Out of Context* (cl3) | *Reference Matching* (cl1, cl2), *Verb Tense* (cl1), *Insertion* (cl1), *Meaning Deviation* (cl1) |
| | CA | *Reference Matching*, *Insertion*, <*unk*> | *Repetition*, *Meaning Deviation*, <*unk*> | *Repetition*, *Meaning Deviation*, <*unk*> | *Omission*, *Meaning Deviation*, *Out of Context*, *Verb Tense* |
| The model has a **negative** tendency related to these classes of errors | Fisher Exact Test | *Meaning Deviation* (vs 33%), *Repetition* (vs 33%, 66%), <*unk*> (vs 33%, 66%) | N/A | N/A | *Repetition* (vs 33%, 66%), <*unk*> (vs all models) |
| | HC | *Verb Tense* (cl0), *Meaning Deviation* (cl1), *Insertion* (cl1) | N/A | *Verb Tense* (cl1), *Out of Context* (cl1) | <*unk*> (all clusters), *Omission* (cl1), *Repetition* (cl0) |
| | CA | *Repetition*, *Meaning Deviation*, *Out of Context* | *Reference Matching*, *Verb Tense* | *Reference Matching*, *Verb Tense* | *Insertion*, <*unk*> |
| The model doesn't exert an influence on these error classes of errors | Fisher Exact Test | *Omission*, *Out of Context*, *Reference Matching*, *Insertion*, *Verb Tense* | *Omission*, *Out of Context*, *Reference Matching*, *Insertion*, *Verb Tense* | *Omission*, *Out of Context*, *Reference Matching*, *Insertion*, *Verb Tense* | *Omission*, *Out of Context*, *Reference Matching*, *Insertion*, *Verb Tense* |
| | HC | N/A | *Out of Context* (cl3) | N/A | *Out of Context* (cl3) |
| | CA | *Verb Tense*, *Omission* | *Omission*, *Out of Context*, *Insertion* | *Omission*, *Out of Context*, *Insertion* | *Repetition*, *Reference Matching* |

Table 5.19: Main model-error conclusions extracted from CA

| Relationship | Models | | | |
|---|---|---|---|---|
| | Back Translation | 33% TED | 66% TED | BPE Transformer |
| The model has a **positive** tendency related to these classes of errors | *Reference Matching*, *Insertion*, <*unk*> | *Repetition*, *Meaning Deviation*, <*unk*> | *Repetition*, *Meaning Deviation*, <*unk*> | *Omission*, *Meaning Deviation*, *Out of Context*, *Verb Tense* |
| The model has a **negative** tendency related to these classes of errors | *Repetition*, *Meaning Deviation*, *Out of Context* | *Reference Matching*, *Verb Tense* | *Reference Matching*, *Verb Tense* | *Insertion*, <*unk*> |
| The model doesn't exert an influence on these error classes of errors | *Verb Tense*, *Omission* | *Omission*, *Out of Context*, *Insertion* | *Omission*, *Out of Context*, *Insertion* | *Repetition*, *Reference Matching* |

quantitative evidence that the results from the CA algorithm are more reliable than the ones from HC. We must stress that the explained variance achieved by CA is about 95%, which may be considered as near to optimal, whilst the Silhouette coefficient for HC is 0.2320, notably far from the desired target of 1. This leads us to follow the logic that the findings of CA are statistically more appealing than the ones from HC. The results for each model are summarized with more depth below:

- **Back Translation** This transformer is probably more subjected to the *Reference Matching* error based on results from HC and CA, as shown in the positive correlation row of Table 5.18, with potentially the same phenomenon happening with the *Insertion* error, denoted by CA. The hypothesis of the *Omission* and <*unk*> errors being positively correlated with this model are discarded by two factors: (1) the suggestion from CA that *Omission* has almost zero correlation, and (2) the Fisher Test accusing that <*unk*> is less frequent for the BT model versus the reduced TED models, which also diverges from CA that shows an angle bigger than 60 degrees (hence, small positive correlation). The BT model also spawns less often (i.e. has a negative correlation) the *Out of Context* error according to CA, which is also true for *Repetition* and *Meaning Deviation*, backed by CA and Fisher (at least when compared to the low-resource models). Lastly, the *Verb Tense* error seems to be supported by both Fisher and CA that it isn't correlated with BT at all, and a similar trend is followed by *Omission*.

- **Reduced TED models (33% & 66%)** These models share similar results in all experiments. The Fisher test and CA indicate that the *Repetition* and *Meaning Deviation* errors are common errors produced by these models. They also produce the <*unk*> token a lot more when compared to BT and BPE based on Fisher and CA. Both CA and Fisher test showed that the prevalence of the *Omission*, *Insertion* and *Out of Context* errors seem not to be affected by an increase of sentences evaluated from the 33% and 66% TED models. Regarding negative correlation, it seems that the *Reference Matching* and *Verb Tense* errors are pointed out to occur less with the models trained on reduced TED models by HC and CA.

- **BPE Transformer** In this transformer, the CA row for positive correlation in Table 5.18 shows that the prevalence of errors belonging to the *Meaning Deviation*, *Verb Tense*, *Out of Context* and *Omission* classes varies in a positive trend with the number of sentences generated by such model. The <*unk*> token exhibited a negative correlation with the BPE model according to all experiments, which makes sense as in practice the model does not produce such tokens, it tries to map the translations to a customized word. In the meanwhile, CA indicates that the *Insertion* error should not also commonly appear when using BPE. Fisher tells the same relatively to the *Repetition* error. Regarding the absence of correlation, CA points that the *Reference Matching* error is uncorrelated with the type of the model.

These were the straightforward findings that correlate models and errors. However there are some deeper reflections and insights that can be extracted and are not so evident in such statements. For instance, one hypothesis is that the use of different domain augmented data induces the model to create translations that do not match the reference. Besides, it is clear that in the positive correlation row of Table 5.18 the Back Translation model is the one most associated with this type of error. This cannot be interpreted as a rule, further studies are required to confirm or not this claim, but the raise of this hypothesis wouldn't be even possible without applying the aforementioned algorithms. Another possible claim is that BPE seems to frequently omit words although it is also efficient against the *Insertion* of unnecessary words. This claim is supported by the CA experiment in the positive and negative correlation rows, respectively.

Speaking more broadly about the models under a more severe low-resource constraint, i.e., the ones trained on the reduced TED datasets: they seem to be more prone to repeating the same words as they do not possess enough vocabulary knowledge to accurately complete a sentence, according to CA and Fisher. Moreover, the reduced vocabulary seems to prevent the occurrence of synonyms, as this issue is more likely to be caught by the *Reference Matching* error, which might have happenend because of the limited options of words. In all experiments, both the models trained on 33% and 66% of the dataset shared similar error distributions, are situated nearby in the graphical visualizations (CA), or are segmented in clusters with a similar share of sentences. All these results provide a strong evidence that these models share some similarities in their behaviour. Despite being subjected to different levels of low-resource constraints, their prevalences in the classes of errors are close enough for us to conclude that the constraints had a similar impact on their performance.

### Sentence complexity and error classes

A similar analysis was conducted to integrate the previous findings but now focusing on the relations between the sentence complexity and the classes of errors. As complexity was included more as a complement to understand how the models behave in terms of producing the classes of errors, it does not make sense to analyse each complexity level following the same methodology previously performed for each model that participated in the study, as in the previous section. The goal here is to understand how sentences belonging to different complexity levels exert an influence in the likelihood of a specific class of error to appear more often or not. Moreover, it is not convenient to correlate the models directly with the sentences complexities in this case since there is no clear value expected from such analysis.

One hypothesis to be evaluated for the sentence complexity dimension is to understand if the criteria established for the CEFR segmentation were correctly followed, enough to see an impact in the distribution of errors given a change in the complexity level. A straightforward approach to validate this is checking if an increase in the sentence complexity level leads to an increase, decrease or no difference in the number of errors made. This analysis was performed over the results of each qualitative experiment and summarized in Table 5.20, but before discussing and extracting its findings, we will start by explaining how it was generated.

Consider the same relationships described at the beginning of Section 5.3.6. Each experiment followed a distinct approach in the process of consolidating its findings, which will be explained in the items below:

1. Fisher Exact test: The results taken from this test always compare levels of sentence complexity distancing two or more CEFR levels, namely A1|B1, A2|B2 or A1|B2. We believe that skipping only one complexity level might not

result in a significant change in the error class distribution, so we will discard adjacent levels like A1|A2. With this in mind, the tests will also analyse the odds ratio, which indicates if a model is producing more or less errors than other, considering one specific class. Therefore, the odds ratio will define what is the relationship between one class of error with some change in the sentence complexity (1, 2 or 3, the same as explained in Section 5.3.6).

2. Hierarchical Clustering: In this experiment we evaluate if an increase in the sentence complexity directly relates to an increase in the prevalence of one class of error. If the prevalence of 2 adjacent complexity levels are very close, but the trend is clear given the complexity levels, we will not consider such a small gap as enough evidence to disqualify the hypothesis that the classes of errors show a complexity trend. In brackets, the cluster where the relationship can be observed is specified, and if applicable, an exception may be mentioned as not strictly following the expected monotonic trend. The three possible trends are the same as the aforementioned relationships.

3. Correspondence Analysis: The results reported in Section 5.3.5 can be easily translated into the relationships provided in Table 5.20, where sentence complexities associated with the antecedent class of error are reported in brackets.

Table 5.20: Main complexity-error conclusions extracted from the qualitative experiments (see text).

| Relationship | Experiment | Errors that match the relationship and the circumstances of occurrence |
|---|---|---|
| Errors directly related to the sentence complexity | Fisher Exact Test | *Reference Matching*, *Omission* (except for A2\|B2), *Out of Context* (except for A2\|B2), *Verb Tense*, *Meaning Deviation* |
| | HC | *Reference Matching* (cl0 with B1 diverging, cl1 with B1 diverging, cl2 with A2 diverging), *Verb Tense* (cl1, cl3), *Meaning Deviation* (cl1 with B1 diverging), *Omission* (cl1, cl3 with B1 diverging) |
| | CA | *Insertion* (A1), *Repetition* (A1), *<unk>* (A1, B2), *Meaning Deviation* (B2), *Verb Tense* (B1, B2), *Reference Matching* (A2), *Out of Context* (A2, B1), *Omission* (A2, B1) |
| Errors inversely related to the sentence complexity | Fisher Exact Test | *Insertion* (except for A1\|B2, A2\|B2), *<unk>* (except for A1\|B2, A2\|B2) |
| | HC | N/A |
| | CA | *Omission* (A1), *Reference Matching* (A2, B2), *Out of Context* (A2, B2), *Insertion* (B1), *Repetition* (B1), *<unk>* (A2, B1), *Meaning Deviation* (A2), *Verb Tense* (A1) |
| Almost no correlation with the sentence complexity | Fisher Exact Test | *Repetition* |
| | HC | *Out of Context* (cl3), *Verb Tense* (cl0) |
| | CA | *Insertion* (A2, B2), *Repetition* (A2, B2), *Meaning Deviation* (A1, B1), *Out of Context* (A1), *Meaning Deviation* (A1), *Reference Matching* (B1), *Omission* (B2) |

Most of the classes of errors (5 out of 8) stood out from the rest for being directly related with complexity: *Reference Matching*, *Omission*, *Out of Context*, *Verb Tense* and *Meaning Deviation*. Some of them in unanimity, like *Reference Matching*, *Omission*, *Meaning Deviation* and *Verb Tense*. The *Out of Context* error is supported by strong evidence provided by Fisher and CA, as indicated in the first relationship in the errors listed in the rows that matches these experiments, but not from HC. In addition, there are some errors that appear to have an inverse relationship with the sentence complexity: *Insertion*, which is supported by Fisher and by CA in the complexity B1, and <*unk*>, which is supported by Fisher and by CA in the complexities A2 and B1. Note that the relationships aren't always pointed out in unanimity, but we understand that the most relevant combination of experiments represent a strong factor where one pattern is the winner. By elimination, all the experiments but HC have indicated that the *Repetition* error seem to not be affected by complexity. This means that presenting more complex or less complex sentences to a model is unlikely to change its probability to happen. It is the only error that is potentially more prone to be modified with the change of the model rather than the complexity of the sentence.

There is a story to tell based on the results that go beyond the straightforward interpretation of them: all experiments have indicated that the sentences seem to have been well segmented by the CEFR criteria. The spacial distribution of the complexity categories in the CA plot in Figure 5.4 endorses the previous insight, the Fisher test also does by finding relevant changes in most complexity pairs located at two levels of distance away. To some extent, but in a less evident manner, the distribution of errors per cluster also supports the argument that a good segmentation must have taken place. The HC algorithm has agglomerated sentences with similar complexity into the Clusters 3 and 4, despite not receiving this information beforehand. It is interesting to note a higher occurrence of errors for specific complexity levels. Not always these errors were positively related, but in most times (5 out of 8, 62.5% to be precise) the errors have shown a positive trend with the increase in the complexity of the sentences presented to the models, whilst for 2 out of 8 classes of errors (25%), they have shown to be negatively related; and for the remainder (12.5%, one error) no trend was observed (absence of correlation). A worrisome scenario would happen if most errors have shown no correlation with the complexity categories, but numbers show that this is far from reality.

# Chapter 6

# Conclusion

The field of Machine Translation has experienced a sharp growth of architectures and techniques, but the intricacies involving the errors of such translations are often overlooked. Translation errors can be produced by either statistical or neural models, despite the increasing popularity and adoption of the neural ones, which are also known to be very data hungry. The growth in demand for data has soared higher than what most datasets can supply, so the MT field is looking for more effective models that can reach desirable performance levels whilst training over small or medium sized datasets. Small datasets are the most common size in representativity, specially in low-resource languages. Moreover, models working under resource constrained environments are more prone to producing errors and to suffer from more complex translation quality constraints. The opportunity to better understand this phenomenon has motivated this work to investigate qualitatively the patterns of errors that certain models present in their translations. All the analyses performed were complemented with experiments that evaluated quantitative metrics such as BLEU, considering its limitations and drawbacks.

To achieve the goal of first describing the possible translation errors, we have made a collaboration with a Brazilian translator specialized in English, which accomplished the task of defining and labelling classes of errors for evaluating automatic translations. In addition to specifying error classes, we searched for another qualitative dimension that could also help in understanding the reason why a model is demonstrating some kind of error pattern, and concluded that sentence complexity is a relevant complement to understand such patterns. The Brazilian translator has also helped to classify each of the translated sentences into CEFR complexity levels [4], which was analysed together with the class of error and the type of the model in an effort to qualitatively understand better the errors that a model produces. The outcomes of this expert analysis were submitted to Statistical and Machine Learning tools, such as Fisher Exact Test, Hierarchical Clustering and Correspondence Analysis, with the goal of extracting hidden insights of model performance considering the error classes created. Before wrapping up the main findings of such experiments, it is convenient to briefly outline the rationale behind this general work organization and experiments conduction.

As an effort to better contextualize the reader in the NMT domain, we introduced the historical NLP models that were the forerunners of the modern Transformer architecture in Chapter 2, wherein the challenge for promoting this qualitative analysis was better defined and also rated in more technical terms. In the following chapter we focused on the architecture that has prevailed upon the others: the Transformer; along with its relevant components, such as the Beam Search module, Positional Encoding and the Attention mechanism. According to the definition of Low-resource given in Chapter 4, we discussed techniques for dealing with the intrinsic challenges of this context, such as Subword Embeddings, Transfer Learning (for Word Embeddings) and Back Translation, which indeed have proven to improve the quality of the translations produced by the naive Transformer. This fact motivated us investigate how such strategies may affect the error patterns observed in these translations..

Some of the hypothesis raised throughout the work regarding ways to address low-resource effects were evaluated in Chapter 5, where firstly we evaluated the impact of increasingly restricting the number of instances from the dataset used for training in the BLEU score of a traditional Transformer. This experiment indicated that a relatively small sample size (66% of the TED Talks dataset, around 157K sentences) seem to be surprisingly sufficient. A complementary experiment compared the benefits of switching to subword with incorporating external knowledge through Word Embeddings, to which BPE demonstrated to be superior to pretrained Fast-Text in all settings. The effectiveness of using Back Translation to address low-resource was also analysed in another experiment, which has shown that augmenting data from the same domain is more effective to our problem. Besides, the percentage of artificially generated translations contibutes to the effectiveness of this approach. Finally, we compare our best models against the Google Translate benchmark, aiming to understand how the low-resource strategies succeeded in terms of performance. The findings of this experiment revealed that our models were able to reach 77.1% of Google Translate's performance, when using the most complete assessment metric (SacreBLEU).

Regarding the qualitative evaluation, this work has considered 2 criteria to analyse the translation quality: the classes of errors, defined with the help of the Brazilian translator, and the sentence complexity, that followed the CEFR criteria. The aspects around how these criteria were defined and evaluated were object of Chapter 5. In the experiments that exploited these dimensions, our findings indicated that some techniques are more prone to lead to some classes of errors than others. In some cases all algorithms identified the same relationship between the model and the classes of errors produced, for instance the Back Translation Transformer showed a proportion of the *Meaning Deviation* errors significantly inferior to other models. All the experiments also showed that BPE is robust against the <*unk*> error, matching our expectations, since this model never produces such a token. The experiments involving Fisher Exact test and Correspondence Analysis point out that the models trained on a reduced dataset are more vulnerable to the *Meaning Deviation* and *Repetition* errors. For all experiments, both reduced models showed a quite similar pattern of errors. The Correspondence Analysis has also indicated that BPE produces the smallest percentage of the *Insertion* error, whilst the opposite happens for Back Translation. The observed agreement between the outcomes of different analysis strategies reinforces the fact that some models are indeed better to mitigate specific classes of errors whilst also being more prone to other error classes. We mentioned here only some the empirical relations found, and must emphasize that the relevance of each finding may vary, depending on their supportive metrics (e.g. explained variance, $p$-value and confidence interval).

Considering the hypothesis that sentences of higher complexity may exert a significant influence on the translation quality, we performed an analysis to validate if the incidence of errors of a particular class of error is higher or smaller when the sentence complexity increases. These experiments showed that for most of the error classes (5 out of 8), a monotonic trend between the percentage of error occurrences and the sentence complexity may be observed. Meanwhile, there are other errors that might appear to have an inverse relationship with complexity, such as the *Insertion* and <*unk*> types, supported by Fisher and Correspondence Analysis. Both Fisher and Correspondence Analysis indicate that the *Repetition* error seems to have no correlation with changes in sentence complexity. It is the only class of error that is potentially more prone to happening depending on the choice of the model rather than with a change in complexity.

We believe that other works in the literature could also benefit from this methodology when trying to better explain the outcomes of other model architectures. Before presenting future study directions, we must stress some challenges that limited our conclusions.

## 6.1 Study limitations

One important limitation of this study is the use of only one reference for both the Tatoeba and the TED Talks datasets. There are situations were the model points out a synonym that isn't mapped in the references, and that is considered as an error when evaluating the BLEU score. This phenomenon was stressed in the qualitative study discussion, which went into further related challenges, such as data quality issues that range from limitations of only one reference to gramatical errors in the official dataset references, and including the interaction bias with the translator. Despite the marginal effect of these issues, we don't believe that they are relevant enough to invalidate any of the conclusions brought by our experiments.

Unfortunately, our sample size of unique sentences could not go much further than 100 sentences, as the interactions with only one translator were under a constrained budget. Adding more translators to the analysis and reaching a consensus could potentially enrich some of the outcomes of our qualitative analysis. The scarcity of Portuguese papers about NMT at the time of writing has led to some additional technical and language intrinsic challenges to contribute to the field, meanwhilst it was also seen as an inspiration to pursue a higher impact.

## 6.2 Next steps

Among the possible directions of improvement for this study, we believe these are the most important: improving the depth of the qualitative analysis by adding more sentences, enriching with more qualitative criteria, or expanding the scope of experiments to contemplate more techniques. We acknowledge the potential of enhancing the criteria if a bigger sample is analysed by a group of translation experts. Maybe there are other types of errors that could be covered, or even a category that was set but could be split into more granular categories to add more details. The qualitative criteria would benefit from receiving more samples and the review of other linguistic experts, thriving diversity and becoming more robust, maybe with the caveat of having to find a consensus to reach an official definition. That would help solving in more depth the subjectivity challenge and bringing more representativeness for the analysis.

Considering that the dataset used for the qualitative study is binary and the variable relationships are complex, algorithms like t-SNE, DBSCAN and Spectral clustering, to name a few, could also be used to extract multidimensional associations. They could be applied to tackle the same problem and extract conclusions and insights that remained hidden in our analysis. We believe that Hierarchical Clustering, Correspondence Analysis and Fisher Exact Test have complemented each other very well, but the possibility of another algorithm revealing unknown correlations exists.

Another study direction could explore expanding the scope to analyse qualitatively more recent NMT algorithms, helping to understand if they are being efficient in addressing the known flaws of the vanilla Transformer, for instance. It would be enriching to find that they improve translation errors of some kind whilst sacrificing others, and perhaps a technique which a different qualitative profile can show up as a better candidate with a low prevalence for many classes of error patterns.

We believe that in the future, models will be much more integrated to people's lives, hence qualitative issues related to prejudice like gender or race bias have to be carefully curated to be identified and filtered out from datasets, mitigating the chances that a model propagates such ideas. A scalable way to achieve this is not available at the time of writing, but analysing qualitative criteria using a similar framework to ours is also an interesting path to be further explored in the NMT field. The applications of NMT are growing in a speed that requires researchers to switch their focus to also consider translation quality, before it impacts crucial aspects like intelligibility and ethics more broadly.

# References

[1] CASELI, H., INÁCIO, M. "NMT and PBSMT Error Analyses in English to Brazilian Portuguese Automatic Translations". In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 3623–3629, Marseille, France, maio 2020. European Language Resources Association. ISBN: 979-10-95546-34-4. Available at: <`https://aclanthology.org/2020.lrec-1.446`>.

[2] ESTRELLA, A., DE OLIVEIRA E SOUZA FILHO, J. B. "Tackling Neural Machine Translation in Low-Resource Settings: a Portuguese Case Study". In: *Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana 2021 (STIL)*, nov 2021. Available at: <`https://sol.sbc.org.br/index.php/stil/article/view/17807`>.

[3] VASWANI, A., SHAZEER, N., PARMAR, N., et al. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN: 9781510860964.

[4] COUNCIL, E. "The CEFR Levels - Council of Europe (COE)". `https://tinyurl.com/cefrlcoe`, 2021. Acessed: 2021-08-12.

[5] GOOGLE. "Translation AI". `https://cloud.google.com/translate`, 2022. Acessed: 2022-06-02.

[6] MIKOLOV, T., CHEN, K., CORRADO, G., et al. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. Available at: <`http://arxiv.org/abs/1301.3781`>.

[7] RONG, X. "Word2vec Parameter Learning Explained", *Computing Research Repository (CoRR)*, v. abs/1411.2738, 2014. Available at: <`http://arxiv.org/abs/1411.2738`>.

[8] JURAFSKY, D., MARTIN, J. H. "Speech and Language Processing (3rd ed. draft)". `https://web.stanford.edu/~jurafsky/slp3/`, 2021. Acessed: 2021-10-14.

[9] MANNING, C. "Lecture 2: Word Vectors, Word Senses, and Neural Classifiers. CS224n: Natural Language Processing with Deep Learning". `http://web.stanford.edu/class/cs224n/slides/cs224n-2021-lecture02-wordvecs2.pdf`, 2021. Acessed: 2021-10-14.

[10] MOSELEY, B., MARKS, P. "Out of the Tar Pit", *Software Practice Advancement (SPA)*, 2006. Available at: <`http://www.shaffner.us/cs/papers/tarpit.pdf`>.

[11] HUTCHINS, W. J. "Machine Translation: A Brief History". In: KOERNER, E., ASHER, R. (Eds.), *Concise History of the Language Sciences*, Pergamon, pp. 431–445, Amsterdam, 1995. ISBN: 978-0-08-042580-1. doi: https://doi.org/10.1016/B978-0-08-042580-1.50066-0. Available at: <`https://www.sciencedirect.com/science/article/pii/B9780080425801500660`>.

[12] KNIGHT, K. "Decoding Complexity in Word-Replacement Translation Models", *Computational Linguistics*, v. 25, n. 4, pp. 607–615, 1999. Available at: <`https://aclanthology.org/J99-4005`>.

[13] CHO, K., VAN MERRIËNBOER, B., GULCEHRE, C., et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, out. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. Available at: <`https://www.aclweb.org/anthology/D14-1179`>.

[14] SHERSTINSKY, A. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", *Computing Research Repository (CoRR)*, v. abs/1808.03314, 2018. Available at: <`http://arxiv.org/abs/1808.03314`>.

[15] BAHDANAU, D., CHO, K., BENGIO, Y. "Neural Machine Translation by Jointly Learning to Align and Translate". In: Bengio, Y., LeCun, Y. (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. Available at: <`http://arxiv.org/abs/1409.0473`>.

[16] MANNING, C. "Lecture 7: Machine Translation, Sequence-to-Sequence and Attention. CS224n: Natural Language Processing with Deep Learning". `http://web.stanford.edu/class/cs224n/slides/cs224n-2021-lecture07-nmt.pdf`, 2021. Acessed: 2021-11-25.

[17] LUONG, T., PHAM, H., MANNING, C. D. "Effective Approaches to Attention-Based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal, set. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. Available at: <`https://www.aclweb.org/anthology/D15-1166`>.

[18] GRAVES, A. "Sequence Transduction with Recurrent Neural Networks", *Computing Research Repository (CoRR)*, v. abs/1211.3711, 2012. Available at: <`http://arxiv.org/abs/1211.3711`>.

[19] SUTSKEVER, I., VINYALS, O., LE, Q. V. "Sequence to Sequence Learning with Neural Networks", *Computing Research Repository (CoRR)*, v. abs/1409.3215, 2014. Available at: <`http://arxiv.org/abs/1409.3215`>.

[20] BRITZ, D., GOLDIE, A., LUONG, M.-T., et al. "Massive Exploration of Neural Machine Translation Architectures". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1442–1451, Copenhagen, Denmark, set. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1151. Available at: <`https://aclanthology.org/D17-1151`>.

[21] MURPHY, K. P. *Probabilistic Machine Learning: An Introduction.* MIT Press, 2022. Available at: <`https://probml.github.io/pml-book/`>.

[22] EISENSTEIN, J. *Introduction to Natural Language Processing.* Adaptive Computation and Machine Learning series. Florence, Italy, MIT Press, 2019. ISBN: 9780262042840. Available at: <`https://books.google.com.br/books?id=72yuDwAAQBAJ`>.

[23] VOITA, E., TALBOT, D., MOISEEV, F., et al. "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned", *Computing Research Repository (CoRR)*, v. abs/1905.09418, 2019. Available at: <`http://arxiv.org/abs/1905.09418`>.

[24] FAN, Y., XIE, S., XIA, Y., et al. "Multi-Branch Attentive Transformer", *Computing Research Repository (CoRR)*, v. abs/2006.10270, 2020. Available at: <`https://arxiv.org/abs/2006.10270`>.

[25] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, v. 15, n. 56, pp. 1929–1958, 2014. Available at: <`http://jmlr.org/papers/v15/srivastava14a.html`>.

[26] CALIXTO, I., LIU, Q., CAMPBELL, N. "Doubly-Attentive Decoder for Multi-modal Neural Machine Translation". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1913–1924, Vancouver, Canada, jul. 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1175. Available at: <`https://www.aclweb.org/anthology/P17-1175`>.

[27] CUI, H., IIDA, S., HUNG, P.-H., et al. "Mixed Multi-Head Self-Attention for Neural Machine Translation". In: *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 206–214, Hong Kong, nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5622. Available at: <`https://www.aclweb.org/anthology/D19-5622`>.

[28] DEVLIN, J., CHANG, M., LEE, K., et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *Computing Research Repository (CoRR)*, v. abs/1810.04805, 2018. Available at: <`http://arxiv.org/abs/1810.04805`>.

[29] BROWN, T., MANN, B., RYDER, N., et al. "Language Models are Few-Shot Learners". In: Larochelle, H., Ranzato, M., Hadsell, R., et al. (Eds.), *Advances in Neural Information Processing Systems*, v. 33, pp. 1877–1901. Curran Associates, Inc., 2020. Available at: <`https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`>.

[30] QUACH, K. "AI Me to the Moon... Carbon Footprint for 'Training GPT-3' Same as Driving to our Natural Satellite and Back". Nov 2020. Available at: <`https://www.theregister.com/2020/11/04/gpt3_carbon_footprint_estimate/`>. Acessed: 2022-02-07.

[31] MEHTA, S., KONCEL-KEDZIORSKI, R., RASTEGARI, M., et al. "DeFINE: DEep Factorized INput Word Embeddings for Neural Sequence Modeling", *Computing Research Repository (CoRR)*, v. abs/1911.12385, 2019. Available at: <`http://arxiv.org/abs/1911.12385`>.

[32] MEHTA, S., GHAZVININEJAD, M., IYER, S., et al. "DeLighT: Deep and Light-weight Transformer". In: *International Conference on Learning Representations*, 2021. Available at: <https://openreview.net/forum?id=ujmgfuxSLrO>.

[33] TAN, Z., WANG, S., YANG, Z., et al. "Neural Machine Translation: A Review of Methods, Resources, and Tools", *Computing Research Repository (CoRR)*, v. abs/2012.15515, 2020. Available at: <https://arxiv.org/abs/2012.15515>.

[34] RANZATO, M., CHOPRA, S., AULI, M., et al. "Sequence Level Training with Recurrent Neural Networks". In: Bengio, Y., LeCun, Y. (Eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. Available at: <http://arxiv.org/abs/1511.06732>.

[35] SHEN, S., CHENG, Y., HE, Z., et al. "Minimum Risk Training for Neural Machine Translation". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1683–1692, Berlin, Germany, ago. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1159. Available at: <https://www.aclweb.org/anthology/P16-1159>.

[36] CHOSHEN, L., FOX, L., AIZENBUD, Z., et al. "On the Weaknesses of Reinforcement Learning for Neural Machine Translation", *Computing Research Repository (CoRR)*, v. abs/1907.01752, 2019. Available at: <http://arxiv.org/abs/1907.01752>.

[37] KE, G., HE, D., LIU, T. "Rethinking Positional Encoding in Language Pre-training", *Computing Research Repository (CoRR)*, v. abs/2006.15595, 2020. Available at: <https://arxiv.org/abs/2006.15595>.

[38] SENNRICH, R., ZHANG, B. "Revisiting Low-Resource Neural Machine Translation: A Case Study". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 211–221, Florence, Italy, jul. 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1021. Available at: <https://www.aclweb.org/anthology/P19-1021>.

[39] SENNRICH, R., HADDOW, B., BIRCH, A. "Neural Machine Translation of Rare Words with Subword Units", *Computing Research Repository (CoRR)*, v. abs/1508.07909, 2015. Available at: <http://arxiv.org/abs/1508.07909>.

[40] BOSTROM, K., DURRETT, G. “Byte Pair Encoding is Suboptimal for Language Model Pretraining”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4617–4624, Online, nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.414. Available at: <`https://aclanthology.org/2020.findings-emnlp.414`>.

[41] LUONG, M.-T., MANNING, C. D. “Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1054–1063, Berlin, Germany, ago. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1100. Available at: <`https://aclanthology.org/P16-1100`>.

[42] CHEN, H., HUANG, S., CHIANG, D., et al. “Combining Character and Word Information in Neural Machine Translation Using a Multi-Level Attention”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1284–1293, New Orleans, Louisiana, jun. 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1116. Available at: <`https://aclanthology.org/N18-1116`>.

[43] BOJANOWSKI, P., GRAVE, E., JOULIN, A., et al. “Enriching Word Vectors with Subword Information”, *Transactions of the Association for Computational Linguistics*, v. 5, pp. 135–146, 2017. doi: 10.1162/tacl_a_00051. Available at: <`https://aclanthology.org/Q17-1010`>.

[44] ZOPH, B., YURET, D., MAY, J., et al. “Transfer Learning for Low-Resource Neural Machine Translation”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1568–1575, Austin, Texas, nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1163. Available at: <`https://www.aclweb.org/anthology/D16-1163`>.

[45] QI, Y., SACHAN, D., FELIX, M., et al. “When and Why Are Pre-Trained Word Embeddings Useful for Neural Machine Translation?” In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 529–535, New Orleans, Louisiana, jun. 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2084. Available at: <`https://www.aclweb.org/anthology/N18-2084`>.

[46] GULCEHRE, C., FIRAT, O., XU, K., et al. "On Integrating a Language Model into Neural Machine Translation", *Comput. Speech Lang.*, v. 45, n. C, pp. 137–148, sep 2017. ISSN: 0885-2308. doi: 10.1016/j.csl.2017.01.014. Available at: <`https://doi.org/10.1016/j.csl.2017.01.014`>.

[47] SENNRICH, R., HADDOW, B., BIRCH, A. "Improving Neural Machine Translation Models with Monolingual Data". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 86–96, Berlin, Germany, ago. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. Available at: <`https://aclanthology.org/P16-1009`>.

[48] PONCELAS, A., SHTERIONOV, D. S., WAY, A., et al. "Investigating Back-Translation in Neural Machine Translation", *Computing Research Repository (CoRR)*, v. abs/1804.06189, 2018. Available at: <`http://arxiv.org/abs/1804.06189`>.

[49] XIA, M., KONG, X., ANASTASOPOULOS, A., et al. "Generalized Data Augmentation for Low-Resource Translation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5786–5796, Florence, Italy, jul. 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1579. Available at: <`https://aclanthology.org/P19-1579`>.

[50] TIEDEMANN, J. "The Tatoeba Translation Challenge - Realistic Data Sets for Low Resource and Multilingual MT". In: Barrault, L., Bojar, O., Bougares, F., et al. (Eds.), *Proceedings of the Fifth Conference on Machine Translation, WMT@EMNLP 2020, Online, November 19-20, 2020*, pp. 1174–1182. Association for Computational Linguistics, 2020. Available at: <`https://aclanthology.org/2020.wmt-1.139/`>.

[51] CETTOLO, M., GIRARDI, C., FEDERICO, M. "WIT3: Web Inventory of Transcribed and Translated Talks", *Proceedings of EAMT*, pp. 261–268, 01 2012.

[52] TIEDEMANN, J. "Parallel Data, Tools and Interfaces in OPUS". In: *Proc. of the Eight International Conf. on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Assoc. (ELRA). ISBN: 978-2-9517408-7-7.

[53] HARTMANN, N., FONSECA, E. R., SHULBY, C., et al. "Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language

Tasks", *Computing Research Repository (CoRR)*, v. abs/1708.06025, 2017. Available at: <`http://arxiv.org/abs/1708.06025`>.

[54] FARES, M., KUTUZOV, A., OEPEN, S., et al. "Word Vectors, Reuse, and Replicability: Towards a Community Repository of Large-Text Resources". In: *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pp. 271–276, Gothenburg, Sweden, maio 2017. Association for Computational Linguistics. Available at: <`https://aclanthology.org/W17-0237`>.

[55] PARKER, R., GRAFF, D., KONG, J., et al. "English Gigaword Fifth Edition", 07 2011. Available at: <`https://doi.org/10.35111/wk4f-qt80`>.

[56] KUDO, T., RICHARDSON, J. "SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing", *Computing Research Repository (CoRR)*, v. abs/1808.06226, 2018. Available at: <`http://arxiv.org/abs/1808.06226`>.

[57] POST, M. "A Call for Clarity in Reporting BLEU Scores". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Belgium, Brussels, out. 2018. Association for Computational Linguistics. Available at: <`https://www.aclweb.org/anthology/W18-6319`>.

[58] LOPER, E., BIRD, S. "NLTK: The Natural Language Toolkit". In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pp. 63–70, Philadelphia, Pennsylvania, USA, jul. 2002. Association for Computational Linguistics. doi: 10.3115/1118108.1118117. Available at: <`https://aclanthology.org/W02-0109`>.

[59] HU, Z., SHI, H., TAN, B., et al. "Texar: A Modularized, Versatile, and Extensible Toolkit for Text Generation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 159–164, Florence, Italy, jul. 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3027. Available at: <`https://aclanthology.org/P19-3027`>.

[60] PASZKE, A., GROSS, S., MASSA, F., et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Proc. Systems 32*, Curran Associates, Inc., pp. 8024–8035, 2019. Available at: <`http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`>.

[61] CASWELL, I., LIANG, B. "Recent Advances in Google Translate". https://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html, 2020. Acessed: 2022-02-03.

[62] JOHNSON, M., SCHUSTER, M., LE, Q. V., et al. "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation", *Transactions of the Association for Computational Linguistics*, v. 5, pp. 339–351, 2017. doi: 10.1162/tacl_a_00065. Available at: <https://aclanthology.org/Q17-1024>.

[63] WU, Y., SCHUSTER, M., CHEN, Z., et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation", *Computing Research Repository (CoRR)*, v. abs/1609.08144, 2016. Available at: <http://arxiv.org/abs/1609.08144>.

[64] DEAN, J. "Google Research: Themes from 2021 and Beyond". https://ai.googleblog.com/2022/01/google-research-themes-from-2021-and.html?m=1#Trend2, 2022. Acessed: 2022-02-21.

[65] ROY, S., CHAKRABARTI, A. "Chapter 11 - A Novel Graph Clustering Algorithm Based on Discrete-Time Quantum Random Walk". In: Bhattacharyya, S., Maulik, U., Dutta, P. (Eds.), *Quantum Inspired Computational Intelligence*, Morgan Kaufmann, pp. 361–389, Boston, 2017. ISBN: 978-0-12-804409-4. doi: https://doi.org/10.1016/B978-0-12-804409-4.00011-5. Available at: <https://www.sciencedirect.com/science/article/pii/B9780128044094000115>.

[66] THEODORIDIS, S., KOUTROUMBAS, K. *Pattern Recognition, Fourth Edition.* Academic Press, 2009. ISBN: 9781597492720.

[67] ROUSSEEUW, P. J. "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis", *Journal of Computational and Applied Mathematics*, v. 20, pp. 53–65, 1987. ISSN: 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. Available at: <https://www.sciencedirect.com/science/article/pii/0377042787901257>.

[68] GREENACRE, M. J., HASTIE, T. J. "The Geometric Interpretation of Correspondence Analysis", *Journal of the American Statistical Association*, v. 82, n. 398, pp. 437–447, 1987. doi: 10.1080/01621459.1987.10478446. Available at: <https://www.tandfonline.com/doi/abs/10.1080/01621459.1987.10478446>.

[69] IZENMAN, A. J. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning.* Springer Publishing Company, Incorporated, 2008. ISBN: 0387781889. Available at: <https://ce.aut.ac.ir/~shiry/lecture/Advanced%20Machine%20Learning/Manifold_Modern_Multivariate%20Statistical%20Techniques%20-%20Regres.pdf>.

[70] NCSS. "Correspondence Analysis". Jan 2022. Available at: <https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Correspondence_Analysis.pdf>. Acessed: 2022-03-07.

# Appendix A

# Academic Publications

This appendix presents the publications produced during the development of this master's dissertation.

- ESTRELLA, A., DE OLIVEIRA E SOUZA FILHO, J. B. "Tackling Neural Machine Translation in Low-Resource Settings: a Portuguese Case Study". In: Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana 2021 (STIL), nov 2021. https://doi.org/10.5753/stil.2021.17807 Available at: <https://sol.sbc.org.br/index.php/stil/article/view/17807>

**Abstract:** Neural machine translation (NMT) nowadays requires an increasing amount of data and computational power, so succeeding in this task with limited data and using a single GPU might be challenging. Strategies such as the use of pre-trained word embeddings, subword embeddings, and data augmentation solutions can potentially address some issues faced in low-resource experimental settings, but their impact on the quality of translations is unclear. This work evaluates some of these strategies on two low-resource experiments beyond just reporting BLEU: errors are categorized on the Portuguese-English pair with thehelp of a translator, considering semantic and syntactic aspects. The BPE sub-word approach has shown to be the most effective solution, allowing a BLEU increase of 59% p.p. compared to the standard Transformer.