

UNIVERSITY OF CAPE TOWN



**Investigating Subword Tokenization Strategies for the
Translation of English into Low-Resource South African
Languages**

Student:

Manala Tyobeka
TYBMAN001

Supervisor:

Dr Juwa Nyirenda

Minor dissertation for the degree of MSc in Statistics and Data Science

Department of Statistical Sciences

November 2025

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

Declaration of Authorship

I, Manala Tyobeka, declare that this thesis titled "**Investigating Subword Tokenization Strategies for the Translation of English into Low-Resource South African Languages**" that was conducted under the supervision of Dr Juwa Nyirenda, is my own work, and where I have consulted with other publications, it is clearly stated. I also declare that this thesis has never been submitted previously for a degree at the University of Cape Town, or any other institution.

Signed:

Signed by candidate

Date: 10 November 2025

Abstract

This thesis investigates how adaptations made to standard subword tokenization algorithms impact the translation performance of Transformer-based Neural Machine Translation (NMT) models that are trained for the translation of text from English into low-resource South African languages. While NMT models are data-hungry; limited research exists to determine how the vocabulary representation, architecture and training strategies of these models can be adapted to better suit South African languages, particularly in conditions with limited time and compute resources. This thesis sources parallel corpora from the Autshumato Project, using roughly 20 000 to 149 000 source-target pairs for model training, that contain translations from English into Northern Sesotho, isiNdebele and Xitsonga. NMT models are then trained on each of the parallel corpora to perform a comparative analysis of standard subword tokenization algorithms with subword regularization, and a task-specific tokenizer for target languages. The task-specific tokenizer narrows the gap between tokenization and translation by leveraging the loss function of a model, that has the same architecture as the NMT model used for translation, to constrain its parameter search. While standard subword tokenization algorithms generally outperform their corresponding subword regularization methods; subword regularization displays superior generalisation capabilities in the face noisy training examples. Relative to standard subword tokenization, the performance of the task-specific tokenizer is comparable; however, it has the capacity to improve the model's predictive accuracy with enough training examples.

Acknowledgements

I would like to sincerely thank Dr Juwa Nyirenda, my supervisor, for his patience, and invaluable guidance throughout this process. There is not much of a dissertation without his support.

I would also like to thank Mr Mahlangu and Mr Boshego for the time taken to help me understand a few of the generated translations by the models deployed in this dissertation.

This research was supported by the National Research Foundation (NRF) of South Africa, through the Masters/Doctoral Scholarship (980000001071)¹.

Lastly, it is important that I acknowledge the many research groups and communities across the continent who are committed to eliminating the under-represented status of African languages in Natural Language Processing research, from Masakhane, SADiLaR, Centre for Text Technology, Data Science for Social Impact, UCT NLP, MakerereNLP, GhanaNLP, NaijaVoices Community, Niger-Volta Language Technologies Institute, Ethio NLP to AfricArXiv, and many others. Inspired by their efforts, my hope is that the research in this dissertation can contribute meaningfully to this vital ecosystem.

¹Please note that the opinions, findings, conclusions and recommendations expressed in this research belong to me. NRF accepts no liability in this regard.

Contents

1	Introduction	10
1.1	Neural Machine Translation	10
1.1.1	Neural Machine Translation within the Context of Low-Resource South African Languages	10
1.2	How Sentences are Represented in Neural Machine Translation	11
1.2.1	Subword Tokenization Algorithms	12
1.2.2	Is Subword Tokenization Enough? A Case for Subword Regularization	12
1.2.3	Can we Narrow the Gap? A Case for Task-specific Tokenization	13
1.3	Thesis Objective	13
2	Background	15
2.1	A Case for Neural Machine Translation	15
2.2	Neural Machine Translation for Low-Resource Languages	16
2.2.1	The Role of Additional Corpora	18
2.2.2	The Role of Model Adaptation	19
2.3	Conclusion	22
3	Data	23
3.1	Autshumato Project for Machine Translation	23
3.1.1	Classification of South African Languages	23
3.1.2	Linguistic Properties of South African Languages	25
3.2	Description of the Parallel Corpora	26
3.2.1	Potential Issues Observed in the Parallel Corpora	27
3.2.2	Preprocessing the Parallel Corpora	28
3.3	Conclusion	31
4	Methods	32
4.1	Standard Subword Tokenization	32
4.1.1	Byte-Pair Encoding	33
4.1.2	Unigram Language Modelling	34
4.1.3	Threshold on Joint Vocabulary	35
4.2	Subword Regularization	36
4.2.1	BPE-Dropout	36
4.2.2	Unigram Language Modelling with Subword Regularization	37
4.3	Task-specific Subword Tokenization	38
4.3.1	Target Neural Unigram Language Model	39
4.4	Neural Machine Translation	40
4.4.1	Architecture of the Transformer	42
4.4.2	Embedding Representations	44
4.4.3	Self-Attention Mechanism	45
4.4.4	Feed-Forward Neural Network	46
4.4.5	Output Probabilities	47
4.4.6	Add and Norm	47
4.4.7	Training	48
4.4.8	Evaluation	51
4.5	Conclusion	54
5	Experimental Setup	56

5.1	Experimental Design	56
5.2	Training Setup: Transformer	56
5.2.1	Software	57
5.2.2	Hardware	57
5.3	Training Setup: Tokenization	58
5.3.1	Experiment 1: Baseline Models	58
5.3.2	Experiment 2: Subword Regularization	59
5.3.3	Experiment 3: Task-Specific Tokenizer	60
5.4	Evaluation Setup	61
5.5	Conclusion	62
6	Results & Discussion	63
6.1	Experiment 1: Standard Subword Tokenization	63
6.1.1	English-to-Northern Sesotho	63
6.1.2	English-to-isiNdebele	64
6.1.3	English-to-Xitsonga	66
6.1.4	Discussion	67
6.2	Experiment 2: Subword Regularization	69
6.2.1	English-to-Northern Sesotho	69
6.2.2	English-to-isiNdebele	70
6.2.3	English-to-Xitsonga	70
6.2.4	Discussion	71
6.3	Experiment 3: Task-Specific Tokenization	72
6.3.1	English-to-Northern Sesotho	72
6.3.2	English-to-isiNdebele	72
6.3.3	English-to-Xitsonga	73
6.3.4	Discussion	73
6.4	Conclusion	74
7	Conclusion	75
7.1	Introduction	75
7.2	Conclusions	75
7.3	Limitations	76
7.4	Future Work	76
A	A Case for Subword Tokenization	78
B	Expectation-Maximisation	81
C	Viterbi Algorithm	82
D	Forward-Filtering and Backward-Sampling Algorithm	83
E	Forward-DP and Backward-A* Algorithm	84
F	Model Predictions	85
G	Post-Processing Translations	89
H	Code	90

List of Figures

2.1	Participants involved in developing and deploying an effective Machine Translation (MT) model (Nekoto et al., 2020).	18
3.1	Configuration of the geographic zones in Africa used to classify Bantu languages (Edricson, 2007).	24
4.1	A configuration of how the Target Neural Unigram Language Model (Target NULM) is jointly trained with an NMT model.	41
4.2	Configuration of the encoding component in the Transformer.	43
4.3	Masked Scaled Dot-Product Attention	46
4.4	Configuration of the decoding component in the Transformer.	55
5.1	Summary of how the standard subword tokenizations algorithms are implemented to prepare sentences for the training and validation of the translation models. The red highlighted blocks refer to the steps taken by the fairseq-preprocess command-line.	59
5.2	Summary of how the subword regularization algorithms are implemented to prepare sentences for the training and validation of the translation models. The red highlighted blocks refer to the steps taken by the fairseq-preprocess command-line.	59
5.3	Summary of how the task-specific tokenization algorithm is implemented to prepare sentences for the training and validation of the translation models. The red highlighted blocks refer to the steps taken by the fairseq-preprocess command-line.	60
5.4	Summary of the machine learning pipeline followed to develop the translation models in this thesis.	61
A.1	Distributions of lengths of English and isiNdebele segments.	78
A.2	Distribution of English-to-isiNdebele segment pair ratios.	79
A.3	Distributions of lengths of whitespace characters on the English and isiNdebele sides of the training corpus, along with the length of subword tokens from a Byte-Pair Encoding (BPE)-generated vocabulary.	79
A.4	Word cloud of subword tokens from a BPE-generated vocabulary.	80

List of Tables

3.1	How South African official spoken languages are further grouped within the <i>Zone S.</i> language classification.	24
3.2	Example of how grammatical meaning is conveyed in an agglutinating language compared to an isolating language.	25
3.3	Example of how the different writing systems in isiNdebele, English, Xitsonga and Northern Sesotho compare to one another.	26
3.4	Examples of inconsistent usage of letter casing in the English-to-isiNdebele parallel corpus.	28
3.5	Example of how punctuation normalization is applied to simplify inconsistent punctuation usage in the English-to-Northern Sesotho parallel corpus.	30
3.6	Overview of the parallel corpora used in this thesis before and after preprocessing.	31
4.1	Example of how the BPE tokenization algorithm is applied to a corpus.	33
4.2	Example of how an unseen word is encoded with an unknown symbol.	34
4.3	Example of a merge table.	36
4.4	Summary of the order of computational complexity, sequential operations and maximum path lengths for a self-attention mechanism, recurrent neural network and convolutional neural network (Vaswani et al., 2023).	42
5.1	Comparison between the hyperparameter configuration for training a baseline Transformer (Vaswani et al., 2023) and the medium-depth Transformer used in this thesis for the translation based on the work of van Biljon et al. (2020). Note that * indicates that we used the Fairseq toolkit’s default value.	58
5.2	Hyperparameter configuration for target language tokenizer. Note that * indicates that we used the default value in the optok4at/machineTranslation repository.	61
6.1	An English-to-Northern Sesotho sentence pair from the test corpus alongside the predictions of the source sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and Unigram Language Modelling (ULM), respectively.	64
6.2	An English sentence from the global test corpus alongside the translations generated of the segment in Northern Sesotho by four professional translators, and the predictions of the sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.	65
6.3	An English-to-isiNdebele sentence pair from the test corpus alongside the predictions of the source sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.	65
6.4	An English sentence from the global test corpus alongside the translations generated of the sentence in isiNdebele by four professional translators, and the predictions of the sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.	66
6.5	An English-to-Xitsonga sentence pair from the test corpus alongside the predictions of the source sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.	67
6.6	An English sentence from the global test corpus alongside the translations generated of the sentence in Xitsonga by four professional translators, and the predictions of the sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.	68

6.7	The Bilingual Evaluation Understudy (BLEU) scores of the Transformers deployed for each language pair in the baseline experiment. Each score includes its 95% confidence interval. The last column contains the p -value for the paired significance tests that were conducted. The asterisk indicates that based on the p -value, the two models were found to behave differently, and the bold font indicates that this is best score and it is statistically significant from the baseline model (in this case, the BPE Transformer is treated as the baseline model).	69
6.8	The BLEU scores of the Transformers deployed for each language pair in the baseline experiment. Each score includes its 95% confidence interval. The last column contains the p -value for the paired significance tests that were conducted. The asterisk indicates that based on the p -value, the two models were found to behave differently, and the bold font indicates that this is best score and it is statistically significant from the baseline model in the first experiment.	72
6.9	The BLEU scores of the Transformers deployed for each language pair in the baseline experiment. Each score includes its 95% confidence interval. The last column contains the p -value for the paired significance tests that were conducted. The asterisk indicates that based on the p -value, the two models were found to behave differently, and the bold font indicates that this is best score and it is statistically significant from the baseline model in the first experiment.	73
F.1	An English-to-Northern Sesotho sentence pair from the test corpus alongside the predictions of the source sentence generated by each of the Transformers.	85
F.2	An English sentence from the global test corpus alongside the translations generated of the sentence in Northern Sesotho by four professional translators, and the predictions of the sentence generated by each of the Transformers.	86
F.3	An English-to-isiNdebele sentence pair from the test corpus alongside the predictions of the source sentence generated by each of the Transformers.	86
F.4	An English sentence from the global test corpus alongside the translations generated of the sentence in isiNdebele by four professional translators, and the predictions of the sentence generated by each of the Transformers.	87
F.5	An English-to-Xitsonga sentence pair from the test corpus alongside the predictions of the source sentence generated by each of the Transformers.	87
F.6	An English sentence from the global test corpus alongside the translations generated of the sentence in Xitsonga by four professional translators, and the predictions of the sentence generated by each of the Transformers.	88

List of Acronyms

MT	Machine Translation
NMT	Neural Machine Translation
RBMT	Rule-based Machine Translation
EBMT	Example-based Machine Translation
SBMT	Statistical-based Machine Translation
NLP	Natural Language Processing
BPE	Byte-Pair Encoding
ULM	Unigram Language Modelling
ESA	Enhanced Suffix Array
EM	Expectation-Maximization
E-Step	Expectation
M-Step	Maximization
HMM	Hidden Markov Model
DP	Dynamic Programming
FFBS	Forward-Filtering and Backward-Sampling
Target NULM	Target Neural Unigram Language Model
KL	Kullback-Leibler
MCE	Multi-class Cross-Entropy
LSTM	Long Short-Term Memory
BLEU	Bilingual Evaluation Understudy
SADiLaR	South African Centre for Digital Language Resources
CText	Centre for Text Technology
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
OOV	out-of-vocabulary
SSMT	Subword Segmental Machine Translation
DPE	Dynamic Programming Encoding

Chapter 1

Introduction

This chapter provides background on the research being conducted in this thesis, briefly describing Neural Machine Translation (NMT) and its effectiveness within the context of low-resource languages. It outlines the purpose of this research, describing how sentences can be represented for the purpose of NMT. Further, it motivates the reason for conducting such an investigation. The chapter concludes with a brief description of the objectives of this thesis, and how they will be evaluated.

1.1 Neural Machine Translation

Language facilitates the exchange of information between people. If we all spoke different languages, then language translation is what keeps people from all corners of the world connected. Machine Translation (MT), within the field Natural Language Processing (NLP), is an approach to automating the translation of sentences from one language into another.

Over the years, MT has seen a shift from rule-based techniques to example- and more statistical-based approaches, and recently to the application of neural networks. NMT is the current state-of-the-art approach to MT. It uses an extensive parallel corpus¹ to train artificial neural networks responsible for modelling the translation from one language into another (Kalchbrenner and Blunsom, 2013; Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014; Sutskever et al., 2014). This approach to MT achieves promising translation performance without manually defining explicit rules for translation (unlike in rule-based approaches). Instead, it defines a statistical distribution over a translation, and uses a parallel corpus to estimate the distribution for the purpose of generating new translations in the future (unlike in example-based approaches). The components of an NMT model used to fit the statistical distribution over a translation are then *jointly* trained to optimise the objective function linked to the model (unlike in statistical-based approaches).

More than 7,000 languages are spoken across the globe (Eberhard and Fennig, 2024); however, research in statistical- and neural network-based NLP models has predominantly focused on a limited number of high-resource languages (Magueresse et al., 2020). In South Africa, for instance, there are 11 official spoken languages. Despite their constitutional recognition, many of these languages are significantly under-represented in NLP research. This disparity stems from their classification as low-resource languages—languages for which there is insufficient computational data available to train modern NLP models effectively. The absence of sufficient data not only leads to poor model performance on these languages, but also drives a declining interest in developing NLP tools for them, thereby reinforcing their under-represented status in NLP research.

1.1.1 Neural Machine Translation within the Context of Low-Resource South African Languages

The success of NMT over earlier MT approaches has been most evident in high-resource language pairs. However, this performance advantage does not extend to low-resource language pairs, where

¹A *parallel* corpus is a collection of sentences in a source language, with each sentence stored alongside its translation written in a target language.

NMT models often under-performed compared to traditional statistical-based MT methods when trained on the limited data from these languages (Koehn and Knowles, 2017; Sennrich and Zhang, 2019). Koehn and Knowles (2017) attribute the poor performance of NMT models in low-resource settings to their heavy reliance on large volumes of training data for optimal results. While this observation holds merit, another key factor is that NMT models for low-resource language pairs are often built using vocabulary representations, model architectures and training strategies originally optimised for high-resource languages. This practice is largely driven by the substantial time and computational resources required to train NMT models effectively. Conducting comprehensive hyperparameter tuning, such as a grid or random search, is often not feasible for many NLP researchers working in low-resource contexts, further limiting the potential for model adaptation and performance improvement. In the case of NMT models trained on South African languages, the models also generally inherit the vocabulary representations, architectures and training strategies of larger models trained on high-resource language pairs. The translation performance of these models is further compromised by the significant morphological, semantic and syntactical differences that exist between South African languages and high-resource languages in NLP research (Nekoto et al., 2020).

Improvements to the translation performance of NMT models trained in low-resource data settings have largely relied on the use of additional monolingual², parallel or multilingual³ corpora. The drawback with such attempts is that their effectiveness is largely dependent on the availability of suitable additional corpora, that is that the additional corpus needs to be large enough, contain a language closely related to, and/or address the same domain as the low-resource language (Sennrich and Zhang, 2019). For NMT models trained on South African languages, translation performance has been shown to benefit from the use of additional corpora, often requiring little-to-no hyperparameter tuning specifically for these languages (Nyoni and Bassett, 2021; Elmadani et al., 2022; Mojapelo et al., 2023). The additional corpora have been used to implement established best practices to improve the translation performance of NMT models trained on low-resource language pairs. These practices include transfer learning, multilingual learning, using models developed through multilingual learning to facilitate zero-shot learning, back-translation and word replacement.

Additionally, NMT models are generally data-hungry, which presents a challenge in low-resource language settings. Sennrich and Zhang (2019) and Araabi and Monz (2020) have shown that by carefully adjusting aspects of an NMT model, such as vocabulary representation, model architecture and regularization strategies, it is possible to significantly improve translation performance, even in data-scarce scenarios. Despite this, there is limited research examining the effectiveness of such adjustments for South African languages. This thesis seeks to contribute to addressing this gap by investigating how tailored modifications to NMT models can enhance translation performance for low-resource South African language pairs.

1.2 How Sentences are Represented in Neural Machine Translation

To better optimise NMT models for translating English into low-resource South African official languages, this thesis focuses on how sentences are represented in the translation process. In NMT, sentences are encoded as a sequence of smaller units, known as tokens. A vocabulary comprises a list of unique tokens that are used to construct sentences from a given corpus. The method used to generate these tokens is referred to as tokenization. Tokens are characterised as words, subwords or individual characters, and how these tokens are characterised has a direct impact on how an NMT model interprets and processes text in the corpus, which in turn influences both its ability to learn language structure and its overall translation performance.

- When **compared to tokens generated via character-based tokenization**, subword tokens are generally more meaningful. Further, sentences represented as a sequence of subword tokens are more compact, allowing for more efficient storage of sentence representations and decoding (Sennrich et al., 2016b); and
- When **compared to tokens generated via word-based tokenization**, subword tokens

²A *monolingual* corpus is a collection of sentences in a language.

³A *multilingual* corpus is a collection of sentences in a source language, with each sentence stored alongside its translations into multiple target languages.

attempt to capture how words in a language are formed, allowing for an NMT model to better learn the relationship between similar words, and for the model to generate unseen words during testing (Sennrich et al., 2016b). The size of the vocabularies generated using subword tokenization algorithms are relatively smaller, which means that less memory is required to store vocabularies (Sennrich et al., 2016b). Lastly, the embedding representations that an NMT model generates from subword tokens can encode a more comprehensive set of features in a word and/or sentence.

The datasets used in this thesis contain parallel translations from English into three low-resource South African languages, namely isiNdebele, Northern Sesotho and Xitsonga. These languages are classified as either *Zone S*. (Guthrie and Institute, 1948; Maho, 2009) or Southern Bantu languages (Janson, 1991-1992), both within the broader the Niger-Congo language family (Eberhard and Fennig, 2024). English, on the other hand, is a West Germanic branch of the Indo-European language family (Eberhard and Fennig, 2024). These language families differ significantly with respect to word formation processes and writing systems, which affects how meaning is constructed and conveyed. As a result, the alignment between English (the source language) and the three Southern Bantu languages (the target languages) is generally not one-to-one. To address this challenge, this thesis adopts a subword-level tokenization strategy that represents tokens as subwords, allowing the NMT model to be better equipped to learn the internal structure of words and generate more accurate mappings between source and target languages (Sennrich et al., 2016b). This subword-based approach also strikes a balance between the strengths and limitations of word-level and character-level tokenization techniques.

1.2.1 Subword Tokenization Algorithms

Two commonly used standard subword tokenization algorithms in NMT modelling are:

1. BPE (Sennrich et al., 2016b); and
2. ULM (Kudo, 2018).

Both subword tokenization algorithms construct a vocabulary by performing a search over the training parallel corpus to find the set of subword tokens that appear in the corpus most frequently. Using the prepared vocabularies, sentences are then mapped to unique subword sequence representations in preparation to be processed by an NMT model. The translation performance of NMT models that use BPE or ULM for tokenization are generally comparable (Kudo, 2018). However, BPE is more commonly used than ULM due to its relative ease to compute (Provilkov et al., 2020).

Since the selection of subword tokens for a vocabulary is based on their frequency in the training parallel corpus, rather than on manually-crafted rules outlining the word formation processes in each language, the subword sequence representations of sentences generated by these standard subword tokenization algorithms may be vulnerable to segmentation errors (Kudo, 2018). The quality of the subword tokens selected for a vocabulary impacts a model’s ability to learn how words and sentences are constructed. In low-resource data settings, segmentation errors are more common due to there being limited examples in the training corpus to inform more appropriate choices for subword tokens.

1.2.2 Is Subword Tokenization Enough? A Case for Subword Regularization

Subword regularization is introduced as a data-driven solution to increase an NMT model’s robustness to segmentation errors, by injecting noise into the model during training (Kudo, 2018). This process involves generating multiple subword sequence representations of sentences in a training corpus, and exposing an NMT model to the different representations during the training. In this thesis, we will consider the following two modifications to the standard BPE and ULM tokenization algorithms, respectively, to output multiple subword sequence representations of sentences in a training corpus:

1. BPE-Dropout (Provilkov et al., 2020); and
2. ULM with subword regularization (Kudo, 2018).

In low-resource data settings, subword regularization helps prevent the model from overfitting on potentially sub-optimal subword sequence representations. Kudo (2018) applies ULM with subword regularization to corpora of different sizes for NMT, and compares the performance of NMT models whose corpora is preprocessed using standard BPE and ULM. The improvements to the translation models were especially prominent in low-resource data settings. Provilkov et al. (2020) report a further improvement to the translation performance of NMT models using BPE-Dropout, compared to using ULM with subword regularization. This is remarkable, since BPE-Dropout retains the simplicity of BPE in its computation.

1.2.3 Can we Narrow the Gap? A Case for Task-specific Tokenization

For a given vocabulary, the process of representing sentences as sequences of subword tokens, using standard subword tokenization or subword regularization methods, is performed separately from the training of an NMT model. In both cases, it is informed by the training corpus, ignoring any other context (Hiraoka et al., 2021). To narrow the gap between translation and tokenization, this thesis considers applying a **task-specific tokenizer for the target language** that accounts for translation (the task) in its development.

The task-specific tokenizer is based on a neural network that is responsible for modelling the probability of observing a subword token for a given vocabulary. Using the estimated probabilities linked to each token, a sequence of tokens can be sampled to represent a sentence. To account for translation in its development, Hiraoka et al. (2021) experiment with developing an NMT model that also tokenizes sentences. This model comprises a source and/or target language tokenizers that are jointly trained with an NMT model.

This joint model significantly increases the number of parameters in the translation model, leading to significantly longer training times. Similar to the application in Hiraoka et al. (2021), to reduce lengthy training times, the task-specific tokenizer in this thesis is pre-trained to fit the distribution of a unigram language model. However, the pre-trained tokenizer is then jointly trained with an NMT model that has the same architecture as the NMT model that will be developed later for translation. The goal is to leverage the loss function of the NMT model to fine-tune the parameters of the task-specific tokenizer, so that it is trained to assign higher probabilities to subword sequence representations of target sentences that minimise the NMT model’s loss function; not to also completely optimise the NMT model. By setting up the problem this way, we hope to avoid the need for lengthy training times observed in Hiraoka et al. (2021).

1.3 Thesis Objective

The lens through which this thesis addresses the limitations of NMT models for low-resource South African languages is by investigating whether these models can benefit from carefully tuning of the vocabulary representation, architecture and/or training strategies to better suit these languages. More specifically, this thesis investigates modifications made to standard subword tokenization algorithms, and how they can impact the translation performance of NMT models trained for the translation of English into South African languages. This thesis, however, will only consider a subset of these languages for the investigation, which includes isiNdebele, Northern Sesotho and Xitsonga. The structure of the investigation is as follows:

1. **Prepare baseline models:** Training and comparing two NMT models for each language pair, such that each language pair is tokenized using the standard BPE (Sennrich et al., 2016b) and ULM (Kudo, 2018) tokenization algorithms.
2. **Examine the impact of subword regularization on translation performance:** Applying BPE-Dropout (Provilkov et al., 2020) and ULM with subword regularization (Kudo, 2018) to generate and to train NMT models on multiple subword sequence representations of sentences in the training corpus, with the goal of increasing the models’ robustness to segmentation errors, and in turn improving their translation performance.
3. **Examine the impact of a task-specific tokenization strategy on translation performance:** Preparing a tokenizer for the target language that leverages the loss function of an NMT model to optimise the set of parameters in its neural network architecture (Hiraoka et al., 2021). Using the trained tokenizer to then preprocess sentences on the target side

of the corpus, while the source side is preprocessed using the standard ULM tokenization algorithm for the training an NMT model for translation.

The translation performance of all the models developed in this thesis will be evaluated using the Bilingual Evaluation Understudy (BLEU) score (Papineni et al., 2002). This score is an automatic, language-independent evaluation metric for MT (Papineni et al., 2002). An increase in the BLEU score, indicates an improvement in a model’s translation performance. In the absence of professional translators to determine whether a difference in BLEU scores reflects a true difference in two models’ translation quality, this thesis will also make use of paired bootstrap resampling (Koehn, 2004) to estimate the statistics required to conduct a paired statistical significance test.

The rest of this thesis is organised as follows: Chapter 2 reviews existing literature within the domain of this research, providing an overview of NMT and the developments related to optimising NMT models for low-resource language pairs. Chapter 3 describes the parallel corpora used in this thesis and how each corpus is prepared for tokenization. Chapter 4 outlines the methods used to address the objectives of this thesis. Chapter 5 details the experimental design, and the evaluation setup to assess the impact of each experiment on the translation model. Chapter 6 discusses the results of the experiments and their implications on NMT for low-resource South African languages. Finally, Chapter 7 concludes the thesis by summarising key findings, limitations and suggesting future research directions for NMT for low-resource South African languages.

Chapter 2

Background

This chapter reviews literature relevant to NMT for low-resource languages, and more specifically low-resource South African languages, to frame the work in this thesis relative to what has been done before. This process involves referring to literature to understand why NMT is the popular approach for MT today, its limitations in low-resource data settings, and what has been done to address these limitations.

2.1 A Case for Neural Machine Translation

The goal of MT is to automate the translation of sentences from a source language into a target language. Over the years, MT has seen a shift from rule-based techniques to example- and more statistical-based approaches (Hutchins, 2001), and recently to the application of neural networks (Kalchbrenner and Blunsom, 2013; Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014; Sutskever et al., 2014). Rule-based Machine Translation (RBMT) involves defining a set of rules to identify structural features in a source sentence, and outlining an appropriate mapping into the target language (Nirenburg, 1989; Hutchins, 2001). It uses a bilingual dictionary to perform the word-for-word replacements (Hutchins, 2001). The three popular approaches for this kind of MT approach include direct, transfer and interlingua. While RBMT can achieve satisfactory translation performance; its performance is largely dependent on the quality of the rules defined, which require extensive knowledge of both languages and translation to generate (Lagarda et al., 2009).

Nagao (1984) introduces Example-based Machine Translation (EBMT) with the aim of eliminating the process of manually defining explicit rules to map translations from one language into another; and instead, relying on a database of basic phrases in the source language alongside their corresponding translations. In EBMT (Nagao, 1984), a source sentence is translated by matching segments of the sentence with similar phrases in the database. The corresponding translations of the selected phrases are then identified from the database. Before the phrase translations are merged to form a translation of the source sentence, they are modified to fit the requirements of how a sentence is to be structured in the target language. While an EBMT model does not require explicit rules to map translations from one language into another; it still requires dictionaries, thesauri and an understanding of linguistics to appropriately modify the translations of the selected phrases (Nagao, 1984).

Brown et al. (1990) introduce Statistical-based Machine Translation (SBMT), and similarly to EBMT, it is a data-driven approach to MT. The approach is based on defining a probability distribution over the translation of a source language into a target language, and using a large database of sentences in the source language alongside their corresponding translations to estimate the distribution. Given a source sentence, an SBMT model (Brown et al., 1990) assumes that every sentence in the target language is a potential translation of the source sentence with some probability. The goal of the model is to find the target sentence that maximises the conditional probability of observing a target sentence given the source sentence. Using Bayes' theorem, maximising the conditional probability is equivalent to maximising the product of the conditional probability of observing the source sentence given the target sentence, that is estimated using a translation model component, and the marginal probability of observing the target sentence, that

is estimated using a language model component (Brown et al., 1990; Lagarda et al., 2009; Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014). From the time it was first introduced, SBMT has since been extended to include other model components that learn additional features to improve the performance of the overall SBMT model, with all of the model components being combined within a log-linear framework (Lagarda et al., 2009; Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014). The four popular approaches to this kind of MT include word-, phrase-, syntax- and hierarchical-based SBMT. When compared to RBMT and EBMT models, SBMT models are far more computationally expensive to train. However, these models can produce reliable translations without explicitly defining a set of rules to identify structural features and outline appropriate mappings from one language into the other; unlike RBMT (Costa-jussà et al., 2010). Moreover, these models infer a probability distribution that enable it to perform a more efficient search for a translation of a source sentence; whereas EBMT models apply case-based reasoning to perform a more rudimentary search over the entire database, matching segments of the source sentence with phrases in the database, and merging translations of the selected phrases to generate a final translation of the source sentence (Costa-jussà et al., 2010). By inferring a probability distribution, SBMT enables its models' to generalise better to unseen source sentences, and increase its robustness to noisy source sentences.

In spite of the success of SBMT, work was done to look into how artificial neural networks could be used to model translation (Kalchbrenner and Blunsom, 2013; Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014; Sutskever et al., 2014). Similarly to SBMT, models developed in NMT define a probability distribution over the translation of a source language into a target language, and it uses a large database of sentences in the source language alongside their corresponding translations to estimate the distribution. Unlike in SBMT, all components of NMT models are jointly trained to maximise the probability of observing a target sentence given a source sentence (Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014). While the initial translation performance of early NMT models were underwhelming (Cho, van Merriënboer, Bahdanau and Bengio, 2014; Bahdanau et al., 2016), these models have since surpassed state-of-the-art SBMT models, and are the popular choice for MT given the fast, growing research in this specific domain (Wu et al., 2016; Kalchbrenner et al., 2017; Zhou et al., 2016; Shazeer et al., 2017).

2.2 Neural Machine Translation for Low-Resource Languages

According to Eberhard and Fennig (2024), there are currently 7,164 living languages spoken in the world. However, research into statistical- or neural network-based NLP models largely focus on 20 of those languages, that is roughly 0.3% of languages (Magueresse et al., 2020). South African languages, except for English, form part of the large group of languages that are under-studied (Martinus and Abbott, 2019b). This on account of these languages being low-resource languages (Magueresse et al., 2020). Depending on the context, the term "low-resource" is used in literature to refer to various phenomena: a language whose indigenous speakers and use cases are in decline, and/or a language lacking in formal writing systems (related to the term *endangered* language); a language that is spoken in a region with a shortfall of competent educators, translators and/or systems in place to deliver the language (related to the term *critical* language); a language that lacks in computational data measured relative to another language or in absolute terms (related to the term *low-density* language); a language that does not rank highly against a metric defined as the product of the number of indigenous speakers of the language in a country and the country's Gross National Product, called *Gross Language Product* (Hammarström, 2009) (related to the term *low-affluence* language); a language that is not commonly taught, but is of national interest in a country outside of the region that the language is indigenous to (related to the term *less commonly taught* language); and/or a language that does not have adequate systems available to enable a swift transition into the language in the case of a sudden and/or disastrous event (related to the term *surprise* language) (Cieri et al., 2016). However, for the remainder of this thesis, a low-resource language will refer to language that does not have sufficient computational data publicly available to effectively train state-of-the-art NLP models. In the case of NMT, the models typically comprise tens of millions of parameters as a baseline (Vaswani et al., 2023).

A dataset used to train statistical- and neural network-based MT models is referred to as a corpus. A monolingual corpus contains lines of text in one language. A parallel corpus contains lines of text in one language, referred to as the source language. Each line of text is aligned to a translation, in what is referred to as the target language. In a multilingual corpus, each line of text in the source

language is aligned to its translation for multiple languages. Over the years, the application of NMT models on public benchmark corpora have achieved leading translation performance relative to earlier approaches to MT (Wu et al., 2016; Kalchbrenner et al., 2017; Zhou et al., 2016; Shazeer et al., 2017). The success of these models, however, has largely been observed in the context of high-resource language pairs (Koehn and Knowles, 2017; Sennrich and Zhang, 2019). Koehn and Knowles (2017) compare the performance of an NMT model whose implementation is based on the Recurrent Neural Network (RNN) Encoder-Decoder architecture (Bahdanau et al., 2016), with two phrase-based SBMT models based on the works of Williams et al. (2016) and Ding et al. (2016), over subsets of different sizes of an English-to-Spanish parallel corpus¹. The results of the comparison indicate that the NMT model performs significantly worse than the statistical-based approaches in low-resource data settings. Koehn and Knowles (2017) attribute this poor performance to the NMT model’s size which requires a *large enough* corpus to adequately train the model.

Beyond the architecture of state-of-the-art NMT models, Martinus and Abbott (2019b), Orife et al. (2020) and Nekoto et al. (2020) highlight some of the further barriers affecting NMT research in the African context, all of which are also relevant to South African languages. These barriers include the lack of written and digital resources across various domains to develop parallel corpora to train NMT models (Martinus and Abbott, 2019b; Orife et al., 2020). The few resources that are available are oftentimes inaccessible, that is that the resources are restricted to specific institutions, are behind paywalls or are published in journals that have not yet been digitised (Martinus and Abbott, 2019b; Orife et al., 2020). When compared to the ongoing NMT research being conducted in high-resource data settings; there is limited research being published on African languages due to the limited number of researchers interested in these languages (Nekoto et al., 2020). In some cases, the research that is published does not include the relevant code and data, limiting the ability of the broader research community to reproduce the work and improve current benchmark scores (Martinus and Abbott, 2019b; Orife et al., 2020; Nekoto et al., 2020). Lastly, efforts to fund NMT research for African languages are few and scattered (Martinus and Abbott, 2019b; Orife et al., 2020). The barrier that a lack of funding creates is further exacerbated by the fact that the necessary participants required to develop and deploy an effective MT model goes beyond the NLP researcher who builds and trains the model; it extends to the individuals responsible for data creation, model evaluation and the end-user. These participants require resources to contribute effectively to this process. Figure 2.1 is sourced from Nekoto et al. (2020), and presents the various participants involved in developing and deploying an effective MT model, how they interact with one another and the external resources that each participant requires to contribute effectively. While not included in the figure, the end-user is who the MT model will be deployed to. The external resources that they require include access to the technology and stable electricity (Nekoto et al., 2020). Based on how the participants interact with one another, it becomes clear how neglecting any aspect of this process can negatively impact the quality and usefulness of the MT model being developed.

In low-resource data settings, it is common practice for the NMT models developed to inherit the vocabulary representation, model architecture and training strategies of models’ that have been optimised on high-resource language pairs. This is on account of the steep time and computational costs associated with training an NMT model (Sennrich and Zhang, 2019). Performing a grid or random search to find the optimal hyperparameter settings for a low-resource language pair is generally not feasible for many NLP researchers, with limited time, resources and financial backing. This is also the case for NMT models trained on South African languages. As a result, the translation performance of these models is not only compromised by the scale of the inherited architecture, and the assumptions made by way of the vocabulary representations and training strategies implemented which are not developed with smaller corpora in mind; the performance is further compromised by the the morphological, semantic and syntactical differences that exist between South African languages and high-resource languages in NLP research (Nekoto et al., 2020).

¹The implementation of the RNN Encoder-Decoder architecture has been modified, via the Nematus toolkit (Sennrich, Firat, Cho, Birch, Haddow, Hitschler, Junczys-Dowmunt, Läubli, Miceli Barone, Mokry and Nădejde, 2017), to improve the model’s translation performance, whilst optimising computational efficiency and its capacity to scale.

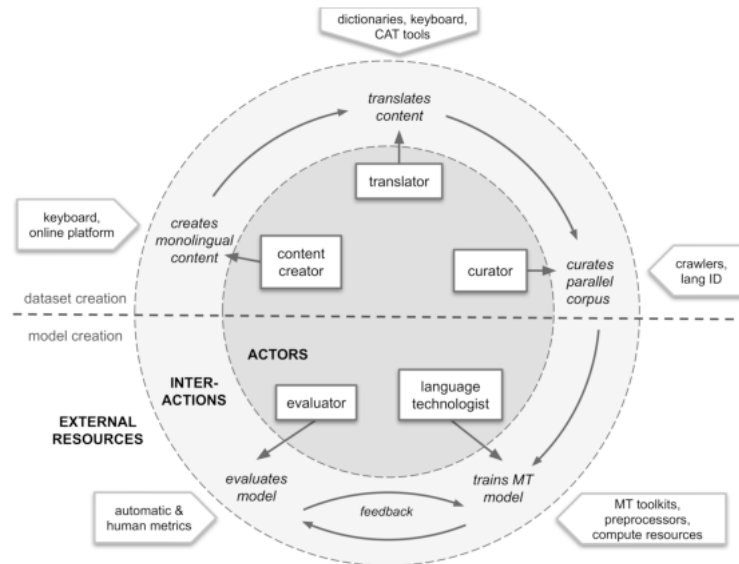


Figure 2.1: Participants involved in developing and deploying an effective MT model (Nekoto et al., 2020).

2.2.1 The Role of Additional Corpora

Several successful attempts to improve the translation performance of NMT models trained on low-resource language pairs have largely relied on the use of additional monolingual, parallel or multilingual corpora (Sennrich and Zhang, 2019). Similar success has also been observed in the context of NMT models trained on South African languages, often with little-to-no hyperparameter tuning (Nyoni and Bassett, 2021; Elmadani et al., 2022; Mojapelo et al., 2023).

Nyoni and Bassett (2021) use additional parallel corpora to apply transfer learning (Torrey and Shavlik, 2009; Zoph et al., 2016), multilingual learning, and models developed through multilingual learning to facilitate zero-shot learning (Dong et al., 2015; Johnson et al., 2017; Ha et al., 2017) to improve the translation performance of Transformer-based NMT models (Vaswani et al., 2023) trained for the translation of English-to-isiZulu, a language pair whose parallel corpus contains approximately 30,000 sentence pairs². The investigation leverages additional parallel corpora that are larger and contain a language that belongs to the same language family as isiZulu, to boost the translation performance of the models developed for the translation of English into isiZulu³.

Elmadani et al. (2022) leverage multiple parallel corpora to develop a large multilingual NMT model (Liu et al., 2020) for the translation into and from English and eight Southern African languages^{4, 5}. The multilingual model uses back-translated sentences to increase its robustness to noise, and synthetic training sentence pairs that are generated using an earlier trained multilingual model for the additional translation directions in the model that have no corresponding parallel corpora⁶. The aim of the investigation is to take advantage of the various translation directions, and the induced noise via the back-translated and synthetic training sentences, to learn a more robust shared representation of all languages used to train the multilingual model, and ultimately improve the translation into and from Southern African languages.

With the aim of improving the translation of English into Northern Sesotho using a Transformer-

²Nyoni and Bassett (2021) describe the architecture of NMT models used in each experiment; however, it is not clear how the authors arrived at the choice for the hyperparameters. This excludes the hyperparameters for the learning rate schedule and optimization algorithm, which they base on the work of Vaswani et al. (2023).

³The additional corpora contain translations from English-to-isiXhosa, English-to-Shona and isiXhosa-to-isiZulu. Similarly to the English-to-isiZulu corpus. The corpora were sourced from Opus Corpus (2019), Omniglot Encyclopedia (2020), Linguanaut Phrases Center (2019), Wild Coast Xhosa Phrase Book (2019). IsiXhosa, Shona, isiZulu are classified as *Zone S*. Bantu languages, within the Niger-Congo language family, and isiZulu and isiXhosa are further classified as Nguni languages with the *Zone S*. classification (Guthrie and Institute, 1948; Maho, 2009; Eberhard and Fennig, 2024).

⁴Elmadani et al. (2022) implement the multilingual NMT model defined by Liu et al. (2020), using the hyperparameter configuration prescribed by Tang et al. (2021).

⁵The eight Southern African languages include Afrikaans, Northern Sesotho, Shona, Siswati, Setswana, isiXhosa, Xitsonga and isiZulu. The corpora were sourced from Meta’s No Languages Left Behind Project (Team et al., 2022).

⁶The multilingual model is trained to translate between English and the eight Southern African languages, as well as between specific pairs of Southern African languages.

based NMT model (Vaswani et al., 2023), Mojapelo et al. (2023) investigate the impact of two data augmentation methods whose function is to increase the size and diversity of an English-to-Northern Sesotho training parallel corpus^{7, 8}. The first method being investigated uses back-translation (Sennrich et al., 2016a) to generate additional training examples. To be more specific, the method uses Northern Sesotho monolingual corpora that have been sourced from various domains to generate synthetic sentences in English. The second method is a word-replacement method that initially trains a supervised Generative Adversarial Network (GAN) to learn a mapping between words in Northern Sesotho and Setswana, for the purpose of developing a bilingual dictionary between the two languages⁹. Using the bilingual dictionary, the word replacement method replaces Setswana words in an English-to-Setswana parallel corpus with their corresponding translation in Northern Sesotho.

NMT models are data-hungry. However, the works of Nyoni and Bassett (2021), Elmadani et al. (2022) and Mojapelo et al. (2023) show that leveraging additional corpora to perform established best practices for NMT in low-resource data settings is an effective approach to improving the translation performance for South African languages. Even so, the effectiveness of these approaches, as evidenced in the above works, is dependent on the availability of suitable additional corpora, that is the additional corpora needs to be large enough, contain a language closely related to and/or share a common domain with the low-resource language (Sennrich and Zhang, 2019).

Sennrich and Zhang (2019) and Araabi and Monz (2020) investigate the impact of modifying the vocabulary representation, architecture and regularization settings of NMT models trained specifically in low-resource data settings. Sennrich and Zhang (2019) train their baseline NMT model using an RNN Encoder-Decoder architecture (Bahdanau et al., 2016), developed in the Nematus toolkit (Sennrich, Firat, Cho, Birch, Haddow, HITSCHLER, Junczys-Dowmunt, Läubli, Miceli Barone, Mokry and Nädejde, 2017), for the translation of German into English, with a training parallel corpus containing 159,000 sentence pairs. Araabi and Monz (2020) on the other hand, train their baseline models using a Transformer-based NMT model (Vaswani et al., 2023), developed in the Fairseq toolkit (Ott et al., 2019), leveraging subsets of a German-to-English parallel corpus for training. The size of the subsets range from 5,000 to 165,000 sentence pairs. In both cases, the search for an optimised NMT model is performed by incrementally applying a single or a number of modifications to the model’s hyperparameter settings. The outcome of both investigations show that one can achieve significant improvements to the model’s translation performance by carefully modifying the vocabulary representation, architecture and regularization settings of the model for low-resource language pairs, when compared to phrase-based SBMT models and NMT models whose settings have been optimised for high-resource language pairs.

2.2.2 The Role of Model Adaptation

Following from the work of Sennrich and Zhang (2019) and Araabi and Monz (2020), the objective of this thesis is to investigate whether NMT models trained on South African languages can benefit from modifications made to how sentences are represented and processed in the model. In order for an NMT model to process a sentence, the sentence needs to be represented as a sequence of smaller units, called tokens. A vocabulary is a list of unique tokens in a training corpus that can be sequenced together to represent a sentence. The process of generating these tokens is called tokenization. Tokens can be characterised as words, subwords, or characters. How tokens are characterised impacts how an NMT model learns from the text in a corpus.

Early NMT approaches often defined tokens as the different words in a sentence (Kalchbrenner and Blunsom, 2013; Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014; Sutskever et al., 2014). The vocabulary in this instance is limited to a fixed number of frequently observed words in the training corpus. This becomes an issue when the model is required to translate rare and unseen words. Early attempts to address this issue involved relying on external dictionaries to translate out-of-vocabulary (OOV) words, or simply copying those words to the translation (Jean et al., 2015; Luong, Sutskever, Le, Vinyals and Zaremba,

⁷Mojapelo et al. (2023) implement the hyperparameter settings for a Transformer-based NMT model (Vaswani et al., 2023) as prescribed in Martinus and Abbott (2019a), with some hyperparameter tuning with respect to the model’s architecture (increasing the number of attention heads), and regularization (decreasing the dropout rate).

⁸All parallel and monolingual corpora used in this study were sourced from Opus Corpus (Tiedemann, 2012), Autshumato Project Machine Translation Evaluation Set (McKellar and Puttkammer, 2020), FLoRes-101 Evaluation Benchmark (Goyal et al., 2022), NCHLT Sepedi Text Corpora (Eiselen and Puttkammer, 2014), and web crawling.

⁹Northern and Sesotho and Setswana are both classified as Sotho-Tswana languages within the *Zone S*. Bantu language classification (Guthrie and Institute, 1948; Maho, 2009).

2015). The disadvantages of the first approach is that (i.) it is dependent on the availability of a digitised bilingual dictionary; (ii.) it is memory-intensive, as it requires that a large dictionary be stored alongside the NMT model; and (iii.) it assumes that the mapping of words in a source and target language is one-to-one (Sennrich et al., 2016b). Copying OOV is a reasonable approach when dealing with proper nouns in a sentence; however, it can prove to be inefficient when morphological and transliteration considerations need to be made (Sennrich et al., 2016b). For example, if we consider the name of an annual event in South Africa called the "State of Nation Address" in English, it is translated into "UkweThulwa kweKulumo kaMongameli yesiTjhaba" in isiNdebele. The morphological differences between the names of the event in English and isiNdebele make the copying of the name from one language into the other infeasible.

To address the limitations of word-based tokenization, research into developing representations of words and sentences which can account for the characters and morphemes used to form them for NMT were explored. Chung et al. (2016) apply character-based tokenization by representing entire sentences as a sequence of characters. Sennrich et al. (2016b) introduce subword tokenization as a way to learn representations of words in a sentence that account for the morphemes used to form them. A subword is a segment of a word: it can be as short as a character, or as long as whole word. The main goal of subword tokenization is to construct a vocabulary that comprises frequently observed words in the training corpus, together with an optimal set of word segments that can be sequenced together to form OOV words (Sennrich et al., 2016b). When compared to the tokens generated using word-based tokenization, subword and character-based tokens can better capture how words are formed. This in turn increases the capacity of the NMT model (i.) to learn the relationship between similar words, by identifying words with shared sequences of characters, and (ii.) to translate OOV words, by translating the distinct elements of the word (Chung et al., 2016; Sennrich et al., 2016b). The vocabularies generated using subword and character-based tokenization algorithms are also relatively smaller, which means that they require less memory to store vocabularies (Sennrich et al., 2016b). Lastly, the embedding representations that the NMT model generates of subword and character-based tokens are able to store a more comprehensive set of features of a word or sentence, since the representations encode information from a less information-dense token¹⁰. When compared to tokens generated using character-based tokens, subword tokens are generally more meaningful, that is that they define boundaries in words and sentences which help the model perform a more efficient mapping between elements in the source and target sentence. Further, sentences represented as a sequence of subword tokens are generally shorter, allowing for more efficient storage of sentence representations and decoding (Sennrich et al., 2016b). Subword tokenization is a typical choice for tokenization in NMT modelling because of its ability to achieve an effective trade-off between strengths and weaknesses of word- and character-based tokenization. It is also the preferred method for tokenization when translating from English into South African languages because of the differences in word formation processes and writing systems used to create and represent meaning between these languages. These differences mean that the mapping of words from one language to another is not always one-to-one¹¹.

To understand the contribution made by this thesis, we consider prior work that has been done with respect to vocabulary representation, architecture, and training strategies of Transformer-based NMT models (Vaswani et al., 2023) for the translation of South African languages. Martinus and Abbott (2019b) present benchmark scores for the translation of English into Setswana, Northern Sesotho, Xitsonga, isiZulu and Afrikaans using Transformers that were trained on parallel corpora sourced from the Autshumato Project (Groenewald and Fourie, 2009). Each model adheres to the hyperparameter configuration prescribed in the Tensor2Tensor toolkit (Vaswani et al., 2018), and the corpora were prepared for translation using the WordPiece subword tokenization algorithm (Schuster and Nakajima, 2012; Wu et al., 2016). WordPiece starts off with an initial vocabulary of all unique characters in the training corpus, it then iteratively merges symbols in the vocabulary; however, a merge is only retained in the vocabulary if it maximises a language model defined over the training corpus. A symbol can be a single character, or a number of consecutive characters in the corpus that have been merged in previous iterations of the algorithm. The iterative process is repeated until a pre-defined number of tokens for the vocabulary is reached.

Martinus et al. (2020) also present benchmark scores for the translation of English into the remaining ten South African official languages using Transformers trained on JW300 parallel corpora (Agić and Vulić, 2019). The hyperparameter configuration for each model is defined in the publication;

¹⁰Assuming that the dimension of the embedding representation stays the same in the NMT model.

¹¹In Chapter 3, we outline the differences in the linguistic properties in more detail.

however, it is not clear how this specific configuration was reached. The models are trained using the JoeyNMT toolkit (Kreutzer et al., 2019), and the corpora are prepared for translation using the BPE subword tokenization algorithm (Sennrich et al., 2016b).

van Biljon et al. (2020) investigate the optimal depth of a Transformer for the translation of English into Setswana, Northern Sesotho and Afrikaans using parallel corpora from the Autshumato Project (Groenewald and Fourie, 2009). Applying a hyperparameter configuration similar to Martinus and Abbott (2019b), van Biljon et al. (2020) use the JoeyNMT toolkit (Kreutzer et al., 2019) to adjust the number of encoder and decoder layers in the Transformer. The results from the study indicate that the medium-depth Transformer (with 3 encoder and decoder layers), outperforms the benchmark scores of the baseline Transformer (with 6 encoder and decoder layers) in Martinus and Abbott (2019b) on two out of the three language pairs. In the case of the translation from English into Northern Sesotho, the medium-depth Transformer performs significantly worse than the benchmark score obtained in Martinus and Abbott (2019b). However, van Biljon et al. (2020) were unable to reproduce the benchmark obtained in Martinus and Abbott (2019b). In fact, attempts to reproduce the benchmark yielded models whose translation performance was similar to that of the medium-depth Transformer.

Rajab (2022) compares two implementations of the BPE tokenization algorithm (Sennrich et al., 2016b) from the subword-nmt toolkit (Sennrich et al., 2016b) and the SentencePiece toolkit (Kudo and Richardson, 2018), on parallel corpora from the Autshumato Project (Groenewald and Fourie, 2009), for the translation of English into Setswana, Northern Sesotho, Xitsonga and isiZulu. Rajab (2022) uses an optimised medium-depth Transformer (van Biljon et al., 2020) for translation¹². The subword-nmt toolkit is the official implementation of the BPE algorithm as described in (Sennrich et al., 2016b). However, the implementation in the SentencePiece toolkit differs from the official implementation, in that it does not require that the words in a sentence be explicitly segmented into a sequence of words, prior to applying the BPE merge operations. The two implementations are evaluated over different vocabulary sizes. The outcome of the publication indicates an improvement for the translation of English into Setswana and isiZulu using the SentencePiece implementation over van Biljon et al. (2020) who apply BPE using the subword-nmt toolkit, and Martinus and Abbott (2019b) who apply WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016) for subword tokenization.

Meyer and Buys (2023) introduce a Subword Segmental Machine Translation (SSMT) model, which combines the tokenization of target sentences with the architecture of an NMT model, for the translation of English to isiXhosa, isiZulu, Siswati, Setswana and Afrikaans. The parallel corpora used in this publication were sourced from `wmt22_african`. SSMT is an adaptation of Dynamic Programming Encoding (DPE) (He et al., 2020). The goal of DPE is to find a token representation of a target sentence that maximises the probability of observing the target sentence given its corresponding source sentence in the corpus. It is able to outperform BPE (Sennrich et al., 2016b) and BPE-Dropout Provilkov et al. (2020) on translation tasks. The improvement in translation performance is attributed to how target sentences are tokenized in DPE: the artificial neural network used for the target language tokenizer conditions the segmentation of each word in the target sentence on a context, and this context accounts for the corresponding source sentence and previous characters in the target sentence (He et al., 2020). Further, the architecture for the artificial neural network resembles an NMT model. The training of the tokenizer, however, takes place separately from the training of the NMT model used for translation. Therefore, the subword sequence representations generated of target sentences are not optimised specifically for the task that they are being deployed for Meyer and Buys (2023). This is what the SSMT model aims to address by combining the tokenization of target sentences and NMT into a single model. A significant improvement of SSMT over BPE, ULM (Kudo, 2018) and DPE was observed for the translation of English into Siswati. That said, this improvement was achieved at the expense of significantly increasing the computational complexity of the NMT model: an SSMT model trains for approximately ten times longer than an NMT model whose sentences are tokenized prior using BPE, ULM or DPE (Meyer and Buys, 2023).

¹²van Biljon et al. (2020) release an optimised medium-depth Transformer for low-resource languages, with respect to the hyperparameter settings, as part of the outputs produced by the publication.

2.3 Conclusion

Following from this chapter, we are able to trace the background of early MT to the approach frequently used today: NMT. Despite the fast, growing popularity of NMT, its application in low-resource data settings has not been as seamless. In this chapter, we unpacked methods that have been applied to improve the performance of these models in low-resource data setting, and more specifically within the context of South African languages. These methods can be grouped into the following two approaches: (i.) the use of additional corpora, and (ii.) modifying the vocabulary representation, architecture and regularization settings of NMT models to better suit low-resource data settings. While the methods grouped in first approach have been very successful, this thesis is interested in contributing to the methods in the second approach for the purposes of understanding the impact different tokenization strategies have on translation into South African languages, and prioritising solutions for NMT into South African languages that promote computational efficiency, in the face of limited time and computational resources. In the next chapter, we examine the datasets used to train, validate and test all the NMT models deployed in this thesis.

Chapter 3

Data

This chapter examines the datasets that are used to train, validate and test all NMT models that are deployed in this thesis. It begins with a description of the datasets, and where they were sourced from. This is then followed by an outline of the potential issues observed in the different datasets, and the steps taken to preprocess them to minimise the issues in preparation for tokenization and modelling.

3.1 Autshumato Project for Machine Translation

This thesis makes use of multiple parallel corpora that have been curated by the Centre for Text Technology (CTeT)¹, as part of the Autshumato Project (Groenewald and Fourie, 2009). The Autshumato Project was originally commissioned by the South African Department of Sport, Arts and Culture to develop NLP tools to assist government translators with translating government documentation into all 11 official languages. This was done for the ultimate goal of further propelling the government’s mission to remove barriers preventing ordinary South Africans from engaging with the operations of their government in their home language, and fostering a multi-lingual society where all 11 official languages in the country hold an equal status (Groenewald and Fourie, 2009).

Since its formation, the Autshumato Project has grown in terms of who it aims to serve and the output generated². In its current iteration, enabled by the funding of the South African Centre for Digital Language Resources (SADiLaR)³, it also promotes open-source innovation for the translation of South African languages. As a result, the project has made a number of parallel corpora publicly available on the SADiLaR resource repository for the training of statistical- and neural network-based MT models⁴. The parallel corpora contain translations from English into isiNdebele, Northern Sesotho, Setswana, Tshivenda, Xitsonga, isiZulu and Afrikaans.

3.1.1 Classification of South African Languages

Guthrie and Institute (1948) propose a formal classification for Bantu languages spoken in Africa based on geographic zones. These languages are largely spoken in regions south of the Equator. Within each geographic zone, the languages are classified into language groups⁵; within each language group, languages are assigned an index number, with some languages being further sub-indexed according to their various dialects. The geographic zones are not entirely based on a genealogical relationship between the language groups classified in them (Guthrie and Institute, 1948; Janson, 1991-1992). However, it is possible for language groups within the same geographic

¹A research centre based at the North-West University (in South Africa) that specialises in NLP research for South African languages.

²See [Autshumato MT Web Service](#).

³A national centre within the South African Department of Science and Innovation, dedicated to supporting NLP research, and the development of tools and language-related studies for South African languages.

⁴Other resources have also been made available via the Autshumato Project, including monolingual corpora, wordlists, translation memories, and other tools and applications to support NLP research for South African languages.

⁵Language groups are also referred to as *decades*.

zone to share a number of linguistic processes. This may be due to the amount of time speakers of these languages have spent living alongside one another in a geographic zone (Janson, 1991-1992).

Updates to the Guthrie and Institute (1948) classification of Bantu languages have since been made in the “New Updated Guthrie List” (Maho, 2009), using relevant publications and credible sources at the time. Some of these updates include the addition of languages to the list, a comprehensive revision of how languages are classified, and the standardisation of language-naming conventions, etc. Figure 3.1, sourced from I, Edricson (2007) under the CC BY-SA 3.0 license, displays the configuration of the geographic zones used to classify Bantu languages on the map. South African official languages, excluding English and Afrikaans, are all classified as *Zone S*. Bantu languages.

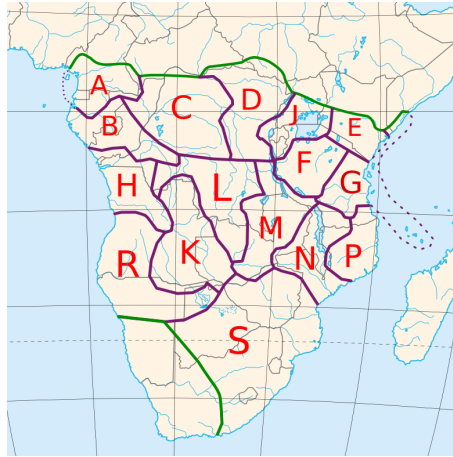


Figure 3.1: Configuration of the geographic zones in Africa used to classify Bantu languages (Edricson, 2007).

Within the geographic *Zone S* classification, languages are classified into the Copi, Nguni, Shona, Sotho-Tswana, Tswa-Rhonga and Venda language groups, spoken in parts of Botswana, Eswatini, Lesotho, Mozambique, Namibia, South Africa, Zambia and Zimbabwe. South African official languages belong to the Nguni, Sotho-Tswana, Tswa-Rhonga and Venda language groups. Table 3.1 outlines which language groups, within the *Zone S* classification, South African official languages are classified into.

Language Groups				
	Nguni	Sotho-Tswana	Tswa-Rhonga	Venda
Languages	isiNdebele	Northern Sesotho	Xitsonga	Tshivenda
	Siswati	Sesotho		
	isiXhosa	Setswana		
	isiZulu			

Table 3.1: How South African official spoken languages are further grouped within the *Zone S* language classification.

The *Zone S* classification is one of the few geographic zone classifications, presented by Guthrie and Institute (1948) and Maho (2009), whose language groups share a genealogical relationship. This is corroborated by Janson (1991-1992), who argues that there exists a *proto*-Bantu language that the Copi, Nguni, Sotho-Tswana, Tswa-Rhonga, Venda and Emakhuwa⁶ language groups all descend from. A *proto*-language is a hypothetical language which more recent languages all descend from. The descendants of a *proto*-language all share a number of fundamental linguistic processes. Janson (1991-1992) classifies these six Bantu language groups as Southern Bantu languages.

The Shona language group is excluded from this classification because of the differences observed in its morphological and phonological processes relative to the Southern Bantu language groups, except for the Venda (Janson, 1991-1992). Languages in the Shona and Venda language groups

⁶According to Guthrie and Institute (1948) and Maho (2009), the Emakhuwa language group is assigned to the *Zone P* classification.

share a number of linguistic processes. Janson (1991-1992), however, attributes these similarities to how closely speakers of these languages have spent living alongside one another.

According to Eberhard and Fennig (2024), the geographic zones presented by Guthrie and Institute (1948) and Maho (2009) all fall under the Narrow Bantu classification within the larger Niger-Congo language family. Under the same language framework, Afrikaans and English are classified as West Germanic languages within the larger Indo-European language family, with Afrikaans belonging to a sub-family of West Germanic languages, called Low Franconian.

3.1.2 Linguistic Properties of South African Languages

The formal classification of languages offers a system to identify patterns, and formulate generalisations about how languages are structured. Spaul et al. (2020) identify key linguistic differences between the four Southern Bantu language groups that contain South African official languages. The key differences that have been identified, relevant to this thesis, include a typological feature that categorises languages based on the word formation process used to append morphemes to stem or root words, and an orthographic feature that categorises languages based on the writing system used to represent the grammatical components of a language.

The typological feature comprises three categories, namely isolating, inflectional and agglutinating languages. These categories are not absolute. Instead, a language is categorised based on its morphological density relative to other languages (Spaul et al., 2020). On the one end of the scale are agglutinating languages whose morphological processes result in the formation of more complex words; and on the other end are isolating languages whose morphological processes are characterised by lower density words (Spaul et al., 2020).

When compared to languages typically categorised as inflectional, English predominantly exhibits features of an isolating language. Languages classified into the Southern Bantu language groups are examples of agglutinating languages. Grammatical meaning in an agglutinating language is conveyed through appending prefixes, infixes and suffixes before and after noun and verb root words (Spaul et al., 2020). Compared to isolating languages, the verbal components in agglutinating languages are especially more complex, because they also encode information indicating the subject and object noun class (Spaul et al., 2020). In the case of English, while it may use a few inflectional word formation processes, the language, like other isolating languages, largely depends on word order to convey grammatical meaning in a sentence (Krzysztof, 2015).

The processes used to convey grammatical meaning in a sentence differ slightly between the Southern Bantu language groups. However, when compared to isolating languages, the resulting words formed in agglutinating languages are generally highly dense, and in some cases are longer written words (Spaul et al., 2020). An example of this is given in Table 3.2. In this example, “thanda” in isiZulu (the agglutinating language) can be directly translated to mean “love” in English (the isolating language). In both sentences “thanda” is the root word. For the first sentence, the morphemes “Ngi-”, “-ya-” and “-ku-”, are appended to the root word to generate a sentence that means “I love you” in English. In the second sentence, a different set of morphemes is appended to the root word which alter the grammatical meaning of the first sentence. The new sentence is directly translated to mean “You are loved”. In the case of the English sentences, unlike in isiZulu, grammatical meaning is altered mostly via the re-arranging of lower density words in the sentence⁷.

	Language Pairs	
	IsiZulu	English
Sentence 1	Ngiyakuthanda.	I love you.
Sentence 2	Uyathandwa.	You are loved.

Table 3.2: Example of how grammatical meaning is conveyed in an agglutinating language compared to an isolating language.

The orthographic feature comprises two categories, namely conjunctive and disjunctive. Similar to the typological feature described above, the categories of this feature are not absolute. The degree to which a language follows a conjunctive or disjunctive writing system is measured relative to

⁷It is worth noting that the second sentence also uses an inflectional word formation process, that is, “love” in the first sentences becomes “lov[e]” + “-ed” in the second sentence

other languages (Prinsloo and de Schryver, 2002). Languages with a conjunctive writing system generally combine grammatical components into a single written word. Whereas, languages with a disjunctive writing system tend to split a single grammatical component into a number of written words (Spaull et al., 2020).

Nguni languages predominately use a conjunctive writing system, and Sotho-Tswana languages adhere to a disjunctive writing system. Spaull et al. (2020) state that Xitsonga and Tshivenda from the Tswa-Rhonga and Venda language groups, respectively, display features associated with conjunctive and disjunctive orthographies.

Prinsloo and de Schryver (2002) present a tool to measure how conjunctive (or disjunctive) the writing system of each South African official language is relative to the writing systems of every other South African official language. Their findings suggest that the degree to which the writing systems of Xitsonga and Tshivenda are disjunctive is indistinguishable from the disjunctive writing systems of the Sotho-Tswana languages. In the case of English and Afrikaans, the degree to which these languages’ writing systems are conjunctive (disjunctive) falls in-between the conjunctive and the disjunctive writing systems. An example of how the different writing systems can take effect is given in Table 3.3, using a sentence written in isiNdebele (conjunctive), Northern Sesotho (disjunctive), Xitsonga (mostly disjunctive) and English (somewhere in-between). In this example, taken from Prinsloo and de Schryver (2002), we see that the sentence written in a language with a conjunctive writing system is characterised by its shorter length, where each word in it is generally much longer. Whereas, the sentence written in a language with a disjunctive writing system is much longer, containing the largest number of one-syllable words in-between more dense words.

Language	Sentence
isiNdebele	Ngo-1994 umphakathi wephasi loke wabona ukuzalwa kombuso omutjha wentando yesitjhaba wamukela iSewula Afrika njengelungu elitjha lomphakathi wesphasi loke. Ekwamukeleni irhahla lelo, inarha lena yajamelana nokutjhijilelwa okutholakala ebujamweni obunjalo. (28 words)
English	In 1994 the international community witnessed the birth of a new democracy and welcomed the new South Africa as the most recent member of its global village. In accepting that honour, this country took on the associated challenges of that position. (41 words)
Xitsonga	Hi 1994, vaaki va misava hinkwayo va vonile ku tswariwa ka demokhi-rasi leyintshwa ni ku amukela Afrika-Dzonga leyintshwa tani hi xirho lexintshwa emutini wa matiko hinkwawo. Hi ku amukela ku fundzhiwa loku, tiko leri ri tekile mintlhontlho leyi fambelanaka ni xiyimo lexi. (42 words)
Northern Sesotho	Ka ngwaga wa 1994 setšhaba sa boditšhabatšhaba se bone pele go ya temokrasi ye mpsha gape sa amogela Afrika Borwa ye Mpsha bjalo ka leloko le lefsa kudu motseng wa sona wa lefase ka bophara. Ge go amogelwa tlotlo yeo, naga ye e tšere dihlotlo tšeo di amanago le maemo ao. (50 words)

Table 3.3: Example of how the different writing systems in isiNdebele, English, Xitsonga and Northern Sesotho compare to one another.

3.2 Description of the Parallel Corpora

Given time and resource constraints, this thesis will only use three of the parallel corpora from the Autshumato Project. The selected corpora contain translations from English into isiNdebele, Northern Sesotho and Xitsonga. The parallel corpora were selected based on the following key features:

1. All three corpora contain language pairs whose source and target languages belong to distant language families, that is, Indo-European (English) versus Niger-Congo (isiNdebele, Northern Sesotho and Xitsonga); and

2. All three target languages in the corpora represent different writing systems observed across South African official languages, that is, conjunctive (isiNdebele), disjunctive (Northern Sesotho), and predominantly disjunctive (Xitsonga).

The selected corpora contain translations of government documentation sourced from the South African government domain (*.gov.za). The documentation includes sections of the constitution, press releases, speeches, and information about various governmental departments and the services that they provide (McKellar and Puttkammer, 2020). Each corpus comprises source-target segment pairs. A segment is a complete sentence, a part of a sentence or even a single word. According to the accompanying meta data file for each corpus, the process of matching source-target segment pairs was done using human translators and an algorithm-based alignment technique⁸.

The effectiveness of an NMT model hinges on the size and quality of the corpora used to train and evaluate the model. The size of the corpora selected for this thesis is small. Therefore, preparing high-quality corpora for modelling is paramount. A high-quality corpus contains source and target segments that are correctly aligned. The source segment is required to convey a thought or an idea in the most natural tone to the source language, and the target segment is an accurate translation generated by a professional translator. An accurate translation not only captures the meaning of the source segment using correct grammar, it also attempts to correctly capture the nuances in the source segment with respect to style and tone. In general, it is preferable for a high-quality corpus to contain segment pairs that are domain-specific, so that the nuances of that domain are better captured in the NMT model (Xu and Koehn, 2017). However, this is at the expense of building a translation model that does not always generalise well to out-of-domain contexts (Martinus et al., 2020).

3.2.1 Potential Issues Observed in the Parallel Corpora

The parallel corpora obtained from the Autshumato Project contain a number of issues that negatively affect the quality of the corpora, and in turn the translation quality of NMT models developed using them. One such issue is the occurrence of misaligned source-target segment pairs. Some source segments are aligned to target segments that contain incomplete translations, or over-translations that include details that are not in the source segment. In other cases, these source segments are left untranslated, meaning that a source segment is aligned to an empty segment on the target side⁹, the source segment is simply copied to the target side, or the target segment provided is completely irrelevant to the corresponding source segment.

Other cases of misaligned source-target segment pairs include a single source segment being aligned to multiple target segments, or multiple source segments being aligned to a single target segment. While such segment pairs could contain valid translations¹⁰, they also introduce ambiguity that an NMT model may have trouble processing during training (Rikters, 2018). From inspecting the corpora that were selected for this thesis, many of these one-to-many and many-to-one source-to-target segment pairs simply contained inaccurate translations.

After splitting the corpora for training, validation and testing: duplicate source-target segment pairs were found in the training and validation corpora. These duplicates can affect how the NMT model learns, caused by the model unintentionally assigning greater importance to duplicate examples during training. Duplicate source segments from the test corpora were also found in the training and validation parallel corpora. The occurrence of these duplicates; however, only affects the validity of the final evaluation of the NMT models that are deployed after training.

When considering the length of segments on either side of a corpus, segments that are too long or short can pose a challenge to an NMT model. The selected corpora for this thesis only contain outlying segments, with respect to length, that are too long. The challenge with training an NMT model with very long segments is that these segments require a model with a much larger parameter space. This is an issue in low-resource data settings. The increased parameter space

⁸An alignment technique could be as simple as using a bilingual dictionary and/or length-based information to match segment pairs Halácsy et al. (2007); or as complex as using a neural network-based model to generate vector representations of segments in the different languages, and searching for common patterns to match segment pairs (Thompson and Koehn, 2019). It remains unclear which alignment technique was used for each parallel corpus based on Groenewald and Fourie (2009), and the accompanying meta data file for each parallel corpus.

⁹and vice versa.

¹⁰that is, paraphrases of a translation linked to a single source segment, or paraphrases of the source segment linked to a single translation.

not only increases the risk of the model over-fitting the training corpus; it also increases the risk of unstable training if segments are highly variable¹¹, causing the model not to converge to an optimal set of parameters during training.

The remaining issues that have been observed in the corpora stem from how the text in each corpus is represented, more specifically the complexity and lack of consistency in how the text is represented. One such issue is the occurrence of non-printable characters¹² in the corpora. The occurrence of such characters increases the complexity of the text, without contributing any relevant information for translation during model training.

Another related issue is the inconsistent use of punctuation marks and letter casing in the corpora. This introduces noise, which in turn increases the complexity of the text. These inconsistencies also compromise the NMT model's ability to learn meaning in each language. Meaning that is captured in how punctuation marks and letter casing are used in a language. Examples of the lack of standardisation of punctuation marks and letter casing usage in the parallel corpora can be seen in Table 3.5 and Table 3.4, respectively. Note that in Table 3.4, the acronym "SONA" refers to the State of Nation Address. On the isiNdebele side, the acronym is spelled out as "UkweThulwa kweKulumo kaMongameli yesiTjhaba".

Language Pairs		
	English	isiNdebele
Sentence 1	THERE ARE A NUMBER OF PROCESSES THAT ALL MUNICIPALITIES are required to perform.	KUNEENKAMBISO EZIMBALWA EKULINDELEKE KOBANA BOKE ABOMASIPALA bazenze.
Sentence 2	Appendix 2 : Glossary of terms	ISITHASISELO 2 : Idlhozari yamathemu
Sentence 3	IF THE SERVICE IS BAD , GOVERNMENT IS BAD AND IF THE SERVICE IS GOOD , GOVERNMENT IS GOOD , IT IS AS SIMPLE AS THAT .	NANGABE UMSEBENZI MUMBI , URHULUMENDE MUMBI BEGODU NANGABE UMSEBENZI MUHLE , URHULUMENDE MUHLE , KUNJENGALOKHO . LOKHU KULIQINISO IPHASI LOKE ALIKAHLUKI KUNSEWULA AFRIKA .
Sentence 4	The SONA is broken down into three parts .	2 . UkweThulwa kweKulumo kaMongameli yesiTjhaba kuhlukaniswe iingaba ezintathu .
Sentence 5	* continuing work to address especially the various non-natural causes of death in our society as well as lifestyle diseases , malaria , the various strains of TB , road accidents and violent crime .	* Ukuragela phambili komsebenzi wokutjheja khulukhulu abonobangela abahlukeneko bokuhlongakala abangasi ngebemvelo emphakathinethu begodu namalwele wendlela esiphila ngayo , imalariya , ubuhlungu obuhlukeneko be-TB , iingozi zendlela nobulelesi bokulwa .

Table 3.4: Examples of inconsistent usage of letter casing in the English-to-isiNdebele parallel corpus.

3.2.2 Preprocessing the Parallel Corpora

Typically, the task of preprocessing a parallel corpus for NMT involves splitting the corpus into training, validation and test corpora, followed by the application of data cleaning and tokenization to each corpus split. The remainder of this chapter will discuss the data splitting and data cleaning methods used in this thesis. Since the objective of this thesis is to investigate the impact of different tokenization strategies, descriptions of the strategies are provided in Chapter 4 (Methods).

¹¹that is, with respect to the lengths of the segments.

¹²Non-printable characters provide a computer with information of the data structure of text, how said text should be formatted on a page or monitor, or how the text is to be transmitted to other devices.

The data cleaning methods applied to corpora from the Autshumato Project for NMT have largely involved removing duplicates of segments. In [Martinus and Abbott \(2019b\)](#), all segment pairs whose source segment is duplicated multiple times in said corpus are removed¹³, before splitting the corpus into training, validation and test corpora to minimise data leakage between the corpora splits.

Using the training corpus from [Martinus and Abbott \(2019b\)](#) for each language pair, [van Biljon et al. \(2020\)](#) clean the corpus by removing¹⁴ all duplicate segment pairs, as well as segment pairs whose source or target segment is duplicated multiple times in the corpus. Using the test corpus split from [Martinus and Abbott \(2019b\)](#), [van Biljon et al. \(2020\)](#) remove segment pairs from the training corpus whose source segments are a duplicate of or are "very similar" to the source segments in the test corpus. The similarity between two segments is measured using the Levenshtein distance ([Levenshtein, 1965](#)), that is a metric that computes the number of insertions, deletions and/or substitutions required to transform one segment to the other ([Kessler, 1995](#)). The segment pairs in the training corpus are then shuffled, to avoid any potential bias, and split to return a training and validation corpus. [Rajab \(2022\)](#) adopts the preprocessing strategy outlined in [van Biljon et al. \(2020\)](#).

The steps taken to preprocess each corpus in this thesis are similar to the earlier works described above. However, a number of additional steps have also been included to minimise the impact of the issues outlined in Subsection 3.2.1 on the quality of each corpus. The steps taken can be grouped into the following three stages: **data splitting**, **text normalization**, and **filtering**. An overview of the corpora before and after preprocessing is presented in Table 3.6. Note that the total number of tokens for both the source and target languages refers to the total number of whitespace characters, which include words, punctuation marks and numbers, on each side of the parallel corpus after preprocessing.

Data Splitting

For each language pair, the parallel corpus obtained from the Autshumato Project ([Groenewald and Fourie, 2009](#)) is split into a training, validation and test corpus. This process involves shuffling segment pairs in the original corpus to minimise any bias, then sampling 3000 segment pairs from the corpus to assign to a test corpus, sampling a further 20% of the remaining segment pairs to assign to a validation corpus, and assigning what remains to a training corpus.

In addition to the original corpora provided as part of the Autshumato Project, [McKellar and Puttkammer \(2020\)](#) publish additional corpora for the evaluation of NMT models. The additional test corpus for each language pair contains 500 source segments that were also sourced from government documentation available on the South African government domain (*.gov.za). However, each source segment corresponds to four translations in the target language that were generated by four professional translators.

Before moving on to text normalization, a search is carried out to locate and remove segment pairs in the training and validation corpora, whose source segments are duplicates of the source segments in the recently split test corpus and the second test corpus provided by [McKellar and Puttkammer \(2020\)](#). In doing so, we eliminate any data leakage that could hamper the evaluation of the NMT models on unseen data.

Text Normalization

The purpose of this stage is to refine and improve the consistency of the text in a parallel corpus to simplify learning during model training. Text normalization is applied to the training and validation corpora, and entails the following steps:

1. Ensuring that all characters in the corpus have been properly encoded as UTF-8 characters, to avoid any loss of information that may occur when reading segment pairs from the raw text files. Fortunately, the corpora obtained contained no invalid UTF-8 characters; therefore, no further processing was required in this regard.
2. Searching for and removing all non-printable characters from the corpus.

¹³Except for the first occurrence of the duplicate in the corpus.

¹⁴Except for the first occurrence of the duplicate in the corpus.

3. Normalizing the punctuation marks in the corpus using the Sacremoses toolkit that is available in the following GitHub repository: [sacremoses](#). The toolkit reduces the types of punctuation marks used in a corpus to a smaller set of commonly used punctuation marks, and replaces the existing punctuation marks in the corpus with an appropriate alternative from the smaller set. An example of how punctuation normalization is performed is given in Table 3.5¹⁵. Prior to applying the toolkit, the following steps are taken to prepare the corpus:
 - 3.1. An em dash (—) is used in the English-to-Northern Sesotho corpus that is not recognised in the Sacremoses toolkit. We replace the occurrence of that dash with a hyphen (-).
 - 3.2. Hyphens are sometimes used as em dashes in the corpora. When such a hyphen is identified, we add a space before and after it.
4. Manually defining the method used to normalize letter casing in each corpus, in the absence of pre-trained isiNdebele, Northern Sesotho and Xitsonga truecasing models, that are responsible for determining the appropriate capitalisation of words in a sentence:
 - 4.1. If an entire source/target segment is written in uppercase, then the segment pair is converted to lowercase; and
 - 4.2. If a source/target segment is partially written in uppercase, then it is ignored because a number of such cases contain acronyms or abbreviations.
5. Ensuring that there is a whitespace between words and punctuation marks¹⁶. For tokenization, words and punctuation marks are treated as separate tokens. The corpora provided did apply spacing between words and punctuation marks; however, this was not done consistently throughout each corpus.

Language Pairs		
	English	Northern Sesotho
Before	A Pan South African Language Board established by national legislation must—	Boto ya Maleme ka moka a Afrika-Borwa yeo e theilwego ka molao wa setšhaba e swanetše —
After	A Pan South African Language Board established by national legislation must -	Boto ya Maleme ka moka a Afrika-Borwa yeo e theilwego ka molao wa setšhaba e swanetše -

Table 3.5: Example of how punctuation normalization is applied to simplify inconsistent punctuation usage in the English-to-Northern Sesotho parallel corpus.

Filtering

The purpose of this stage is to filter out noisy segment pairs from each parallel corpus that are potentially harmful to the NMT model’s ability to learn the correct translations. Filtering is applied to the training and validation corpora, and it entails the following steps:

1. Filtering out segment pairs with empty entries on either the source or target side.
2. Filtering out (i.) duplicate segment pairs, (ii.) segment pairs whose source segment is identical to the corresponding target segment, and (iii.) segment pairs whose source/target segment has been duplicated multiple time in the corpus. In cases (i.) and (iii.), we filter out all segment pairs except for first occurrence of the duplicate in the corpus.
3. Measuring the length of segments on the source and target side of a corpus, that is the number of whitespace characters which include words, numbers and punctuation marks, in each segment. The training corpus is used to find the length of outlying segments on the

¹⁵In this example, punctuation normalization simplifies the usage of the different dashes (i.e., a hyphen, en dash and em dash) into a sole form (i.e., a hyphen). Hyphens and en dashes are used similarly, in that they are commonly used to append a word or a morpheme to a word. Whereas, an em dash breaks a sentence. Punctuation normalization allows us to distinguish between an em dash and the other dash punctuation marks via a whitespace between the previous word and the punctuation mark.

¹⁶An exception is made for hyphens used to join a morpheme/word to a word. This is done to distinguish between the standard hyphen usage and the hyphen being used as an em dash.

source and target side. Segment pairs whose source/target segment is an outlier with respect to length are then filtered out.

4. Computing source-target ratios of segments pairs in a corpus, using the length of each source segment and dividing it by the length of its corresponding target segment. The training corpus is used find segment pairs with outlying ratios. Segment pairs whose ratios are an outlier are then filtered out. The motivation behind this step is to try and filter out segment pairs whose source segments are aligned to incomplete, over-translations, or completely irrelevant translations on the target side.

	Languages		
	isiNdebele	N. Sesotho	Xitsonga
Total number of segment pairs before preprocessing	128 382	44 980	450 000
Total number of tokens	1 241 715	487 008	1 754 422
Total number of corresponding English tokens	1 663 670	385 409	1 496 127
Total number of unique tokens	164 887	27 565	41 029
Total number of unique corresponding English tokens	37 709	28 465	32 888
Number of segment pairs for training	89 783	20 994	149 115
Number of segment pairs for validation	23 761	6336	43 390
Number of segments pairs for testing	3 000 (+500)	3 000 (+500)	3 000 (+500)

Table 3.6: Overview of the parallel corpora used in this thesis before and after preprocessing.

3.3 Conclusion

Data plays the necessary role of clarifying a problem and guiding the decisions taken to address them in machine learning. This thesis uses parallel corpora, sourced from the Autshumato Project (Groenewald and Fourie, 2009), that contain translations from English into isiNdebele, Northern Sesotho and Xitsonga. This chapter discussed how these languages are classified and their linguistic properties. This was done to better understand the correspondence between words, and how they are represented in a sentence in isiNdebele, Northern Sesotho and Xitsonga (the target languages in the corpora), relative to English (the source language in each corpus). The size of the corpora used in this thesis are small; therefore, preparing high-quality corpora for modelling is paramount. This chapter concluded with an outline of the issues observed in the corpora that affect their quality, and the heuristic steps that would be taken to preprocess corpora to mitigate these issues, in the absence of professional translators and/or linguists to correct individual segments in the corpora.

In the following chapter, we will describe the methods that will be used to tokenize the corpora, and model the translations for each language pair, using the preprocessed corpora. However, before proceeding to Chapter 4, refer to Appendix A where an analysis is conducted to show empirically the differences that exist between the languages in a parallel corpus, as a way to further motivate the decision taken in this research to apply tokenization at a subword-level for NMT.

Chapter 4

Methods

Recall that the purpose of this thesis is to investigate whether the translation performance of NMT models trained for the translation of English into a subset of low-resource South African languages can benefit from modifications made to standard subword tokenization algorithms. This thesis will apply methods that can help minimise the learning gap of the NMT models, given that the models are trained on language pairs whose source and target languages differ significantly in terms of linguistic features; methods that are data-efficient, given how small the selected parallel corpora are; and finally, are task-specific. Instead of preparing a tokenizer that is applicable to almost any NLP task, the task-specific improvement to standard subword tokenization forces the tokenizer to account for the downstream task being carried out, as well as the corresponding source or target language, in its development.

This chapter describes the methods used to perform the investigation outlined above. It describes two standard subword tokenization algorithms used in NMT, followed by the modifications made to these algorithms to improve the translation performance of NMT models, namely subword regularization, and an NMT-specific tokenization algorithm. The chapter then concludes with a description of the architecture of the NMT model that will be used for translation.

4.1 Standard Subword Tokenization

Subword tokenization is the final step taken to preprocess a parallel corpus for NMT. The vocabulary generated from this approach to tokenization contains commonly used words in a training corpus, together with a set of subwords that can be sequenced together to reproduce rare and unseen words in the training, validation and test corpora (Sennrich et al., 2016b). Recall that a subword is a segment of a word: it can be as short as a character ranging to as long as a whole word.

The vocabularies for the source and target side of a corpus can be constructed in one of two ways. The first approach involves constructing separate vocabularies for the source and target side. In general, these separate vocabularies are more compact in size, and are more representative of the language that each vocabulary is being constructed for (Sennrich et al., 2016b). The second approach constructs a joint vocabulary for the both the source and target side. This is done to improve the consistency of how words are segmented on both sides of the corpus (Sennrich et al., 2016b). It is especially effective for proper nouns: such an approach increase's the model's capacity to correctly predict the same name in a translation during validation and testing.

Joint vocabularies, however, are vulnerable to over-representing one side of the parallel corpus over the other. This can be minimised by implementing a minimum threshold for subword tokens that the joint vocabulary is allowed to retain, that is that a subword token is only retained in the vocabulary if it has been observed on **both** sides of the training corpus (Sennrich, Birch, Currey, Hermann, Haddow, Heafield, Barone and Williams, 2017). The minimum threshold helps to ensure that the joint vocabulary constructed is representative of both sides of a corpus, and it is more compact in terms of size.

There are two commonly used subword tokenization algorithms in NMT, namely BPE (Sennrich et al., 2016b) and ULM (Kudo, 2018). This thesis will use both subword tokenization algorithms

to construct joint vocabularies for each language pair for NMT.

4.1.1 Byte-Pair Encoding

Preparing a joint vocabulary for both sides of a parallel corpus using the BPE tokenization algorithm can be broken down into two major steps: an initialisation, and an iterative step. The first step constructs an initial vocabulary that contains all unique single-character tokens in the training corpus. Each word in the training corpus is then also represented as a sequence of characters, with a special end-of-word symbol ($\langle/w\rangle$) concatenated to the final character of each word¹.

The second step merges symbols found in the training corpus, and updates the vocabulary to include unique symbols that appear most frequently in the corpus. A symbol can be a single character, or a number of consecutive characters in the corpus that were merged in previous iterations of the algorithm. For each iteration, this step performs the following sub-steps:

1. Compute a count of all consecutive symbol pairs in the training corpus. For the purpose of ensuring efficiency, apply a constraint that forces the algorithm to only consider consecutive symbols that are contained within a single word;
2. Merge the symbol pair that occurs most frequently in the training corpus, and replace all occurrences of the symbol pair in the corpus with the newly merged symbol;
3. Update the vocabulary so that it only contains all unique symbols found in the training corpus.

The iterative step is repeated until a pre-defined number of merge operations have been performed. The final vocabulary constructed from this step is then modified to also include all single-character tokens from the initial vocabulary. The inclusion of these single-character tokens ensures that all parts of any word can be encoded by a token: if not by a symbol in the vocabulary, then as a character or a sequence of characters.

Consider the following dictionary containing all unique words in a corpus alongside their frequency count: {'low $\langle/w\rangle$ ':5, 'lower $\langle/w\rangle$ ':2, 'newest $\langle/w\rangle$ ':6, 'widest $\langle/w\rangle$ ':3}. Table 4.1 presents an example of how the BPE tokenization algorithm is applied to the corpus. Recall that in the initialisation step, each word in the dictionary is represented as a sequence of characters, {'l o w $\langle/w\rangle$ ':5, 'l o w e r $\langle/w\rangle$ ':2, 'n e w e s t $\langle/w\rangle$ ':6, 'w i d e s t $\langle/w\rangle$ ':3}. The resulting vocabulary after five merge operations comprises the following subword tokens: {'est $\langle/w\rangle$ ', 'new', 'lo', ' ', 'e', 'w', 's', 't $\langle/w\rangle$ ', 'l', 'o', 'n', 'w $\langle/w\rangle$ ', 'i', 'd', 'r'}.

Iter.	Best Symbol Pair	New Subword Token Representations
1	('e', 's')	{ 'l o w $\langle/w\rangle$ ':5, 'l o w e r $\langle/w\rangle$ ':2, 'n e w e s t $\langle/w\rangle$ ':6, 'w i d e s t $\langle/w\rangle$ ':3 }
2	('es', 't $\langle/w\rangle$ ')	{ 'l o w $\langle/w\rangle$ ':5, 'l o w e r $\langle/w\rangle$ ':2, 'n e w e s t $\langle/w\rangle$ ':6, 'w i d e s t $\langle/w\rangle$ ':3 }
3	('l', 'o')	{ 'l o w $\langle/w\rangle$ ':5, 'l o w e r $\langle/w\rangle$ ':2, 'n e w e s t $\langle/w\rangle$ ':6, 'w i d e s t $\langle/w\rangle$ ':3 }
4	('n', 'e')	{ 'l o w $\langle/w\rangle$ ':5, 'l o w e r $\langle/w\rangle$ ':2, 'n e w e s t $\langle/w\rangle$ ':6, 'w i d e s t $\langle/w\rangle$ ':3 }
5	('ne', 'w')	{ 'l o w $\langle/w\rangle$ ':5, 'l o w e r $\langle/w\rangle$ ':2, 'n e w e s t $\langle/w\rangle$ ':6, 'w i d e s t $\langle/w\rangle$ ':3 }

Table 4.1: Example of how the BPE tokenization algorithm is applied to a corpus.

¹The implementation of [Sennrich et al. \(2016b\)](#)'s BPE tokenization algorithm has since been updated. According to the official implementation available in the following GitHub repository: [subword-nmt](#), the special end-of-word symbol is concatenated to the final character of each word, as opposed to it being a separate symbol in the sequence of characters used to represent a word. For example, the word 'lower' is represented as 'l o w e r $\langle/w\rangle$ ', instead of 'l o w e r $\langle/w\rangle$ '. By concatenating the end-of-word symbol to the final character of each word, tokens are encoded with their positional information, that is tokens can be distinguished as word-internals (e.g., 'l', 'o', 'w', 'e') or word-finals (e.g., 'r $\langle/w\rangle$ '). In doing so, ambiguity that can arise concerning where a token should be positioned in sequence is minimised.

How to tokenize a sentence:

Tokenizing a sentence refers to the process of representing a sentence as a sequence of tokens from a vocabulary. Under the BPE framework, a sentence is tokenized by splitting it into a sequence of words, and encoding each unique word in the sentence as a sequence of subword tokens from the vocabulary.

To encode a word as sequence subword tokens, the subword tokens in the vocabulary need to be arranged in descending order of length². The word is then encoded by iterating over the tokens in the vocabulary, starting with the longest token. At each iteration, a search is performed to find a segment in the word that matches the current token being iterated over in the vocabulary. If a segment in the word is found to match the token, then all positions in the word where that segment is found are replaced with the token.

When tokenizing sentences in the validation and test corpora, these corpora may contain symbols that have not been observed in the training corpus. In such cases, the symbols are encoded with an unknown symbol token (`<unk>`). Consider the word “smallest”. Using the vocabulary that was constructed in Table 4.1, Table 4.2 shows how this word that was not observed in the training corpus is encoded with an unknown symbol token.

Iter.	Subword Sequence Representation
1	{‘small’, ‘est</w>’}
2	{‘s’, ‘mall’, ‘est</w>’}
3	{‘s’, ‘ma’, ‘l’, ‘l’, ‘est</w>’}
4	{‘s’, ‘<unk>’, ‘l’, ‘l’, ‘est</w>’}

Table 4.2: Example of how an unseen word is encoded with an unknown symbol.

4.1.2 Unigram Language Modelling

The ULM tokenization algorithm (Kudo, 2018) uses a unigram language model to estimate the probability distribution underlying how a sentence, X , is tokenized using subword tokens from a learned vocabulary, $\mathcal{V}$. Let $\mathbf{X} = \{x_1, \dots, x_p\}$, such that $x_i \in \mathcal{V}$, be a subword sequence representation of sentence X , then the probability of $\mathbf{X}$ is defined as:

$$P(\mathbf{X}) = \prod_{i=1}^p P(x_i), \quad x_1, \dots, x_p \in \mathcal{V} \text{ and } \sum_{v=1}^{|\mathcal{V}|} P(x_v) = 1, \quad (4.1)$$

where the sums being equal to 1 is a constraint. By applying the unigram language model to estimate the probability distribution underlying how a sentence is tokenized, we assume that for a given vocabulary the occurrence of each token in a sentence is independent, and the probability of each token is a measure of how often said token is observed in the language pair relative to all of the tokens in the vocabulary³. Therefore, the probability of the sequence representation of a sentence is simply the product the probabilities corresponding to each token in the sequence.

The process of preparing a joint vocabulary using the ULM tokenization algorithm is also broken down into an initialisation, and an iterative step. In the first step, a large initial vocabulary is constructed containing all unique single-character tokens, and frequently observed subword tokens in the training corpus⁴. Similarly to BPE, the only subword tokens included in the large vocabulary are those contained within a single word.

The goal of the second step is to find an optimal subset of tokens from the initial vocabulary. It leverages the distribution defined in Equation 4.1, and the training corpus to estimate the probabilities of the tokens from a given vocabulary. The estimated probabilities are then used to

²The process of encoding a word as sequence of subword tokens is outlined in a [blog post](#) (Mao, 2019) authored by Lei Mao. [Papers With Code](#) (a website that has since been sunsetted as of July 2025) recognised this blog post as a supplementary resource for understanding BPE.

³Recall that we are preparing a joint vocabulary for both sides of each parallel corpus.

⁴Kudo (2018) proposes applying BPE (Sennrich et al., 2016b) with an adequate number of merge operations, or the Enhanced Suffix Array (ESA) algorithm (Nong et al., 2009), to prepare a large vocabulary containing subwords tokens that are frequently observed in the training corpus.

inform which tokens to filter out from the vocabulary until a pre-defined vocabulary size is reached. For each iteration, the following sub-steps are carried out:

1. Apply the Expectation-Maximization (EM) algorithm to optimise the probability of observing each token in the given vocabulary, that is $P(x_v)$ for each $x_v \in \mathcal{V}$. In Appendix B, we provide a brief explanation detailing how the classical EM algorithm works.
2. Using the optimised probabilities computed for the tokens in the vocabulary, find an optimal subset of tokens to retain in the vocabulary. To determine which tokens to remove from the vocabulary, compute a $loss_v$ value for each token, $x_v \in \mathcal{V}$:
 - $Q(\theta|\theta^{(t)})$ is the expectation of the log-likelihood function over all valid subword sequence representations of sentences, given the sentences in training corpus, and the current estimate of the probabilities computed for the tokens in the vocabulary. In Appendix B, we detail how $Q(\theta|\theta^{(t)})$ is computed.
 - The $loss_v$ value is a measure of how much $Q(\theta|\theta^{(t)})$ decreases by, if $Q(\theta|\theta^{(t)})$ is computed with the corresponding token, x_v , removed from the current vocabulary; and
3. Update the vocabulary by removing $(100 - \eta)\%$ tokens with the lowest $loss_v$ values. These tokens are assumed to contribute the least to how sentences in the training corpus are tokenized, since the removal of them results in a negligible decrease to the function $Q(\theta|\theta^{(t)})$. An exception is, however, made for single-character tokens in the vocabulary. Similar to BPE, these tokens are kept in the vocabulary to ensure that all words in the training corpus can be encoded by tokens in the vocabulary.

Once the above iterative step is complete, the resulting vocabulary consists of tokens alongside their probabilities, which estimate how often each token is observed in the language pair.

How to tokenize a sentence:

A sentence, X , is tokenized by searching for the most probable sequence of tokens, $\mathbf{X}^*$, from the set of all valid subword sequence representations of the sentence, $\mathcal{S}(X)$, generated using tokens from the recently constructed vocabulary:

$$\mathbf{X}^* = \operatorname{argmax}_{\mathbf{X} \in \mathcal{S}(X)} P(\mathbf{X}) \quad (4.2)$$

A brute force approach for searching for the most probable subword sequence representation of a sentence would be to search for all valid subword sequence representations of the sentence in the set $\mathcal{S}(X)$, and compute the corresponding probabilities for each representation, $P(\mathbf{X})$ for $\mathbf{X} \in \mathcal{S}(X)$. However, this approach is computationally expensive. Its order of computational complexity is given by $O(|\mathcal{V}|^T)$, where T is the number of characters in the sentence (Jurafsky and Martin, 2024). As the length a sentence increases, the order of computation increases exponentially.

The ULM tokenization algorithm uses the Viterbi algorithm (Viterbi, 1967) to search for the most probable subword sequence representation of a sentence. This algorithm applies Dynamic Programming (DP) to perform a more efficient search over the vocabulary. By doing so, it reduces the order of computational complexity for searching for the most probable subword sequence representation to $O(|\mathcal{V}|^2 \cdot T)$ (Jurafsky and Martin, 2024). In Appendix C, we provide a brief explanation detailing how the Viterbi algorithm works.

4.1.3 Threshold on Joint Vocabulary

Recall that when constructing a joint vocabulary, a minimum threshold is applied to determine which tokens are to be retained in the vocabulary: a token is only retained in the vocabulary if it has been observed on both sides of the training corpus. In practice, thresholding is applied after sentences in the training, validation and test parallel corpora have been tokenized. This means that the joint vocabulary passed to the NMT model will only contain tokens that been observed on both sides of the training corpus. Further, the positions in the subword sequence representations of sentences in the training, validation and test corpora that contain tokens that have been removed from the vocabulary, will be mapped to the unknown symbol token. By exposing the NMT model to unknown symbol tokens during training, we increase its robustness to unseen words and symbols during test time (Provilkov et al., 2020).

4.2 Subword Regularization

The tokens selected for a vocabulary constructed by the standard subword tokenization algorithms are selected based on their frequency in the training corpus, rather than on a set of manually-crafted rules outlining the word formation processes in each language. These tokenization algorithms then use the vocabulary to output unique subword sequence representations of sentences in the training, validation and test corpora. The implication of this approach, is that the unique subword sequence representations that are generated of sentences are prone to segmentation errors, errors that in turn impact the model’s capacity to learn the underlying word formation processes in each language. In low-resource data settings, these segmentation errors are especially common because there are limited examples in the training corpus to inform appropriate choices for tokens for the vocabulary.

Kudo (2018) proposes applying subword regularization to how sentences in the training corpus are tokenized. This process involves generating multiple subword sequence representations of sentences, and subjecting the model to the multiple representations during training. By injecting noise into the model, by way of the multiple subword sequence representations, subword regularization prevents the NMT model from over-fitting on potentially sub-optimal subword sequence representations of sentences in the training corpus.

The two subword regularization techniques considered in this thesis are modifications of the BPE and ULM tokenization algorithms, respectively, namely BPE-Dropout (Provilkov et al., 2020) and the ULM with subword regularization (Kudo, 2018).

4.2.1 BPE-Dropout

The process of generating multiple subword sequence representations of sentences in the training corpus using BPE-Dropout is made up of major two steps, that is an initialisation, and an iterative step. In the first step, this method applies standard BPE to the training corpus to construct a joint vocabulary, and a merge table. The merge table outlines which symbols in the training corpus have been merged to form tokens in the vocabulary, and how the merge operations are ranked in order of importance. Following from the example in Table 4.1, Table 4.3 is the corresponding merge table for the vocabulary that was generated in that example.

Merge Operation	Frequency Count
(‘e’, ‘s’)	9
(‘es’, ‘t</w>’)	9
(‘l’, ‘o’)	7
(‘n’, ‘e’)	6
(‘ne’, ‘w’)	6

Table 4.3: Example of a merge table.

In the second step, BPE-Dropout generates multiple subword sequence representations for each sentence in the training corpus, by generating a new subword sequence representation for each word in the sentence. In preparation for this iterative step, each word is represented as a sequence of characters⁵. BPE-Dropout then generates a new subword sequence representation of the word from its encoding as a sequence of character tokens by performing the following sub-steps iteratively:

1. Use the merge table to compile a list of the merge operations that are possible from the symbols in the current sequence representation of the word;
2. From the list of merge operations, exclude merge operations from the list with probability, p_{drop} . If the same merge operation occurs multiple times in a word, then each occurrence of the merge operation is treated independently when deciding whether to exclude it from the list⁶;

⁵Like with the standard BPE, a special end-of-word character (</w>) is concatenated to the final character in the word.

⁶Provilkov et al. (2020) present the following example of a sequence representation of the word "merger": {‘m’, ‘e’, ‘r’, ‘g’, ‘e’, ‘r’}. From the sequence representation, the list of possible merge operations is given by [(‘m’, ‘e’), (‘e’, ‘r’), (‘r’, ‘g’), (‘g’, ‘e’), (‘e’, ‘r’)]. In BPE-Dropout, both occurrences of (‘e’, ‘r’) will be treated independently in the list.

3. If the list of merge operations is empty, then the iterative procedure is concluded, and it returns the current sequence representation of the word. Else, it applies the merge operation from the list with the highest priority, according to the merge table, to the current sequence representation of the word.

The probability, denoted as p_{drop} , is a hyperparameter that controls the degree that words are segmented by in BPE-Dropout. If $p_{drop} = 0$, then no merge operations are excluded from the list of possible merge operations, and the resulting subword sequence representation of a word is equivalent to a representation that would be generated by standard BPE. If $p_{drop} = 1$, then all merge operations are excluded from the list, and the resulting subword sequence representation of a word is a sequence of single-character tokens.

When applying BPE-Dropout to tokenize sentences in the training corpus, the process may result in a small number of unknown symbols in the corpus. [Provilkov et al. \(2020\)](#) conduct experiments comparing (i.) a vocabulary developed by BPE, with (ii.) the same vocabulary, however, it has been modified to also include the unknown symbols that were generated as a result of BPE-Dropout. The outcome of the experiments suggest that there isn't a significant difference between the two approaches. This thesis will not modify the vocabulary that is developed using BPE when applying BPE-Dropout. This is done to increase the model's robustness to unknown symbols during test time, and to ensure that the same settings are used in BPE-Dropout and BPE for comparability⁷.

How to tokenize sentence:

The different subword sequence representations of sentences in the training corpus are obtained by applying the BPE-Dropout algorithm described above to each word in the sentence, to generate a new representation of the word. Similarly to [Provilkov et al. \(2020\)](#), sentences in the validation and test corpora are tokenized in the same manner that they are tokenized under the standard BPE framework, that is that unique subword sequence representations are generated for each sentence in these corpora.

4.2.2 Unigram Language Modelling with Subword Regularization

The process of generating multiple subword sequence representations of sentences in the training corpus using the ULM tokenization algorithm with subword regularization is also made up of two major steps, namely an initialisation, and a sampling step. In the first step, this method applies the standard ULM tokenization algorithm to the training corpus to construct a joint vocabulary, such that each token in the vocabulary has a probability estimating its occurrence in the language pair.

Let X be a sentence, and $S(X) = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_l\}$ the set of subword sequence representations of sentence X that are constructed from the given vocabulary. Following from Equation 4.1, we can compute the probability of each subword sequence representation in the set $S(X)$. To generate a new subword sequence representation of the sentence, X , the second step of the ULM tokenization algorithm with subword regularization samples a subword sequence representation from a multinomial distribution:

$$P(\mathbf{X}_i) \cong \frac{P(\mathbf{X}_i)^\alpha}{\sum_{j=1}^l P(\mathbf{X}_j)^\alpha} \quad i \in \{1, \dots, l\}, \quad (4.3)$$

where, $\alpha \in \mathcal{R}^+$ is the hyperparameter controlling the smoothness of the multinomial distribution, that is that the smaller α is, the closer the distribution is to approximating a uniform distribution; the larger α is, the more likely the distribution is to assigning larger probabilities to subword sequence representations in $S(X)$ that maximise the distribution in Equation 4.1. $l \in \mathcal{Z}^+$ is the hyperparameter controlling the size of $S(X)$, that is the number of subword sequence representations that the multinomial distribution can sample from. When l is a finite integer, then the set $S(X)$ is restricted to the l -best subword sequence representations of sentence X that maximise Equation 4.1. [Kudo \(2018\)](#) apply the Forward-DP Backward-A* ([Nagata, 1994](#)) to determine the l -best subword sequence representations of the sentence, X , constructed from a given vocabulary. When $l \rightarrow \infty$, then the set $S(X)$ takes into account all possible subword sequence representations of the sentence X . [Kudo \(2018\)](#) apply the Forward-Filtering and Backward-Sampling (FFBS)

⁷This thesis is only interested in analysing the impact that exposing the NMT model to different subword sequence representations of sentences in the training corpus has on model performance, whilst maintaining the same vocabulary size across models.

algorithm (Scott, 2002) to sample from all possible subword sequence representations constructed from the given vocabulary⁸. The larger l is, the more aggressive the regularization is due to the number of available subword sequence representations that can be sampled. Experiments in Kudo (2018) indicate the NMT models trained in low-resource data settings benefit more from this aggressive form regularization, when compared to models trained in high-resource data settings. For this reason, this thesis sets the hyperparameter $l \rightarrow \infty$.

How to tokenize a sentence:

The different subword sequence representations of sentences in the training corpus are obtained by applying the ULM tokenization algorithm with subword regularization sampling protocol described above. Kudo (2018) presents two approaches for tokenizing sentences in the validation and test parallel corpora. The first approach tokenizes sentences in these corpora in the same manner that sentences are tokenized under the standard ULM framework, that is that the Viterbi algorithm is applied to search for a unique subword sequence representation for each sentence that maximises the Equation 4.1.

The second approach applies the Forward-DP Backward-A* algorithm (Nagata, 1994), to search for the n -best subword sequence representations of source sentences in the validation and test corpora. During validation and testing, the NMT model generates translations, $\mathbf{Y}_1, \dots, \mathbf{Y}_n$, from the n subword sequence representations of a source sentence X , denoted as $\mathbf{X}_1, \dots, \mathbf{X}_n$. However, the NMT model will only output the translation, $\mathbf{Y}^*$, maximises the following score:

$$\text{score}(\mathbf{X}_i, \mathbf{Y}_i) = \frac{\log P(\mathbf{Y}_i | \mathbf{X}_i)}{|\mathbf{Y}_i|^\lambda} \quad i \in \{1, \dots, n\}, \quad (4.4)$$

where, $|\mathbf{Y}_i|$ is the number of tokens in the subword sequence representation of the translation, and $\lambda \in \mathbb{R}^+$ is a hyper-parameter that is responsible for penalising shorter translations generated of the source sentence.

To remain consistent with how sentences in the validation and test corpora are tokenized in BPE-Dropout, this thesis will apply the first approach for tokenizing sentences in the validation and test corpora.

4.3 Task-specific Subword Tokenization

For both standard subword tokenization and subword regularization methods, the process of tokenizing sentences is performed separately from the training of the NMT model. This means that the selection of tokens in the vocabulary, and how sentences are tokenized is only informed by the training corpus, ignoring any other context (Hiraoka et al., 2021). This thesis applies a tokenizer that accounts for NMT in its development. For a given vocabulary, the tokenizer is responsible representing a sentence as a sequence of tokens. The tokenizer applied in this thesis will specifically only tokenize target sentences.

The target language tokenizer is based on a neural network that models the probability of observing the tokens in the vocabulary. Using the estimated probabilities for the tokens, a sequence of tokens can be sampled to represent a sentence. To account for NMT in its development, the target language tokenizer is trained jointly with an NMT model.

Jointly training the target language tokenizer with an NMT model facilitates a way to streamline tokenizing sentences and training an NMT model into one step. However, this significantly increases the number of parameters there are in the translation model, leading to longer model training times. Given the time and resource constraints that this thesis operates in, this is not ideal. To reduce lengthy model training times, this thesis will pre-train the target language tokenizer to approximate the probabilities of the tokens estimated by the ULM tokenization algorithm (Hiraoka et al., 2021), and then it will jointly train the tokenizer with an NMT model that has same architecture as the NMT model that will be used for translation. In doing so, the tokenizer can leverage the loss function of the NMT model to fine-tune its parameters.

The architecture of the target language tokenizer is based on the work of Hiraoka et al. (2021). The distinction between this thesis and Hiraoka et al. (2021) lies in the implementation of this

⁸In Appendix D, we provide a brief explanation detailing how the FFBS algorithm works.

tokenizer: [Hiraoka et al. \(2021\)](#) aim to develop a model that jointly tokenizes and translates sentences; whereas, this thesis aims to use the tokenizer that has been jointly trained with model, that has same architecture as the translation model, to prepare sentences for the training of a separate translation model. Since the joint model in this thesis is just used to fine-tune the target language tokenizer, the assumption is that the process will not require as many steps to train, relative to the training of the joint model in [Hiraoka et al. \(2021\)](#).

4.3.1 Target Neural Unigram Language Model

Let Y be a target sentence, and $\mathbf{Y} = \{y_1, \dots, y_q\}$ be a subword sequence representation of the sentence, such that each y_i is a subword token from a vocabulary $\mathcal{V}$. The Target Neural Unigram Language Model (Target NULM) is the architecture for the target language tokenizer ([Hiraoka et al., 2021](#)). Similarly to ULM, it assumes that the probability distribution of $\mathbf{Y}$ is given by Equation 4.1, that is:

$$P(\mathbf{Y}) = \prod_{j=1}^q P(y_j), \quad y_1, \dots, y_q \in \mathcal{V} \text{ and } \sum_{v=1}^{|\mathcal{V}|} P(y_v) = 1, \quad (4.5)$$

where the sums being equal to 1 is a constraint. The Target NULM comprises an identical, simple, position-wise, fully-connected, feed-forward neural network to estimate the probability of observing each token in the vocabulary. Each token in the vocabulary, $y_v \in \mathcal{V}$, can be represented as one-hot vector, $\mathbf{y}_v \in \mathbb{R}^{|\mathcal{V}|}$, such that the position of the non-zero entry in the vector corresponds to the token's index, v , in the vocabulary. The Target NULM begins by receiving a token as input, and returning the following score:

$$o(\mathbf{y}_v) = [\sigma(e(\mathbf{y}_v)\mathbf{W}_1 + b_1)]\mathbf{W}_2 + b_2, \quad v \in \{1, \dots, |\mathcal{V}|\}, \quad (4.6)$$

where, $e(\mathbf{y}_v)$ is an embedding of the token, y_v , that is obtained by applying a linear transformation to the one-hot vector representation of the token, that is $e(\mathbf{y}_v) = \mathbf{x}_v\mathbf{W}_0$; σ denotes the tanh activation function; $\mathbf{W}_0 \in \mathbb{R}^{|\mathcal{V}| \times d_e}$, $\mathbf{W}_1 \in \mathbb{R}^{d_e \times d_f}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_f \times 1}$ are weight matrices; and $b_1 \in \mathbb{R}^{d_f}$ and $b_2 \in \mathbb{R}$ are bias vectors.

To estimate the probability of each token, the Target NULM applies a softmax function to scores generated for each token in the vocabulary:

$$P(\mathbf{y}_v) = \frac{\exp(o(\mathbf{y}_v))}{\sum_{w=1}^{|\mathcal{V}|} \exp(o(\mathbf{y}_w))}, \quad \sum_{v=1}^{|\mathcal{V}|} P(\mathbf{y}_v) = 1, \quad (4.7)$$

where the sums being equal to 1 is a constraint. Since the occurrence of each token in the subword sequence representation of the target sentence is assumed to be independent, the probability of the the subword sequence representation is computed by taking the product of the probabilities of the tokens, estimated by the Target NULM:

$$P(\mathbf{Y}) = P(\{y_1, \dots, y_q\}) = \prod_{i=1}^q P(y_i). \quad (4.8)$$

How to train a Target NULM:

The process of training the Target NULM is broken down into two major steps, namely a pre-training, and a fine-tuning step. The pre-training step begins with applying ULM to construct a joint vocabulary, $\mathcal{V}$. Each token in the vocabulary corresponds to a probability measuring its occurrence in the language pair. Kullback-Leibler (KL) divergence is then used to pre-train the Target NULM ([Hiraoka et al., 2020](#)).

Under the KL divergence framework, we let P denote the probability distribution of observing tokens in the vocabulary, which is approximated by ULM; and Q denote the probability distribution of observing the tokens, which will be approximated by the Target NULM. The goal of the KL divergence is to quantify the difference between the two distributions P and Q :

$$D_{KL}(P||Q) = \sum_{v=1}^{|\mathcal{V}|} P(y_v) \log \left(\frac{P(y_v)}{Q(y_v)} \right) \quad (4.9)$$

For each parameter update, the goal of pre-training is to search for the set of parameters in the Target NULM that minimise the KL divergence, $D_{KL}(P||Q)$.

The fine-tuning step involves jointly training the target language tokenizer with a model that has same architecture as the NMT model used for translation. By doing so, the target language tokenizer can leverage the loss function of the “NMT model” to fine-tune its parameters. Let $(X, Y) \in \mathcal{D}$ denote a source-target sentence pair in the training corpus. The process of leveraging the loss function requires that we define a loss value for each sentence pair, L_Y :

1. Using the probabilities estimated for the tokens in the vocabulary by the Target NULM, apply the Forward-DP Backward-A* algorithm (Nagata, 1994) to search for the m -best subword sequence representations of target sentence Y , that is $\mathbf{Y}_1, \dots, \mathbf{Y}_m$, and compute the probability for each subword sequence representation, $P(\mathbf{Y}_i)$ for $i \in \{1, \dots, m\}$ ⁹;
2. Using the probabilities estimated for the tokens in the vocabulary by ULM, apply the FFBS algorithm (Scott, 2002) to sample a subword sequence representation of the source sentence X , that is $\mathbf{X}$;
3. Pass the source sentence to the "NMT model" for processing, denoted $f(\mathbf{X})$; and compute a loss value measuring the difference between the model's prediction and each subword sequence representation of the target sentence. The loss function for the “NMT model” is denoted as $g(\cdot, \cdot)$:

$$L_{Y_i} = g(f(\mathbf{X}), \mathbf{Y}_i) \quad i \in \{1, \dots, m\} \quad (4.10)$$

4. Compute the overall loss value for the sentence pair, L_Y , by taking the weighted sum of the loss values computed in the previous step for each of the m subword sequence representations of the target sentence:

$$L_Y = \sum_{i=1}^m a_i L_{Y_i}, \quad (4.11)$$

such that each weight, a_i , is computed by normalising the probability of the subword sequence representation of the target sentence across the probabilities of the m -best subword sequence representations:

$$a_i = \frac{P(\mathbf{Y}_i)}{\sum_{j=1}^m P(\mathbf{Y}_j)} \quad i \in \{1, \dots, m\}. \quad (4.12)$$

Under this joint training framework, the parameters of the NMT model are being trained to minimise the expected loss over the subword sequence representations of each target sentence in the training corpus, L_Y . By doing so, the parameters of the Target NULM update in such a way that the tokenizer is encouraged to allocate a higher probability to the subword sequence representation of the target sentence that minimises the expected loss.

How to tokenize sentences:

Using the probabilities estimated by the fine-tuned Target NULM, we apply the Viterbi algorithm (Viterbi, 1967) to search for the most probable sequence of subword tokens to represent each target sentence in the training and validation corpora. Using the probabilities estimated by ULM, we apply the Viterbi algorithm to search for the most probable sequence of subword tokens to represent each source sentence in the training, validation and test corpora.

We generate unique subword sequence representations of sentences in the validation and test corpora to remain consistent with how the standard subword tokenization and the subword regularization methods evaluate the NMT models.

4.4 Neural Machine Translation

NMT is introduced in early literature as an approach to MT that uses artificial neural networks to model the translation of text from one language into another (Kalchbrenner and Blunsom, 2013; Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014; Sutskever et al., 2014). Typically, its models are composed of an encoding and decoding component. The encoding component reads a source sequence, and maps it to some vector representation.

⁹In Appendix E, we provide a brief explanation detailing how the Forward-DP Backward-A* algorithm works.

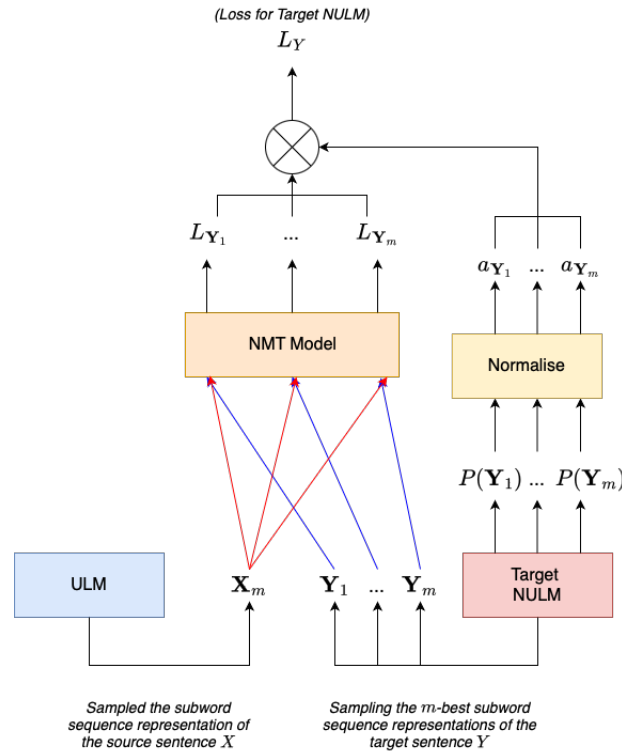


Figure 4.1: A configuration of how the Target NULM is jointly trained with an NMT model.

This vector representation can be as simple as a fixed-length dimensional vector, or as complex as a sequence of vectors (Bahdanau et al., 2016; Luong, Pham and Manning, 2015); at its core, it functions as a tool to summarise all necessary information in the source sequence to generate a translation of the associated source sentence. The decoding component then interprets the output of the encoding component to predict an appropriate target sequence. Early NMT models made use of recurrent and convolutional neural network architectures in their encoding and decoding components (Kalchbrenner and Blunsom, 2013; Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014; Cho, van Merriënboer, Bahdanau and Bengio, 2014), which went on to be replaced by stacks of these neural network architectures in both components (Sutskever et al., 2014; Wu et al., 2016; Gehring et al., 2017). Common choices for recurrent neural network architectures included the Long Short-Term Memory (Hochreiter and Schmidhuber, 1997), and the Gated Recurrent Unit (Cho, van Merriënboer, Çaglar Gülçehre, Bahdanau, Bougares, Schwenk and Bengio, 2014) networks. The introduction of the Transformer architecture (Vaswani et al., 2023) for NMT aimed to replace these recurrent and convolutional neural network architectures with self-attention mechanisms to improve the training efficiency of the model¹⁰. This would be accomplished by **minimising**:

1. The amount of sequential operations required to process a source-target sequence pair during training, and instead implementing an architecture that prioritises increased parallelization of processes in the model;
2. The order of computational complexity in the model when compared to recurrent and convolutional neural network architectures, that is, the self-attention mechanism reduces its order of computational complexity by ensuring that the length of sequence representations of sentences is shorter than the dimensions of the vectors in the sequence used to represent tokens; and
3. The maximum path length between tokens in the source-target sequence pair, which in turn improves the NMT model's ability to capture long-range dependencies that exist between the tokens in the sequence pair.

The above improvements to the training efficiency of NMT models are especially useful as both

¹⁰The Transformer is introduced in 2017; however, the original publication introducing the work has since been revised. The latest version to date of the same publication was released on 2 August 2023.

Layer Type	Computational Complexity	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

Table 4.4: Summary of the order of computational complexity, sequential operations and maximum path lengths for a self-attention mechanism, recurrent neural network and convolutional neural network (Vaswani et al., 2023).

the size of the parallel corpora used to train these models, and the length of sentences in the corpora increases. Table 4.4 presents a summary of how the order of computational complexity, sequential operations and maximum path lengths differ between the self-attention mechanism, and recurrent and convolutional neural network architectures. NMT models based on the Transformer architecture have shown to successfully achieve state-of-the-art translation performance: producing more reliable translations, whilst using only a fraction of the training costs required to prepare earlier state-of-the-art NMT models which relied on recurrent and convolutional neural network architectures (Wu et al., 2016; Kalchbrenner et al., 2017; Zhou et al., 2016; Gehring et al., 2017; Shazeer et al., 2017).

4.4.1 Architecture of the Transformer

Let (X, Y) be a source-target sentence pair, with $\mathbf{X} = \{x_1, \dots, x_p\}$ denoting the subword sequence representation of X , and $\mathbf{Y} = \{y_1, \dots, y_q\}$ the subword sequence representation of Y . The Transformer is trained to learn the conditional probability distribution of observing sentence Y given sentence X .

Using the chain-rule, the conditional probability distribution can be decomposed such that it is equivalent to computing the product of the probability of observing each token in the target sequence conditioned on the preceding tokens in the sequence and the source sequence:

$$\begin{aligned}
 P(Y|X) &= P(\mathbf{Y}|\mathbf{X}) \\
 &= P(\{y_1, \dots, y_q\}|\mathbf{X}) \\
 &= \prod_{j=1}^q P(y_j|\{y_1, \dots, y_{j-1}\}; \mathbf{X}).
 \end{aligned} \tag{4.13}$$

By decomposing the conditional probability distribution, we equip the Transformer with an autoregressive property. This property ensures that predictions made for any token in the target sequence can only depend on the preceding tokens in the sequence. The main components of the Transformer architecture include an encoding and a decoding component.

Encoder:

The encoding component is made up of N encoder layers. Following from the work of Sutskever et al. (2014), it is common practice in NMT to exploit the benefits of stacking multiple artificial neural networks in the encoding and decoding components to improve the translation performance of these models. The source sequence is fed into the encoding component as input; however, before the encoder layers can process the input, each token in the source sequence is transformed into an embedding representation. The encoding component then outputs a new representation of the source sequence that captures the relationship between each token and all the tokens in the source sequence.

Each encoder layer contains two sub-layers:

1. a Multi-Head Self-Attention Mechanism;
2. a simple, position-wise, fully-connected Feed-Forward Neural Network.

The output of each sub-layer is a sequence of vectors, such that each vector is of dimension d . As a way to regularize the Transformer during training, Vaswani et al. (2023) suggest applying

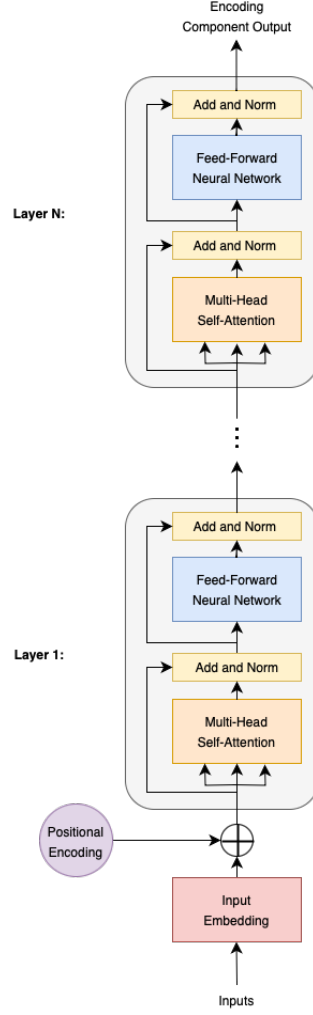


Figure 4.2: Configuration of the encoding component in the Transformer.

dropout (Srivastava et al., 2014) to the output sequence of each sub-layer. The output sequence is then adjusted via a residual connection (He et al., 2015), followed by layer normalization (Baptiste et al., 2016), all before being passed to the subsequent sub-layer. A visualisation of the encoding component is provided in Figure 4.2.

Decoder:

Similarly to the encoding component, the decoding component is also made of N decoder layers. During training, it receives the target sequence as one of its inputs. To ensure that the Transformer adheres to the auto-regressive property, tokens in the sequence are shifted to the right, resulting in a new target sequence $\mathbf{Y} = \{y_0, y_1, \dots, y_q\}$, where y_0 is a special beginning of sequence token ($\langle \text{bos} \rangle$). The new target sequence is then converted to an embedding representation.

The decoding component uses the embedding representation, combined with the output of the encoding component, to approximate the conditional probability distribution, $P(Y|X)$. Following from Equation 4.13, the decoding component is trained to output the probability of observing the next token y_j in the target sequence for each position $j \in \{1, \dots, q\}$ in the sequence:

$$P(y_j | \{y_0, \dots, y_{j-1}\}; \mathbf{X}). \quad (4.14)$$

Each decoder layer contains three sub-layers:

1. a masked Multi-Head Self-Attention Mechanism;
2. a Multi-Head Self-Attention Mechanism;
3. a simple, position-wise, fully-connected Feed-Forward Neural Network.

The output of each sub-layer is a sequence of vectors, such that each vector is of dimension d . As a way to regularize the Transformer during training, dropout is applied to the output sequence of each sub-layer. The output sequence is then adjusted via a residual connection, followed by layer normalization, all before being passed to the subsequent sub-layer. A visualisation of the decoding component is provided in Figure 4.4.

4.4.2 Embedding Representations

Recall that we apply a subword tokenization algorithm to construct a joint vocabulary, $\mathcal{V}$, containing subword tokens found on both sides of the corpus. Sentences are then encoded as a sequence of subwords from the vocabulary. In order for the Transformer to be able to read these sequence representations as input, subword tokens in the vocabulary are mapped to one-hot vector representations of dimension $|\mathcal{V}|$, such that the position of the non-zero entry in the vector corresponds to the token's index in the vocabulary. By representing tokens as one-hot vectors, the assumption is that each token is independent of every other token in the vocabulary; therefore, each sequence representation of one-hot vectors fed to the Transformer as input captures no information about the underlying relationship between the tokens in the sequence.

The embeddings layer in the encoding and decoding components of the Transformer is responsible for transforming the one-hot vectors to vector representations that capture the relationship between the tokens in the vocabulary. Suppose that we convert the subword sequence representation of the source sentence to $\mathbf{X} = \{x_1, \dots, x_p\} = \{\mathbf{x}_1, \dots, \mathbf{x}_p\}$, where each $\mathbf{x}_i \in \mathbb{R}^{|\mathcal{V}|}$ is a one-hot vector encoding of the subword token x_i , and in a similar fashion we convert the subword sequence representation of the target sentence to $\mathbf{Y} = \{y_0, y_1, \dots, y_q\} = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_q\}$, where each $\mathbf{y}_j \in \mathbb{R}^{|\mathcal{V}|}$ is a one-hot vector encoding of the subword token y_j . An embedding representation of the source or target sequence is generated by applying a linear transformation to each one-hot vector in the sequence representation: $e(\mathbf{x}_i) = \mathbf{x}_i \mathbf{W}$ for $i \in \{1, \dots, p\}$, and $e(\mathbf{y}_j) = \mathbf{y}_j \mathbf{W}$ for $j \in \{0, 1, \dots, q\}$.

Following from the work of Vaswani et al. (2023), the same trainable weight matrix, $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$, is used in both embedding layers of the encoding and decoding components, and it is scaled by a factor of $\sqrt{d}$. This means that the embedding representation of the source and target sequence is computed as follows:

- $e(\mathbf{x}_i) = \mathbf{x}_i \left(\sqrt{d} \times \mathbf{W} \right)$, for $i \in \{1, \dots, p\}$; and
- $e(\mathbf{y}_j) = \mathbf{y}_j \left(\sqrt{d} \times \mathbf{W} \right)$ for $j \in \{0, 1, \dots, q\}$.

Positional Encoding:

The Transformer replaces the recurrent and convolutional neural network architectures in the encoding and decoding components, with self-attention mechanisms, to minimise the amount of sequential operations required to process a sequence. However, the sequential nature of recurrent and convolutional neural network architectures enable an NMT model to automatically extract information about the relative and/or absolute position of tokens in a sequence (Vaswani et al., 2023; Gehring et al., 2017).

The Transformer can be equipped to extract information about the order of tokens in a sequence by adjusting each embedding representation of sequence with a positional encoding of the tokens in the sequence. The positional encoding corresponding to the source sequence is denoted as $\mathbf{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_p\}$, where each $\mathbf{p}_i \in \mathbb{R}^d$. Vaswani et al. (2023) define the vectors in the positional encoding as follows:

- consider the i^{th} vector in the positional encoding, $\mathbf{p}_i$, elements of the vector positioned at even indices, $2f$, are evaluated as $\mathbf{p}_{(i,2f)} = \sin(i/10000^{2f/d})$; and
- consider the i^{th} vector in the positional encoding, $\mathbf{p}_i$, elements of the vector positioned at odd indices, $2f + 1$, are evaluated as $\mathbf{p}_{(i,2f+1)} = \cos(i/10000^{2f/d})$,

for $i \in \{1, \dots, p\}$ and $f \in \{0, \dots, \frac{d}{2}\}$. This definition is the same for computing the positional encoding for the target sequence, which is denoted as $\mathbf{P}' = \{\mathbf{p}'_0, \mathbf{p}'_1, \dots, \mathbf{p}'_q\}$, where each $\mathbf{p}'_i \in \mathbb{R}^d$. Vaswani et al. (2023) represent each vector in the positional encoding as a series of sine and cosine functions because it enables the Transformer to extract information about each token relative to

neighbouring tokens in a sequence. This encoding can also easily be extended to longer sequences in the test corpus.

Following from the work of Vaswani et al. (2023) and Gehring et al. (2017), the positional encoding is integrated into the Transformer by summing the embedding representation of a sequence with its corresponding positional encoding

- $\mathbf{X}' = \{e(\mathbf{x}_1) + \mathbf{p}_1, \dots, e(\mathbf{x}_p) + \mathbf{p}_p\}$ is the sum of the embedding representation of the source sequence and its corresponding positional encoding, such that each $e(\mathbf{x}_i) + \mathbf{p}_i \in \mathbb{R}^d$; and
- $\mathbf{Y}' = \{e(\mathbf{y}_0) + \mathbf{p}'_0, e(\mathbf{y}_1) + \mathbf{p}'_1, \dots, e(\mathbf{y}_q) + \mathbf{p}'_q\}$ is the sum of the embedding representation of the target sequence and its corresponding positional encoding, such that each $e(\mathbf{y}_j) + \mathbf{p}'_j \in \mathbb{R}^d$.

4.4.3 Self-Attention Mechanism

The self-attention mechanism is given two sequences as input: $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_s\}$, such that each $\mathbf{a}_i \in \mathbb{R}^d$, and $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_t\}$, such that each $\mathbf{b}_j \in \mathbb{R}^d$. It begins by applying a linear transformation on the vectors in sequence $\mathbf{A}$, which results in a sequence of query vectors $\{\mathbf{q}_1, \dots, \mathbf{q}_s\}$, such that each $\mathbf{q}_i \in \mathbb{R}^{d_k}$. It then also applies two separate linear transformations on the vectors in sequence $\mathbf{B}$, which results in a sequence of key vectors, $\{\mathbf{k}_1, \dots, \mathbf{k}_t\}$, such that each $\mathbf{k}_j \in \mathbb{R}^{d_k}$, and a corresponding sequence of value vectors, $\{\mathbf{v}_1, \dots, \mathbf{v}_t\}$, such that each $\mathbf{v}_j \in \mathbb{R}^{d_v}$.

For each query vector, $\mathbf{q}_i$, the self-attention mechanism computes a score measuring how compatible $\mathbf{q}_i$ is with each key vector in the sequence $\{\mathbf{k}_1, \dots, \mathbf{k}_t\}$. Using the scores, it outputs a vector representation that summarises the compatibility of $\mathbf{q}_i$ with every key vector. This computation is implemented in the self-attention mechanism using a scaled dot-product attention function:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} \right) \mathbf{V}, \quad (4.15)$$

where, $\mathbf{Q} \in \mathbb{R}^{s \times d_k}$ is a matrix of query vectors stacked on top of one another, similarly $\mathbf{K} \in \mathbb{R}^{t \times d_k}$ is a matrix of the stacked key vectors, and $\mathbf{V} \in \mathbb{R}^{t \times d_v}$ is a matrix of the stacked value vectors.

Each row of the resulting matrix obtained from computing $\mathbf{Q}\mathbf{K}^\top$ in the attention function, is equivalent taking the dot product of a query vector, $\mathbf{q}_i$, with each key vector in the sequence $\{\mathbf{k}_1, \dots, \mathbf{k}_t\}$. The compatibility score of two vectors is computed using a dot product because the dot product is able to measure the degree of similarity between the vectors. The matrix $\mathbf{Q}\mathbf{K}^\top$ is then scaled by a factor of $\frac{1}{\sqrt{d_k}}$ to prevent unstable training that may occur as a result of large values produced by the dot-product operation (Vaswani et al., 2023).

The attention function then applies a softmax function row-wise. This means that the compatibility scores computed for each query vector, $\mathbf{q}_i$, are converted into probabilities, such that $\sum_{j=1}^t P \left(\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}} \right) = 1$ for each $i \in \{1, \dots, s\}$. Finally, each row of probabilities linked to $\mathbf{q}_i$ is then multiplied with the values matrix, $\mathbf{V}$, to generate a transformed vector representation of the query vector of dimension d_v .

Multi-Head Self-Attention:

Vaswani et al. (2023) propose that instead of mapping the vectors in sequences $\mathbf{A}$ and $\mathbf{B}$ to query, and key-value vectors, respectively, of the same dimension, that is, $d_k = d_v = d$; the vectors in sequences $\mathbf{A}$ and $\mathbf{B}$ should be mapped to query, and key-value vectors of smaller dimension, that is that $d_k, d_v < d$. Vaswani et al. (2023) further propose that instead of applying a single self-attention mechanism; we apply multiple self-attention mechanisms in parallel and concatenate the outputs to generate a new representation for each vector in sequence $\mathbf{A}$:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{concat}(\text{head}_1, \dots, \text{head}_H) \mathbf{W}_M \\ \text{s.t } \text{head}_h &= \text{Attention}(\overline{\mathbf{A}}\mathbf{W}_h^Q, \overline{\mathbf{B}}\mathbf{W}_h^K, \overline{\mathbf{B}}\mathbf{W}_h^V) \quad h \in \{1, \dots, H\} \end{aligned} \quad (4.16)$$

where, $\overline{\mathbf{A}} \in \mathbb{R}^{s \times d}$ is a matrix of stacked vectors from the sequence $\mathbf{A}$, and $\overline{\mathbf{B}} \in \mathbb{R}^{t \times d}$ is a matrix of stacked vectors from the sequence $\mathbf{B}$. $\mathbf{W}_h^Q, \mathbf{W}_h^K \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}_h^V \in \mathbb{R}^{d \times d_v}$ are the weight matrices that are responsible for transforming vectors in sequences $\mathbf{A}$ and $\mathbf{B}$ to query, and key-value vectors, respectively. Each head_h represents a single self-attention mechanism, and its output is a matrix of dimension $s \times d_v$. The multi-head self-attention mechanism concatenates the outputs of the H

self-attention mechanisms row-wise to produce a matrix of dimension $s \times Hd_v$, and transforms the matrix using weight matrix $\mathbf{W}_M \in \mathbb{R}^{Hd_v \times d}$. Each row in the resulting matrix, corresponds to the transformed representation of the vectors in the sequence $\mathbf{A}^{11}$.

The set of weight matrices, $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V$, for each head $_h$ in the multi-head self-attention mechanism are assigned unique initial values. This allows the self-attention mechanism in each head $_h$ to capture insights about the input sequences, with respect to the attention function, from different representation subspaces (Vaswani et al., 2023).

Similarly to Vaswani et al. (2023), this thesis defines the dimensions of the query, and key-value vectors as $d_k = d_v = d/H$. This ensures that the computational cost of the multi-head self-attention mechanism is comparable to a single self-attention mechanism where $d_k = d_v = d$.

Applications of Multi-Head Self-Attention in the Transformer:

The Transformer uses three different types of multi-head self-attention mechanisms in its architecture. The multi-head self-attention mechanism in the encoding component is only provided the output of the previous layer in the encoding component as its input, therefore, $\mathbf{A} = \mathbf{B}$. This means that the role of the multi-head self-attention mechanism in the encoding component is to transform each token in the source sequence to a vector representation that better encapsulates its relationship relative to all tokens in the sequence.

The second multi-head self-attention mechanism is in the decoding component. It is also only provided the output of the previous layer in the decoding component as its input, therefore, $\mathbf{A} = \mathbf{B}$. Since the Transformer is characterised by the auto-regressive property, this multi-head attention mechanism needs be trained so that when it processes a token in the target sequence, it ignores all subsequent tokens in the sequence. To do this, the multi-head self attention mechanism applies a masking function that maps subsequent tokens to $-\infty$ before applying the softmax function in the scaled dot-product function (see Figure 4.3). As a result, this multi-head self-attention mechanism is responsible for transforming each token in the target sequence to a vector representation that better encapsulates its relationship relative to all preceding tokens in the sequence, along with the token itself.

$$\begin{array}{c}
 \begin{array}{ccccc}
 \mathbf{q}_0 \cdot \mathbf{k}_0 & \mathbf{q}_0 \cdot \mathbf{k}_1 & \mathbf{q}_0 \cdot \mathbf{k}_2 & \mathbf{q}_0 \cdot \mathbf{k}_3 & \mathbf{q}_0 \cdot \mathbf{k}_4 \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} \\
 \mathbf{q}_1 \cdot \mathbf{k}_0 & \mathbf{q}_1 \cdot \mathbf{k}_1 & \mathbf{q}_1 \cdot \mathbf{k}_2 & \mathbf{q}_1 \cdot \mathbf{k}_3 & \mathbf{q}_1 \cdot \mathbf{k}_4 \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} \\
 \mathbf{q}_2 \cdot \mathbf{k}_0 & \mathbf{q}_2 \cdot \mathbf{k}_1 & \mathbf{q}_2 \cdot \mathbf{k}_2 & \mathbf{q}_2 \cdot \mathbf{k}_3 & \mathbf{q}_2 \cdot \mathbf{k}_4 \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} \\
 \mathbf{q}_3 \cdot \mathbf{k}_0 & \mathbf{q}_3 \cdot \mathbf{k}_1 & \mathbf{q}_3 \cdot \mathbf{k}_2 & \mathbf{q}_3 \cdot \mathbf{k}_3 & \mathbf{q}_3 \cdot \mathbf{k}_4 \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} \\
 \mathbf{q}_4 \cdot \mathbf{k}_0 & \mathbf{q}_4 \cdot \mathbf{k}_1 & \mathbf{q}_4 \cdot \mathbf{k}_2 & \mathbf{q}_4 \cdot \mathbf{k}_3 & \mathbf{q}_4 \cdot \mathbf{k}_4 \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)}
 \end{array} \\
 \mathbf{QK}^T \\
 \sqrt{(d_k)}
 \end{array}
 +
 \begin{array}{ccccc}
 0 & -\infty & -\infty & -\infty & -\infty \\
 0 & 0 & -\infty & -\infty & -\infty \\
 0 & 0 & 0 & -\infty & -\infty \\
 0 & 0 & 0 & 0 & -\infty \\
 0 & 0 & 0 & 0 & 0
 \end{array}
 =
 \begin{array}{ccccc}
 \mathbf{q}_0 \cdot \mathbf{k}_0 & -\infty & -\infty & -\infty & -\infty \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & -\infty & -\infty & -\infty \\
 \mathbf{q}_1 \cdot \mathbf{k}_0 & \mathbf{q}_1 \cdot \mathbf{k}_1 & -\infty & -\infty & -\infty \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & -\infty & -\infty \\
 \mathbf{q}_2 \cdot \mathbf{k}_0 & \mathbf{q}_2 \cdot \mathbf{k}_1 & \mathbf{q}_2 \cdot \mathbf{k}_2 & -\infty & -\infty \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & -\infty \\
 \mathbf{q}_3 \cdot \mathbf{k}_0 & \mathbf{q}_3 \cdot \mathbf{k}_1 & \mathbf{q}_3 \cdot \mathbf{k}_2 & \mathbf{q}_3 \cdot \mathbf{k}_3 & -\infty \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} \\
 \mathbf{q}_4 \cdot \mathbf{k}_0 & \mathbf{q}_4 \cdot \mathbf{k}_1 & \mathbf{q}_4 \cdot \mathbf{k}_2 & \mathbf{q}_4 \cdot \mathbf{k}_3 & \mathbf{q}_4 \cdot \mathbf{k}_4 \\
 \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)} & \sqrt{(d_k)}
 \end{array}$$

Figure 4.3: Masked Scaled Dot-Product Attention

The third multi-head self-attention mechanism is also in the decoding component. However, it is provided two sequences as its input: $\mathbf{A}$ denotes the output of the previous sub-layer in the decoder layer, and $\mathbf{B}$ denotes the output of the encoding component, therefore, $\mathbf{A} \neq \mathbf{B}$. This means that the role of this multi-head self-attention mechanism is to transform each token in the target sequence to a vector representation that better encapsulates its relationship relative to all tokens in the source sequence. It is what enables the Transformer to condition each token in the target sequence on the source sequence.

4.4.4 Feed-Forward Neural Network

Recall that the output of each sub-layer in the encoding and decoding components of the Transformer is a sequence of vectors, such that each vector is of dimension d . The feed-forward neural network in both components of the Transformer is an identical simple, position-wise, fully-connected feed-forward neural network that processes each vector in the sequence separately.

¹¹The sequence used to produce the sequence of query vectors.

Each vector in the sequence represents a token in the subword sequence representation of a sentence. The feed-forward neural network applies a non-linear transformation on each vector to generate a new vector representation of the token that maximises the likelihood of observing a target sentence given the corresponding source sentence. The architecture of the feed-forward neural network is defined as follows:

$$o(\mathbf{v}) = [\sigma(\mathbf{v}\mathbf{W}_1 + \mathbf{b}_1)]\mathbf{W}_2 + \mathbf{b}_2 \quad (4.17)$$

where, $o(\cdot) \in \mathbb{R}^d$ is the output of the feed-forward neural network given a vector, $\mathbf{v} \in \mathbb{R}^d$, as input; $\sigma(\cdot)$ is a ReLU activation function; $\mathbf{W}_1 \in \mathbb{R}^{d \times d_{ff}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{ff} \times d}$ are the weight matrices; and $\mathbf{b}_1 \in \mathbb{R}^{d_{ff}}$ and $\mathbf{b}_2 \in \mathbb{R}^d$ are the bias vectors.

The role of the multi-head self-attention mechanism is to identify the relationship between the vectors in a sequence representation of a sentence. The feed-forward neural network identifies patterns within the vectors used to represent tokens in the sequence in order to generate more appropriate representations of the tokens. This action supports the multi-head self-attention mechanism's objective aimed at appropriately capturing how the tokens in a subword sequence relate to one another.

4.4.5 Output Probabilities

When the Transformer is given a source sentence to translate, it estimates the probability of the next token in the target sequence. Meaning that for each position of the target sequence, the model estimates $P(\mathbf{y}_j | \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{j-1}\}; \mathbf{X})$ for $j \in \{1, \dots, q\}$.

The Transformer computes each output probability, $P(\mathbf{y}_j | \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{j-1}\}; \mathbf{X})$, by applying a linear transformation on the j^{th} vector of the decoding component's output sequence, resulting in a vector of dimension $|\mathcal{V}|$; and applying a softmax function to the vector. Each position in the resultant vector corresponds to the probability of observing each token in the vocabulary in the j^{th} position of the target sequence. The token with the highest probability is the most probable token for the j^{th} position of the target sequence.

In addition, Vaswani et al. (2023) propose using the same weight matrix that is used to generate embedding representations for both source and target sentences, for the linear transformation that converts the vectors in the decoding component's output sequence to vectors of dimension $|\mathcal{V}|^{12}$. If the weight matrix in the embedding layer is $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$, then weight matrix for the linear transformation is $\mathbf{W}^T \in \mathbb{R}^{d \times |\mathcal{V}|}$. This weight sharing strategy is referred to as Three-way Weight Tying (Press and Wolf, 2017), and it was introduced as a way to significantly reduce the number of parameters in an NMT model without eroding translation performance.

4.4.6 Add and Norm

The output of each sub-layer in the encoder and decoder layers is modified before being passed to a subsequent sub-layer. Vaswani et al. (2023) proposes applying dropout (Srivastava et al., 2014), a residual connection (He et al., 2015), and layer normalization (Ba et al., 2016) in this order to the output.

Dropout:

Due to the large number of parameters in the Transformer, it is vulnerable to over-fitting sentences in the training corpus, especially in low-resource data settings (Srivastava et al., 2014). It is also vulnerable to complex co-adaptation, which means that the utility of certain parameters in the model are dependent on subsets of other parameters in the model (Hinton et al., 2012). Dropout (Srivastava et al., 2014) attempts to address these issues by introducing an efficient way of approximating the process of sampling and combining exponentially many different Transformers.

In the Transformer, dropout is applied to the output of each sub-layer in the encoder and decoder layers, as well as subsequent to the summation of the embedding representations and positional encodings in the two embedding layers of the model (Vaswani et al., 2023). In both cases, dropout is applied to a sequence of vectors. During the training of the Transformer, for each forward pass, elements in the vectors are mapped to zero with probability $1 - p$. Each instance of dropout, with each forward pass, is equivalent to sampling a sub-network of the Transformer. The process

¹²Without the scaling factor of $\sqrt{d}$.

of combining these sub-networks during testing, however, is approximated by applying a single optimised Transformer with no dropout, such that its parameters are re-scaled by a factor of p .

The effect of dropout is that it prevents the Transformer from becoming too dependent on certain parameters in the model to make predictions. It also encourages the remaining parameters in the sub-network of a Transformer to learn how to make predictions without relying on a specific set of parameters, that is that it minimises co-adaptation. Finally, by combining the sub-networks during tests, dropout helps strengthen the model's capacity to generalise.

Residual Connection:

In general, simply stacking multiple layers in a deep neural network achieves better accuracy over a more shallow network (He et al., 2015). A similar improvement is observed in NMT: Sutskever et al. (2014) and Wu et al. (2016) stack multiple Long Short-Term Memory (LSTM) networks in the encoding and decoding components of the model, and are able to improve the translation performance over more shallow NMT models. However, these LSTM networks can only be stacked up to a certain point before the improvements to model performance slow down, and the model becomes too difficult to train (Wu et al., 2016).

He et al. (2015) report that while increasing the depth of a neural network makes it vulnerable to overfitting; there comes a point when the increasing depth causes training accuracy to degrade. This is attributed to a problem with how the network learns, and residual connections (He et al., 2015) are introduced as a computationally efficient way to address this problem.

If a stack of non-linear layers in a neural network can approximate an underlying mapping $\mathcal{H}(\mathbf{x})$, where $\mathbf{x}$ is the input into these layers, then it follows that the same stack of layers can also approximate a residual mapping that is defined as $\mathcal{F}(x) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$ (He et al., 2015). Residual connections impose that the stack of non-linear layers instead approximate $\mathcal{F}(x)$, and its output is adjusted by summing the output of the stack of non-linear layers with its corresponding input, that is $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ (He et al., 2015). This simple re-formulation improves the gradient flow in deeper networks, and in turn increases the ease of training these models (He et al., 2015; Wu et al., 2016).

Layer Normalization:

Layer normalization (Ba et al., 2016) is introduced as a technique to normalise the neurons in a hidden layer of a neural network. It helps stabilise the training of a network, as its size and depth increases (Ba et al., 2016). By computing individual layer statistics, layer normalization eliminates the batch-dependent constraints observed in batch normalization (Ioffe and Szegedy, 2015). Ba et al. (2016) also show that this normalization technique works effectively in RNNs, improving both the training times and generalisation capabilities of these networks.

Vaswani et al. (2023) propose applying layer normalization after applying the residual connection to the output sequence of each sub-layer, in the encoding and decoding components of the Transformer. For each vector in the output sequence, $\{\mathbf{v}_1, \dots, \mathbf{v}_t\}$, layer normalization statistics are computed over all elements in the vector, $\mathbf{v}_i \in \mathbb{R}^d$:

$$\mu_i = \frac{1}{d} \sum_{f=1}^d \mathbf{v}_{(i,f)} \quad \sigma_i^2 = \frac{1}{d} \sum_{f=1}^d (\mathbf{v}_{(i,f)} - \mu_i)^2 \quad i \in \{1, \dots, t\} \quad (4.18)$$

Using the statistics computed for each vector in the sequence, layer normalization normalizes a vector and adjusts it via gain, $\mathbf{g} \in \mathbb{R}^d$, and bias, $\mathbf{b} \in \mathbb{R}^d$, parameters that are optimised during model training:

$$\mathbf{v}_i^* = \frac{\mathbf{g}}{\sigma_i} \odot (\mathbf{v}_i - \mu_i) + \mathbf{b}, \quad (4.19)$$

where, $\odot$ denotes element-wise multiplication.

4.4.7 Training

The goal of training is to find the set of parameters, $\boldsymbol{\theta}$, for the Transformer that maximise the likelihood of observing each target sentence in the training corpus, $\mathcal{D}$, given its corresponding source sentence, computed across all sentence pairs in the corpus:

$$\mathcal{L}(X, Y; \boldsymbol{\theta}) = \prod_{n=1}^{|\mathcal{D}|} P(Y^{(n)} | X^{(n)}, \boldsymbol{\theta}) \quad (4.20)$$

In practise, the process of searching for the optimal set of parameters begins with parameter initialisation, where parameters in the Transformer are assigned initial values. A loss function is then defined. Given the model’s current set of parameters, the loss function evaluates how well a translation generated by the model matches the target sentence in the corpus. The goal is to minimise the loss function. A gradient-based optimisation algorithm is then applied to iteratively search for the set of parameters that minimise the loss function. Given the size of the Transformer, an optimisation algorithm is chosen based on its ability to simplify the training of the model.

Parameter Initialisation

Parameter initialisation involves assigning initial values to parameters in the Transformer to the start the process of training with. This thesis makes use of three different initialisation techniques for the different parameters in the Transformer, namely constant initialisation, normal initialisation, and Xavier (Glorot) uniform initialisation (Glorot and Bengio, 2010):

- Constant initialisation assigns pre-defined values to each parameter in the model;
- Normal initialisation assigns values that have been sampled from a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ, σ^2 are pre-defined mean and variance parameters; and
- Xavier (Glorot) uniform initialisation is typically applied to the weight parameters of a model that are responsible for transforming vectors from one dimension, `fan_in`, to another, `fan_out`. The technique assigns values to the elements in a weight matrix that have been sampled from the following uniform distribution (Glorot and Bengio, 2010):

$$\mathcal{U}\left[-\frac{\sqrt{6}}{\sqrt{\text{fan_in} + \text{fan_out}}}, \frac{\sqrt{6}}{\sqrt{\text{fan_in} + \text{fan_out}}}\right].$$

Vaswani et al. (2023) do not provide details regarding which techniques are applied to initialise parameters in the Transformer; therefore, this thesis applies the techniques used in the implementation of the Transformer in the Fairseq toolkit (Ott et al., 2019). Weight matrices in the embedding layers and the decoding component’s output linear layer are initialised using the normal initialisation technique, and the remaining weight matrices in the Transformer use the Xavier uniform initialisation. The bias vectors in the feed-forward neural network layer, and the gain and bias vectors used in layer normalization are initialised using the constant initialisation technique.

Loss Function

The next step of training a Transformer is defining a loss function. Translation is categorised as a classification task: the Transformer predicts the probability of observing the next token (a class) in a subword sequence representation of a target sentence. The tokens in each target sequence are chosen from a vocabulary, $\mathcal{V}$ (the set of classes). Therefore, the Multi-class Cross-Entropy (MCE) is an appropriate loss function to evaluate the differences between the predicted and true target sequences. MCE defines a separate loss for each token, $\mathbf{y}_j \in \mathbb{R}^{|\mathcal{V}|}$, in the target sequence as follows:

$$L_{\mathbf{y}_j} = - \sum_{f=1}^{|\mathcal{V}|} y_{j,f} [\log P(y_{j,f})], \quad (4.21)$$

where, $y_{j,f} \in \{0, 1\}$ is the f -th entry in the vector, $\mathbf{y}_j$. Recall that when the Transformer estimates the probability of each token in the target sequence it outputs a vector of probabilities, $P(\mathbf{y}_j | \mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{j-1}, \mathbf{X}) \in \mathbb{R}^{|\mathcal{V}|}$; therefore, each entry in the vector, denoted as $P(y_{j,f})$, is the probability that the token $\mathbf{y}_j$ is classified as f -th token in the vocabulary, $\mathcal{V}$.

What is label smoothing? The likelihood function of the Transformer is transformed into its log-

likelihood form, and decomposed as follows:

$$\begin{aligned}\mathcal{L}(X, Y; \boldsymbol{\theta}) &= \prod_{n=1}^{|\mathcal{D}|} P(Y^{(n)} | X^{(n)}, \boldsymbol{\theta}) \\ &= \prod_{n=1}^{|\mathcal{D}|} P(\mathbf{Y}^{(n)} | \mathbf{X}^{(n)}, \boldsymbol{\theta})\end{aligned}\quad (4.22)$$

$$\begin{aligned}\ell(X, Y; \boldsymbol{\theta}) &= \log \left(\prod_{n=1}^{|\mathcal{D}|} P(\mathbf{Y}^{(n)} | \mathbf{X}^{(n)}, \boldsymbol{\theta}) \right) \\ &= \sum_{n=1}^{|\mathcal{D}|} \log P(\mathbf{Y}^{(n)} | \mathbf{X}^{(n)}, \boldsymbol{\theta}) \\ &= \sum_{n=1}^{|\mathcal{D}|} \log \left(\prod_{j=1}^{q^{(n)}} P(\mathbf{y}_j^{(n)} | \{\mathbf{y}_0^{(n)}, \mathbf{y}_1^{(n)}, \dots, \mathbf{y}_{j-1}^{(n)}\}, \mathbf{X}, \boldsymbol{\theta}) \right) \quad (\text{Equation 4.13}) \\ &= \sum_{n=1}^{|\mathcal{D}|} \sum_{j=1}^{q^{(n)}} \log P(\mathbf{y}_j^{(n)} | \{\mathbf{y}_0^{(n)}, \mathbf{y}_1^{(n)}, \dots, \mathbf{y}_{j-1}^{(n)}\}, \mathbf{X}, \boldsymbol{\theta})\end{aligned}\quad (4.23)$$

As a result, the goal of training the Transformer is then adapted to finding the set of parameters, $\boldsymbol{\theta}$, that maximise the log-likelihood, $\ell(X, Y; \boldsymbol{\theta})$. This formulation means that for each position in the target sequence, the model is trained to assign very large probabilities to the token in the vocabulary that matches $\mathbf{y}_j^{(n)}$. However, this can lead to the model over-fitting on the training corpus (Szegedy et al., 2015). This is especially harmful, if the training corpus contains incorrect translations. To remedy this, Vaswani et al. (2023) propose applying label smoothing (Szegedy et al., 2015) as a way to inject noise into the tokens of the target sequence representations in the training corpus, and in turn reduce over-fitting.

Label smoothing (Szegedy et al., 2015) is a technique that adjusts the one-hot encodings of tokens in the target sequence representation in the training corpus, such that the non-zero entry is replaced with $(1 - \epsilon) + \frac{\epsilon}{|\mathcal{V}|}$, and the zero entries are replaced with $\frac{\epsilon}{|\mathcal{V}|}$ for some small pre-defined value ϵ . It then follows that the loss for each token, $\mathbf{y}_j \in \mathbb{R}^{|\mathcal{V}|}$, in the target sequence becomes:

$$L_{\mathbf{y}_j} = - \sum_{f=1}^{|\mathcal{V}|} \left((1 - \epsilon)y_{j,f} + \frac{\epsilon}{|\mathcal{V}|} \right) \log P(y_{j,f}), \quad (4.24)$$

and loss computed over the entire training corpus becomes:

$$L_{\mathcal{D}} = - \frac{1}{|\mathcal{D}|} \sum_{n=1}^{|\mathcal{D}|} \sum_{j=1}^{q^{(n)}} \sum_{f=1}^{|\mathcal{V}|} \left((1 - \epsilon)y_{j,f}^{(n)} + \frac{\epsilon}{|\mathcal{V}|} \right) \log P(y_{j,f}^{(n)}). \quad (4.25)$$

Optimiser

The final step of training a Transformer is applying an optimisation algorithm. Following from Vaswani et al. (2023), this thesis uses a computationally efficient, gradient-based optimisation algorithm, called Adam (Kingma and Ba, 2017). The name of this optimisation algorithm is derived from adaptive moment estimation: it computes adaptive learning rates for each parameter in the model using estimates of the mean and variance of the gradients that have been derived from the first- and second-order moments (Kingma and Ba, 2017).

Let $L_{\mathcal{D}}(\boldsymbol{\theta})$ be the loss computed over the entire training corpus, the goal of Adam is to search for parameters, $\boldsymbol{\theta}$, that minimise the loss function. For each time step $t = 1, \dots, T$, we use the parameter update found at time step $t - 1$ to compute the gradient of the loss function, denoted as $\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} L_{\mathcal{D}}(\boldsymbol{\theta}_{t-1})$.

Similar to gradient descent, Adam iteratively updates the parameters in the model. However, in practice, (i.) the loss is computed for a mini-batch, $\mathcal{B}$, as opposed to the entire training corpus; and (ii.) at each iteration, the algorithm keeps track of the exponential moving averages of the

gradients, $\mathbf{m}_t$, and the squared gradients, $\mathbf{v}_t$:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t \quad (\text{mean}) \quad (4.26)$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2, \quad (\text{uncentred variance}) \quad (4.27)$$

where, $\beta_1, \beta_2 \in [0, 1)$ are hyperparameters controlling the decaying rates of the moving averages, and $\mathbf{g}_t^2$ applies element-wise multiplication.

How do we implement Adam? The algorithm is initialised at time step $t = 0$, $\mathbf{m}_t$ and $\mathbf{v}_t$ are set to vectors of zeros, and the initial set of parameters is denoted as $\boldsymbol{\theta}_0$. For each iteration $t = 1, 2, \dots$, the following steps are carried out:

1. Evaluate the gradient of the loss function using the parameters of the model that were computed in a previous time step:

$$\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} L_{\mathcal{B}}(\boldsymbol{\theta}_{t-1}); \quad (4.28)$$

2. Update the estimates of the mean, $\mathbf{m}_t$, and variance, $\mathbf{v}_t$, of the gradient using Equations 4.26 and 4.27.
3. Since the estimates of the mean and variance of the gradient were both initialised as vectors of zeros, they are bias towards to zero; therefore, the estimates need to be corrected for bias, where β_1^t and β_2^t both denote β_1 and β_2 to the power of t , respectively:

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t} \quad (4.29)$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}. \quad (4.30)$$

4. Using the estimates of the bias-corrected mean and variance of the gradient, the parameters in the model can be updated, where α denotes is a pre-defined learning rate:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}. \quad (4.31)$$

How do we define the learning rate, α ? Following from the work of [Vaswani et al. \(2023\)](#), instead of defining the learning rate as a fixed scalar value, this thesis defines a learning rate schedule that determines how the learning rate will shift over the duration of training:

1. For the first `warmup_steps` training steps, this schedule allows the the learning rate to increase linearly; then
2. It starts to decrease proportionally to the square root of the training step number, `step_num`:

$$\alpha = \frac{\min(\text{step_num}^{-0.5}, \text{step_num} \cdot \text{warmup_steps}^{-1.5})}{\sqrt{d}}, \quad (4.32)$$

where, d is the dimension of the vectors in the Transformer model.

What is backpropagation? It worth recalling that similar to many other machine learning tasks, the gradients in a model are computed using backpropagation ([Nielsen, 2015](#)). This algorithm begins by computing a forward pass that generates outputs from the Transformer, given source sentences in a mini-batch and the set of model parameters that were computed in a previous time step, $\boldsymbol{\theta}_{t-1}$. Using the output from the forward pass, backpropagation evaluates the loss function, $L_{\mathcal{B}}(\boldsymbol{\theta}_{t-1})$, and performs a backward pass that relies on the chain-rule to compute partial derivatives of the current loss function with respect to the parameters in the model, $\boldsymbol{\theta}$. These partial derivatives are in fact the gradients, $\mathbf{g}_t$.

4.4.8 Evaluation

During training, the Transformer prioritises parallelisation; however, for inference it incorporates some sequential computing in the decoding component in order to predict tokens in the target sequence. The Transformers in this thesis will be evaluated on test and validation corpora during

inference. The translation performance of these models will be evaluated against the BLEU score metric (Papineni et al., 2002). This score is an automatic, language-independent evaluation metric commonly used in MT. In the absence of professional translators to determine whether an increase in the BLEU score reflects a true difference in the models' translation quality, paired statistical significance tests will also be conducted between models. The statistics required to perform these tests are estimated using the paired bootstrap resampling method (Koehn, 2004).

Inference

During inference, similar to training, the encoding component takes a source sequence, where each token in the sequence is encoded as a one-hot vector, as input. It then outputs a sequence of vectors that captures how the tokens in the source sequence relate to one another.

The decoding component during inference, however, does not have access to the corresponding target sequence. Therefore, using the beginning of sentence token, together with the output of the encoding component it will predict the first token, $\hat{y}_1$, in the target sequence. For each time step $t \in \{1, 2, \dots\}$, the predicted token, $\hat{y}_t$ is then fed back into the decoding component as input, and together with the output of the encoding component, it will predict the next token, $\hat{y}_{t+1}$, in the target sequence. This iterative step is repeated until (i.) a special end-of-sentence token $\langle \text{eos} \rangle$ is predicted, or (ii.) the length of the predicted target sequence exceeds a pre-defined maximum sequence length.

The goal of the Transformer during inference is predict the most probable translation according to Equation 4.13:

$$\hat{\mathbf{Y}} = \underset{\mathbf{Y}}{\operatorname{argmax}} P(\mathbf{Y}|\mathbf{X}). \quad (4.33)$$

To do this, the Transformer uses a beam search to perform an efficient search for the most probable translation (Sutskever et al., 2014). At each time step, the decoder keeps track of b partial hypotheses. A partial hypothesis is a sequence of the first few tokens in a potential translation of the source sentence. At each time step, every token in the vocabulary is appended to each partial hypothesis, the beam only retains the b most probable partial hypotheses. When (i.) an end-of-sentence token is appended to a partial hypothesis, or (ii.) its length exceeds the pre-defined maximum sequence length, then it is moved to a set of completed translations, $\mathcal{C}(Y)$. The decoder outputs a translation from the set of completed translations that satisfies Equation 4.33:

$$\hat{\mathbf{Y}} = \underset{\mathbf{Y} \in \mathcal{C}(Y)}{\operatorname{argmax}} P(\mathbf{Y}|\mathbf{X}). \quad (4.34)$$

BLEU

The BLEU score (Papineni et al., 2002) is a commonly used evaluation metric for MT. An n -gram is a sequence of n consecutive words in a sentence. The goal of the BLEU is to provide a sophisticated count of the number of n -grams in a model's predicted translation that match n -grams found in a reference translation. The reference translation is a translation generated by a professional translator. The more matches found between the two translations, the higher the BLEU score.

The process of counting matching n -grams is equivalent to computing a precision metric. The BLEU score in particular, computes a modified n -gram precision metric over the entire corpus for n -grams of length $n \in \{1, \dots, N\}$:

$$p_n = \frac{\sum_{\hat{\mathbf{Y}} \in \mathcal{C}} \sum_{g \in \mathcal{G}(\hat{\mathbf{Y}})} \operatorname{count}_{clip}(g)}{\sum_{\hat{\mathbf{Y}} \in \mathcal{C}} \sum_{g' \in \mathcal{G}(\hat{\mathbf{Y}})} \operatorname{count}(g')} \quad n \in \{1, \dots, N\}, \quad (4.35)$$

where, $\hat{\mathbf{Y}} \in \mathcal{C}$ is a model predicted translation from the set of model predicted translations, $\mathcal{C}$, of sentences in an evaluation corpus. For translation, $\hat{\mathbf{Y}}$, $\mathcal{G}(\hat{\mathbf{Y}})$ is the set of unique n -grams, $g \in \mathcal{G}(\hat{\mathbf{Y}})$, in the sentence. For each n -gram, $g \in \mathcal{G}(\hat{\mathbf{Y}})$, we compute $\operatorname{count}_{clip}(g)$, a measure of the total number of valid times the n -gram g occurs in the predicted translation, relative to the reference translation:

$$\operatorname{count}_{clip}(g) = \min(\operatorname{count}, \max_ref_count), \quad (4.36)$$

where, $\max_ref_count$ is the maximum number of times the g appears in a single reference translation, and count is the number of times it appears in the predicted translation.

The denominator of the modified precision metric, p_n , is the sum of the total number of times each n -gram, $g' \in \mathcal{G}(\hat{\mathbf{Y}})$, appears in the predicted translation, $\hat{\mathbf{Y}}$, computed for all predicted translations in the set, $\mathcal{C}$.

How to combine different p_n metrics across different n -grams? When $p_n = p_1$, the modified precision metric measures how many matching words the predicted translations are able to generate accurately, no matter the order. However, when $p_n = p_{>1}$, the metric measures how fluent the predicted translations are, by computing how many matching sequences of n consecutive words the predicted translations are able to accurately generate. The BLEU score combines the modified precision metrics for different n -gram lengths as follows:

$$\sum_{n=1}^N w_n \log p_n, \quad (4.37)$$

What is the brevity penalty? A penalty applied to predicted translations that are too short because they could potentially be incomplete translations. The penalty is computed over the set of all predicted translations, $\mathcal{C}$. Let r denote the effective reference length computed by summing the length of a reference translation that best matches each predicted translation in the set $\mathcal{C}$ ¹³, and c denote the sum of the lengths of all predicted translations in the set $\mathcal{C}$. The brevity penalty is, therefore, given by¹⁴:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{1 - \frac{r}{c}} & \text{if } c \leq r. \end{cases} \quad (4.38)$$

What is the formula for the BLEU score? The BLEU score combines the weighted modified n -gram precision scores with the sentence brevity penalty to compute the score over an entire set of predictions, $\mathcal{C}$, as follows:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^N w_n \log p_n\right) \in [0, 1]. \quad (4.39)$$

Similar to [Papineni et al. \(2002\)](#), this thesis will let $N = 4$ and apply uniform weights $w_n = \frac{1}{N}$ to allow for comparison with work in other publications.

Statistical Significance Testing

A paired statistical significance test is used to determine whether the performance of two translation models on a single evaluation corpus reflects a true difference in the models' translation quality ([Koehn, 2004](#)). In particular, the significance test will help us determine whether the difference in the BLEU scores computed by both models is statistically significant.

This test is dependent on the paired bootstrap resampling method ([Koehn, 2004](#)) to estimate all necessary statistics. Let $\mathbf{A} = \{\mathbf{A}_1, \dots, \mathbf{A}_n\}$ and $\mathbf{B} = \{\mathbf{B}_1, \dots, \mathbf{B}_n\}$ denote the translations generated by models A and B, respectively, of source sentences in an evaluation corpus. The method is initialised by pairing the translations generated by the two models: $\mathcal{C}(\mathbf{A}, \mathbf{B}) = \{(\mathbf{A}_1, \mathbf{B}_1), \dots, (\mathbf{A}_n, \mathbf{B}_n)\}$. It then prepares S new sets, such that each set contains n translation pairs that have been sampled from the set of paired translations, $\mathcal{C}(\mathbf{A}, \mathbf{B})$, with replacement. For each new set, $\mathcal{C}_i(\mathbf{A}, \mathbf{B})$, the paired bootstrap resampling method then computes the BLEU scores for translations generated by model A and translations generated by model B, and outputs the difference, $\text{BLEU}_i(\mathbf{A}) - \text{BLEU}_i(\mathbf{B})$ for $i \in \{1, \dots, S\}$.

Using the BLEU score differences computed for the S sets, we can approximate a distribution of the BLEU score differences between models A and B ([Koehn, 2004](#)). From the distribution, we can obtain estimations of the mean and variance.

What are the hypotheses for the significance test? The difference in BLEU scores are assessed against the following hypotheses in a statistical significance test:

- H_0 : There is no difference between the BLEU scores computed for models A and B ($\mu_A = \mu_B$);

¹³Each candidate translation can correspond to a multiple reference translations generated by different translators.

¹⁴Note that the modified precision metric already penalises translations that are too long. The assumption is that a sentence that is too long either contains words that have been not observed in reference translation, or it contains repeated counts of a correctly translated word; in both cases count_{clip} adjusts for it.

- H_1 : There is a difference between the BLEU scores computed for models A and B ($\mu_A \neq \mu_B$).

Assuming that the null hypothesis is true, a p -value represents the probability that the absolute difference between the BLEU scores on an evaluation corpus computed for models A and B are equal to or more extreme than the observed absolute difference. Therefore, if the p -value is found to be very small: we have sufficient evidence to reject the null hypothesis, and conclude that the translation performance of the two models, with respect to the BLEU score, is statistically different.

4.5 Conclusion

This chapter presented the three approaches to subword tokenization that this thesis is investigating for the translation of English into South African languages, namely standard subword tokenization, subword regularization and a task-specific tokenizer for the target languages. This was followed by a description of the architecture of the NMT model that will be used for translation, that is the Transformer. The chapter concluded with an outline of how the Transformer is trained and evaluated.

The following chapter outlines the experimental design. Given the computational resource and time constraints that the research in this thesis is being conducted in, the chapter will also discuss how the methods presented in this chapter will be implemented in each experiment.

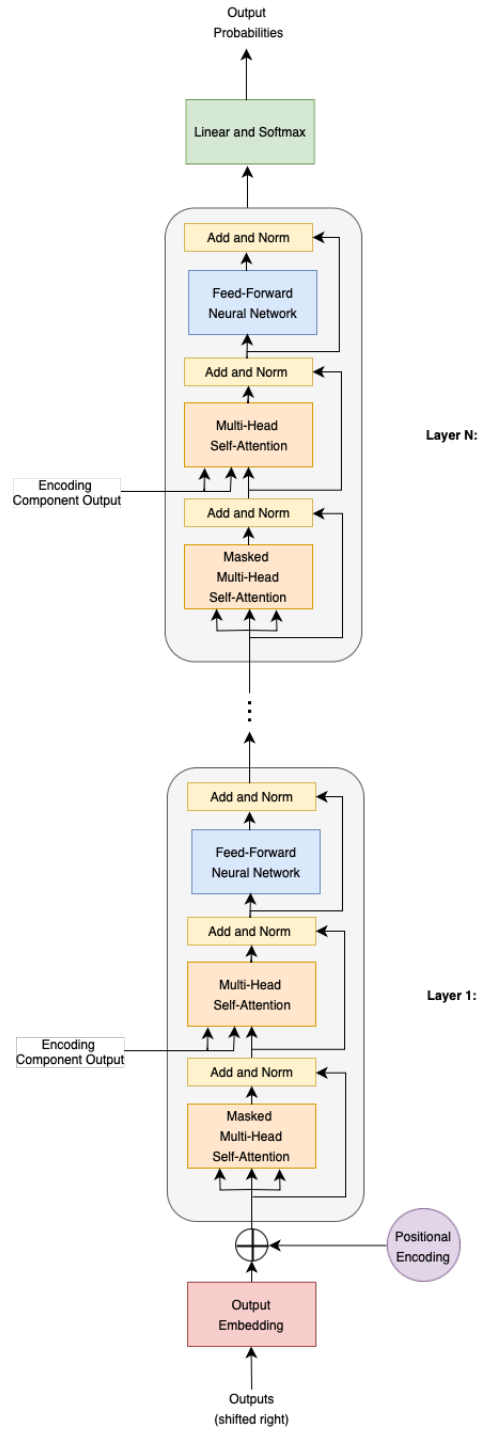


Figure 4.4: Configuration of the decoding component in the Transformer.

Chapter 5

Experimental Setup

This chapter begins with an overview of the experiments and their implementation to address the objectives of this thesis. It includes a description of the hyperparameter settings for all of the NMT models used in each experiment, as well as the hardware and software used to train and deploy the models. The chapter outlines how the different tokenization strategies in each experiment will be implemented to prepare sentences for model training and validation, and it concludes with an outline of how the experiments will be evaluated on the validation and test corpora.

5.1 Experimental Design

To determine whether the translation performance of NMT models trained for the translation of English into Northern Sesotho, isiNdebele and Xitsonga benefit from modifications made to standard subword tokenization algorithms, this thesis will conduct the following three experiments:

1. **Prepare baseline models:** This experiment serves as a benchmark experiment for which all subsequent experiments in this study will be measured against. It involves training and comparing two NMT models for each language pair, with the sentences in each language pair being tokenized using the methods outlined in Chapter 4 on BPE (Sennrich et al., 2016b) and ULM (Kudo, 2018).
2. **Examine the impact of subword regularization on translation performance:** In this experiment, both standard subword tokenization algorithms are modified to perform subword regularization. BPE is modified for subword regularization through BPE-Dropout (Provilkov et al., 2020), and ULM is modified through ULM with subword regularization (Kudo, 2018). In this experiment, we are interested in examining the impact each subword regularization algorithm has on translation performance relative to its corresponding standard subword tokenization algorithm.
3. **Examine impact of a task-specific tokenization strategy on translation performance:** In this final experiment, a neural network-based tokenizer (Hiraoka et al., 2021) is trained under the same framework as ULM to tokenize target sentences. This target language tokenizer, however, leverages the loss function of an NMT model that has the same architecture as the NMT model used for translation to constrain its parameter search during training. In this experiment, we are interested in examining the impact that accounting for translation when developing a target language tokenizer has on translation performance relative to standard ULM.

5.2 Training Setup: Transformer

All the experiments described above use Transformer-based NMT models (Vaswani et al., 2023) for translation. The hyperparameter settings applied in each model are based on the work of van Biljon et al. (2020), and can be found listed in Table 5.1. van Biljon et al. (2020) released a medium-depth Transformer with hyperparameters that have been optimised for low-resource languages. The original implementation of the medium-depth Transformer is available in the following GitHub repository: `optimal_transformer_depth`. It differs from the baseline Transformer (Vaswani et al.,

2023) in terms of depth, that is to say that the medium-depth Transformer has fewer layers in the encoder and decoder stack. It is also more narrow, meaning that it has fewer attention heads, and the output dimension of the embeddings layer and the hidden layer in the feed-forward neural network are all smaller. The medium-depth Transformer uses a higher dropout rate, and a smaller batch size and vocabulary size, all of which have been found to be beneficial in low-resource data settings (Sennrich and Zhang, 2019). Lastly, given that it has fewer parameters to optimise during training, when compared to the baseline Transformer, it requires fewer steps to train. Table 5.1 compares the hyperparameter settings in the baseline Transformer (Vaswani et al., 2023) to the medium-depth Transformer used in this thesis based on the work of van Biljon et al. (2020).

van Biljon et al. (2020) train and evaluate the translation performance of the medium-depth Transformer using parallel corpora that were sourced from the Autshumato Project (Groenewald and Fourie, 2009) for the translation of English into Setswana, Northern Sesotho and Afrikaans. The medium-depth Transformer outperforms baseline Transformers, developed by Martinus and Abbott (2019b), on two out of the three language pairs, and it obtains similar scores on the English-to-Northern Sesotho language pair (van Biljon et al., 2020). The Transformer models in this thesis use similar hyperparameter settings to the models used in van Biljon et al. (2020) because of their promotion of training efficiency, and demonstrated effectiveness when trained in low-resource data settings.

5.2.1 Software

The original implementation of the medium-depth Transformer was developed using the JoeyNMT toolkit (Kreutzer et al., 2019). JoeyNMT is an open-source toolkit for NMT that is built on PyTorch’s machine learning framework. However, this thesis develops its Transformers using the Fairseq toolkit (Ott et al., 2019). Fairseq is a more general open-source toolkit that is used for a number of sequence modelling tasks, including NMT. Similar to JoeyNMT, Fairseq is also built on PyTorch.

We use the Fairseq toolkit because the third experiment in this thesis uses the original implementation for jointly training source and target language tokenizers with a Transformer-based NMT model that was released by Hiraoka et al. (2021). The implementation is developed using Fairseq scripts, and is available in the following GitHub repository: [optok4at/machineTranslation](https://github.com/optok4at/machineTranslation). The third experiment also requires that the NMT model used to fine-tune the target language tokenizer resemble the NMT model used for translation. Since JoeyNMT and Fairseq are targeted for different users, the assumptions driving the architecture of the models, and how these models are implemented differ¹. By using the Fairseq toolkit, we can maintain greater consistency between the NMT model used to fine-tune the target language tokenizer and the model used for translation.

5.2.2 Hardware

All models in this thesis are trained on Google Colab, using a single T4 GPU that is provided by the platform:

- The Transformer models used for **translation** are trained for a total of 100 epochs, and the parameters in each model are updated after every batch. The time taken to train a Transformer across the different experiments ranges between 156 minutes and 233 minutes for the translation of English into isiNdebele; 45 minutes and 67 minutes for the translation of English into Northern Sesotho; and 140 minutes and 191 minutes for the translation of English into Xitsonga.
- The Transformer models used to **fine-tune the target language tokenizer**, in the third experiment, are trained for a total of 30 epochs, and the parameters in each model are updated after every batch. The time taken to jointly train a Transformer with each target language tokenizer ranges between 186 minutes and 708 minutes across the different language pairs.

¹JoeyNMT was developed specifically for beginner NLP researchers, meaning that it uses fewer scripts, and it has a simple codebase to develop common NMT architectures and enable model customisations (Kreutzer et al., 2019); whereas, Fairseq was developed for seasoned NLP researchers and industry professionals, meaning that it has a larger codebase, it supports more and advanced model customisations for a number of sequence modelling tasks, and it supports faster training and inference times (Ott et al., 2019).

Description of Hyperparameter	"Baseline" based on Vaswani et al. (2023)	"Ours" based on van Biljon et al. (2020)
Number of layers in the encoding component	6	3
Number of attention heads in the encoder's multi-head self-attention mechanism	8	4
Dimension of outputs generated by the embedding layer and each sub-layer of the encoder	512	256
Dimension of encoder's feed-forward neural network's hidden layer	2048	1024
Number of layers in the decoding component	6	3
Number of attention heads in the decoder's multi-head self-attention mechanism	8	4
Dimension of outputs generated by the embedding layer and each sub-layer of the decoder	512	256
Dimension of decoder's feed-forward neural network's hidden layer	2048	1024
Batch size	25 000	4096
Joint vocabulary size	37 000	4000
Dropout rate, $1 - p$	0.1	0.25
The hyperparameter for Adam optimiser, (β_1, β_2)	(0.9, 0.98)	(0.9, 0.999)
The hyperparameter for Adam optimiser, ϵ	10^{-9}	10^{-8} *
Number of warmup steps for learning rate scheduler	4000	1000
The hyperparameter for label smoothing, ϵ	0.1	0.1

Table 5.1: Comparison between the hyperparameter configuration for training a baseline Transformer (Vaswani et al., 2023) and the medium-depth Transformer used in this thesis for the translation based on the work of van Biljon et al. (2020). Note that * indicates that we used the Fairseq toolkit's default value.

5.3 Training Setup: Tokenization

For each experiment, this section discusses how the joint vocabulary is constructed for a model, and how sentences in the training and validation corpora are tokenized in preparation for model training and validation.

5.3.1 Experiment 1: Baseline Models

The official implementation of the BPE tokenization algorithm, as described in Sennrich et al. (2016b), is used to construct a joint vocabulary. It is available in the following GitHub repository: [subword-nmt](#). Similarly to van Biljon et al. (2020), the vocabulary is constructed for 4000 merge operations. The official implementation of the ULM tokenization algorithm, as described in Kudo (2018), is also used to construct a joint vocabulary. It is available in the following GitHub repository: [sentencepiece](#). The vocabulary is constructed so that it contains 4000 tokens.

Once the vocabulary is constructed, sentences in the training and validation corpus are mapped to unique subword sequence representations. A command-line in Fairseq, called `fairseq-preprocess`, is applied to the tokenized sentences in the training and validation corpora to ensure that the sentences are in the correct format for model training and validation:

1. From the tokenized sentences in the training corpus, `fairseq-preprocess` is able to extract

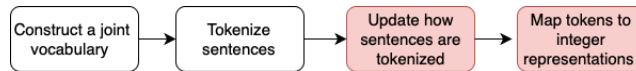


Figure 5.1: Summary of how the standard subword tokenization algorithms are implemented to prepare sentences for the training and validation of the translation models. The red highlighted blocks refer to the steps taken by the `fairseq-preprocess` command-line.

the joint vocabulary;

2. It then applies thresholding to the vocabulary, ensuring that tokens that are not observed at least once on both sides of the training corpus are removed from the vocabulary;
3. Positions in the subword sequence representation of sentences, in the training and validation corpora, that held tokens that have been removed from the vocabulary are mapped to the unknown symbol tokens; and
4. The tokens in the vocabulary are mapped to integer representations based on their index in the vocabulary. These integer representations are equivalent to one-hot vector encodings, in that the integer represents the position of the non-zero entry in the one-hot vector.
5. Output: the tokens in the vocabulary alongside their frequency in the training corpus, saved in a text file; and the integer representation of tokenized sentences in the training and validation corpora, saved in binarised files for efficient storage and data loading purposes.

5.3.2 Experiment 2: Subword Regularization

The implementation of BPE-Dropout in the `subword-nmt` repository is used to generate different subword sequence representations of sentences in the training corpus. The use of this implementation is approved by [Provilkov et al. \(2020\)](#) in the official implementation of the subword regularization algorithm that is available in the following GitHub repository: `BPE-Dropout`. The official implementation of ULM with subword regularization, as described in [Kudo \(2018\)](#), is used to sample different subword sequence representations of sentences in the training corpus. It is available in the following GitHub repository: `sentencepiece`.

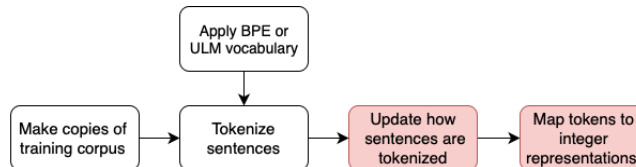


Figure 5.2: Summary of how the subword regularization algorithms are implemented to prepare sentences for the training and validation of the translation models. The red highlighted blocks refer to the steps taken by the `fairseq-preprocess` command-line.

When training an NMT model, both [Provilkov et al. \(2020\)](#) and [Kudo \(2018\)](#) generate a new subword sequence representation of sentences at the beginning of every epoch, or at the beginning of every batch. However, R Sennrich (2021) states on the `subword-nmt` repository’s accompanying README text file that a similar behaviour can be replicated, if multiple copies are made of the training corpus prior to training the model. While this modification introduces a new hyperparameter, that is to say that the number of copies made of the training corpus needs to be decided prior to model training; it also reduces the length of the training times, because the multiple subword sequence representations are now prepared during preprocessing.

Following from this recommendation, 25 copies are made of the training corpus for each language pair as a starting point. The vocabularies constructed by the standard subword tokenization algorithms, prior to any thresholding, are used to generate the different subword sequence representations of sentences in the modified training corpus. The hyperparameter configuration for the subword regularization algorithms are presented below. In both cases, [Provilkov et al. \(2020\)](#) and [Kudo \(2018\)](#) recommend these hyperparameters for low-resource data settings²:

- For BPE-Dropout, we assign a value of 0.1 for the dropout probability, p_{drop} ;

²[Provilkov et al. \(2020\)](#) makes the recommendation in the BPE-Dropout repository.

- For ULM with subword regularization, we assign a value of 0.2 for multinomial distribution’s smoothing parameter, α .

Using the same vocabulary, sentences in the validation corpus are mapped to unique subword sequence representations. Besides the tokenized sentences in the training and validation corpora, the `fairseq-preprocess` command-line is passed an additional input: the vocabulary returned by the `fairseq-preprocess` command-line for each standard subword tokenization algorithm. To ensure that sentences are in the correct format for model training and validation, the command-line performs the following steps:

1. Since thresholding has already been applied to the input vocabulary, positions in the subword sequence representation of sentences, in the training and validation corpora, that held tokens that are not in the vocabulary are mapped to unknown symbol tokens;
2. The tokens in the vocabulary are mapped to integer representations based on their index in the vocabulary;
3. Output: the input vocabulary, saved in a text file; and the integer representation of tokenized sentences in the training and validation corpora, saved in binarised files for efficient storage and data loading purposes.

5.3.3 Experiment 3: Task-Specific Tokenizer

The implementation for jointly training source and target language tokenizers with a Transformer-based NMT model, as described in [Hiraoka et al. \(2021\)](#), is used to train the task-specific tokenizer for the target language. It is available in the following GitHub repository: [optok4at/machineTranslation](#).

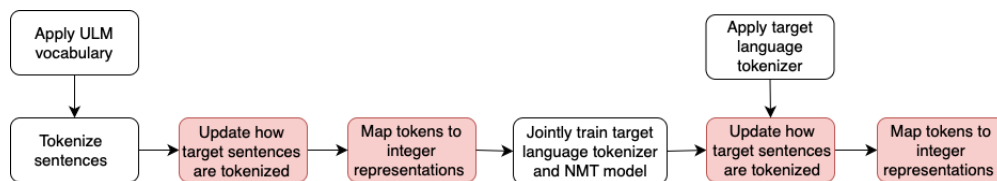


Figure 5.3: Summary of how the task-specific tokenization algorithm is implemented to prepare sentences for the training and validation of the translation models. The red highlighted blocks refer to the steps taken by the `fairseq-preprocess` command-line.

For this experiment, we use the sentences in the training and validation corpora that have been tokenized using ULM to initially train and validate the target language tokenizer. The `fairseq-preprocess` command-line is presented these tokenized sentences, alongside the ULM vocabulary that was returned by the `fairseq-preprocess` command-line in the first experiment:

1. Since thresholding has already been applied to the input vocabulary, positions in the subword sequence representation of sentences, in the training and validation corpora, that held tokens that are not in the vocabulary are mapped to unknown symbol tokens;
2. The tokens in the vocabulary are mapped to integer representations based on their index in the vocabulary;
3. Output: the input vocabulary, saved in a text file; and the integer representation of tokenized sentences in the training and validation corpora, saved in binarised files for efficient storage and data loading purposes.

The [optok4at/machineTranslation](#) repository provides scripts to jointly train a target language tokenizer with a Transformer-based NMT model. The joint model is trained using the output from `fairseq-preprocess` command-line described above. During training, the joint model is provided the original ULM vocabulary, without any thresholding to (i.) pre-train the target language tokenizer, and (ii) sample subword sequence representations of source sentences from. Recall that ULM with subword regularization is applied to source side when training the target language tokenizer. Table 5.2 presents the hyperparameter configuration for the joint model. Letting $\alpha = 0.2$ is based on the recommendation made by [Kudo \(2018\)](#) for sampling subword sequence representations in low-resource data settings.

Description of Hyperparameter	"Ours"
Dimension of target language tokenizer's embedding representations, d_e	64*
Dimension of target language tokenizer's hidden layer, d_f	96*
Number of m -best subword sequence representations sampled on the target side during training, m	3
Multinomial distribution's smoothing parameter for sampling subword sequence representations on the source side during training, α	0.2

Table 5.2: Hyperparameter configuration for target language tokenizer. Note that * indicates that we used the default value in the [optok4at/machineTranslation](#) repository.

The joint model is trained for 30 epochs. During training, parameter updates occur after every batch is processed; however, the model is only evaluated on the validation corpus after every 10 epochs. Once the training is complete, the parameter update in the target language tokenizer that minimises the MCE loss of the joint model is saved, and is used to tokenize target sentences in the training and validation corpora for the training and validation of the translation model. The corresponding source sentences in the training and validation corpora are tokenized using standard ULM. The `fairseq-preprocess` command-line is then presented these tokenized sentences, alongside the ULM vocabulary that was returned by `fairseq-preprocess` command-line in the first experiment to ensure that the sentences are in the correct format to train and validate the translation model:

1. Since thresholding has already been applied to the input vocabulary, positions in the subword sequence representation of sentences, in the training and validation corpora, that held tokens that are not in the vocabulary are mapped to unknown symbol tokens;
2. The tokens in the vocabulary are mapped to integer representations based on their index in the vocabulary;
3. Output: the input vocabulary, saved in a text file; and the integer representation of tokenized sentences in the training and validation corpora, saved in binarised files for efficient storage and data loading purposes.

5.4 Evaluation Setup

Chapter 3 outlines how the raw text files for each language pair are split into a training, validation and test corpora, as well as the steps taken to clean the training and validation corpora. Chapter 4 describes how the different subword tokenization strategies work, and this chapter describes how each of the tokenization strategies are applied to the training and validation corpora in preparation for model training and validation.

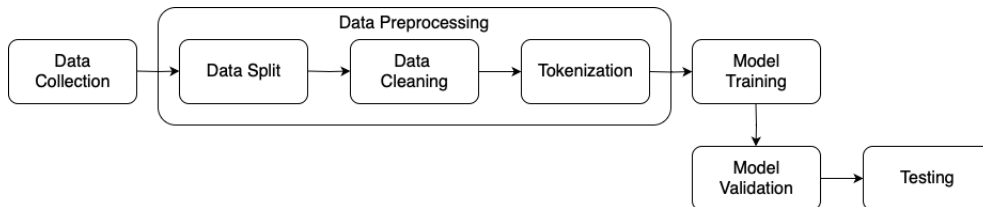


Figure 5.4: Summary of the machine learning pipeline followed to develop the translation models in this thesis.

As mentioned earlier in this chapter, the translation models are trained for 100 epochs. During training, parameter updates occur after every batch is processed; however, each model is only evaluated on a validation corpus after every n_{val} parameter updates, and after every 25 epochs³. For

³ $n_{val} = 2000, 6000, 7000$ for the translation of English into Northern Sesotho, Xitsonga and isiNdebele, respec-

the translation models in the second experiment, recall that 25 copies are made of each training corpus prior to training. To establish a basis for comparison with the baseline experiment, translation models in this experiment are trained for 4 epochs. During training, parameter updates also occur after every batch is processed, and each model is evaluated on the validation corpus after every n_{val} parameter updates, and after every epoch.

Once training is complete, the parameter update in a model resulting in the highest BLEU score on a validation corpus is deployed for further evaluation. While the validation corpus allows us to monitor the behaviour of the model on unseen text during training; the validation corpus, like the training corpus, has been preprocessed, and as such, cannot function as an unbiased corpus to evaluate a model.

We compute the BLEU score of an optimised translation model on the test corpus split. Since no data cleaning is applied on the test corpus, many of the issues observed in the raw text files may still be found in this corpus. This, however, presents an opportunity to evaluate how a translation model handles misaligned or noisy sentence pairs that may contain non-printable characters, or display inconsistent usage of punctuation marks and letter casing.

Finally, we compute the BLEU score of an optimised translation model on an additional corpus provided by [McKellar and Puttkammer \(2020\)](#), referred to as the global test corpus. As mentioned in Chapter 3, this corpus contains 500 source sentences, such that each sentence is aligned to four translations that have been prepared by professional translators. While this corpus is smaller in size, it is more accurate and consistent. Therefore, it presents an opportunity to evaluate whether the translation models were able to learn accurate translations, in spite of the noisy examples in the training corpus that remained after preprocessing. Figure 5.4 summarises the machine learning pipeline that was followed to develop the translation models in this thesis.

Using the BLEU scores computed on the test and global test corpora, we also conduct paired statistical significance tests between the models developed in experiments two and three, with their corresponding baseline model in the first experiment. This is done to determine whether the difference in BLEU scores reflect a true difference in how the tokenization strategies impact the translation performance of the models. The computation of all BLEU scores and significance tests was implemented using the [sacrebleu](#) toolkit ([Post, 2018](#)).

5.5 Conclusion

This chapter provided an overview of how the methods described in Chapter 4 are implemented to address the objectives of this thesis, taking into account the time and computational resource constraints impacting this study. It also outlined how each of the translation models are individually evaluated. The following chapter presents the results for each of the experiments, and discusses these results with respect to the objectives of this thesis.

Chapter 6

Results & Discussion

In this chapter we report and discuss the results that are obtained from conducting the experiments outlined in Chapter 5. For each experiment, we present the BLEU scores for the NMT models deployed for each language pair. The scores are computed on the language pair’s corresponding validation, test and global test corpora. This is followed by a discussion of each experiment, where we analyse each model’s scores across the different evaluation corpora. Paired significance tests are conducted to assess the impact that modifying standard subword tokenization algorithms have on the translation performance of NMT models. The discussion for each experiment considers the conclusions drawn from the significance tests to qualify whether the differences in BLEU scores across the paired models¹ reflects a statistically significant difference in the impact that the tokenization strategies have, and what those differences could mean for the translation of low-resource South African languages.

6.1 Experiment 1: Standard Subword Tokenization

Recall that this experiment is the benchmark that all subsequent experiments in this study are measured against. For each language pair, we will present the BLEU scores for two Transformer-based NMT models:

- A Transformer trained on a corpus that was tokenized via BPE (Sennrich et al., 2016b); and
- A Transformer trained on a corpus that was tokenized via ULM (Kudo, 2018).

6.1.1 English-to-Northern Sesotho

A Transformer was trained on a corpus that was tokenized using BPE for the translation of English into Northern Sesotho. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 19.6 on the validation corpus. Even though this language pair had the smallest training corpus, it achieved BLEU scores of 18.16 and 19.98 on the test and global test corpora, respectively.

A second Transformer was trained on a corpus that was tokenized using ULM for the translation of English into Northern Sesotho. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 19.84(+0.24) on the validation corpus, and scores of 17.85(-0.31) and 19.78(-0.20) on the test and global test corpora, respectively. The values in the brackets reflect the BLEU score difference between the two models.

The Transformer used in Martinus and Abbott (2019b) was evaluated on a test corpus with 3000 sentence pairs, and the same global test corpus used in this study. The evaluation of the Transformer (Martinus and Abbott, 2019b) on the global test corpus was conducted in a later publication (Martinus and Abbott, 2019a). It obtained BLEU scores of 24.16 and 10.94 on a test and the global test corpora, respectively. Both English-to-Northern Sesotho Transformers deployed in this experiment scored higher on the global test corpus than the Transformer in Martinus and Abbott (2019b),

¹NMT model trained on data prepared with standard subword tokenization algorithm versus NMT model trained on data prepared with a modified subword tokenization algorithm.

using a smaller corpus, as a result of the additional data cleaning techniques applied in preprocessing, to train models with a more shallow and narrow architecture. The higher BLEU score on the test corpus in [Martinus and Abbott \(2019b\)](#), relative to its global test corpus score and the test scores of Transformers in this experiment, reveal the propensity of the larger Transformer to overfit on noisy low-resource corpora. For the remainder of this chapter, we will continue to see that the gap between the test and global test scores of the Transformers deployed in this thesis are not as wide as the scores observed in [Martinus and Abbott \(2019b\)](#) and [Martinus and Abbott \(2019a\)](#).

The optimised medium-depth Transformer in [van Biljon et al. \(2020\)](#) is evaluated on a test corpus with 3000 segment pairs, and it achieved a comparable BLEU score to the English-to-Northern Sesotho Transformers in this experiment of 17.6, despite the differences in implementation. Table 6.1 and Table 6.2 each present an example of the translations generated by both Transformers of a sentence from the test and global test corpora, respectively.

Type	Sentence
Source	Registered testing stations may only examine and test motor vehicles in accordance with their grading.
Reference	Ditešene tše di ngwadišitšwego di ka dira diteko tša dikoloi tša bogolo bjoo bo dumeletšwego ka gona.
Prediction (BPE)	Dipeistara tše di ngwadišitšwego di ka dirwa fela gomme tša rekišwa dikoloi go ya ka dikoloi go ya ka difatanaga.
Prediction (ULM)	Diteše tša diteko tše di ngwadišitšwego di ka hlalobja gomme diteko tša diteko tša dikoloi go ya ka mabaka a bona

Table 6.1: An English-to-Northern Sesotho sentence pair from the test corpus alongside the predictions of the source sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.

In general, both Transformers do not generate identical translations to the reference translations provided in the test and global test corpora, as evidenced by the BLEU scores obtained that range between 17.85 and 19.98, out of 100. However, when inspecting the models' predictions, we are able to match words (1-gram) and sequences of words (n -grams) in the predictions with n -grams in the reference translations. This demonstrates the models' capacity to at least translate parts of the source sentences. In some cases when the predicted translation is not identical to the reference translation, we also observe the models' capacity to generate meaningful translations that align with the intent of source sentence. To showcase these key findings, we refer to the example in Table 6.1. We see that the predictions are able to generate a few matching n -grams with the reference, especially in the initial portion of the translation. We also notice that the reference translates "testing stations" as "[d]itešene", which when back-translated means "stations"; whereas, the ULM Transformer translates the source phrase as "[d]iteše tša diteko". While the prediction is not identical to the reference, we can see that the model does try to generate a meaningful translation that aligns with the intent of the source phrase: "[d]iteše" is incorrectly spelled, and should be written as "[d]itešene"; however, we see that the prediction when back-translated attempts to translate the source phrase as the "[t]he stations of the tests".

In the case of the example presented in Table 6.2, we see that the predictions appear to be incomplete translations: the word "charter" is not translated; whereas, the references translate the word to "tšhata" or "lengwalotumelelo". However, we do see that both predictions are able to replicate a few n -grams across the different reference translations.

6.1.2 English-to-isiNdebele

A Transformer was trained on a corpus that was tokenized using BPE for the translation of English into isiNdebele. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 11.59 on the validation corpus. The training corpus used in this case had more than four times the number of sentence pairs than the English-to-Northern Sesotho training corpus, yet it achieved BLEU scores of 10.85 and 10.20 on the test and global test corpora, respectively.

Type	Sentence
Source	We want you, our client, to judge us according to the standard of service we set out in this charter.
References	1. Re nyaka gore wena, modiriši wa rena, go re ahlola ka maemo a tirelo ao re a filego ka mo gare ga tšhata ye. 2. Re nyaka gore wena, moreki wa rena, o re ahlole go ya ka boemo bja tirelo bjoo re bo beago ka go lengwalotumelelo le. 3. Re rata gore wena moreki, o ahlole mošomo wa rena go ya ka maemo a tirelo ao a lego ka gare ga tšhata ye. 4. Re nyaka wena, khastama ya rena, go re ahlola go ya le ka maemo a tirelo yeo re e beyago ka go lengwalotumelelo le.
Prediction (BPE)	Re nyaka, klaemete ya rena, go ya ka maemo a rena go ya ka maemo a tirelo ye re e beilwego ka mo nakong ye.
Prediction (ULM)	Re nyaka gore o itlama badiredi ba rena bao ba nago le maswanedi a go ya ka maemo a tirelo ye re bego re lebane le yona

Table 6.2: An English sentence from the global test corpus alongside the translations generated of the segment in Northern Sesotho by four professional translators, and the predictions of the sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.

A second Transformer was trained on a corpus that was tokenized using ULM for the translation of English into isiNdebele. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 11.97(+0.38) on the validation corpus, and scores of 11.26(+0.41) and 9.59(-0.61) on the test and global test corpora, respectively. The values in the brackets reflect the BLEU score difference between the two models.

According to the accompanying meta data file for this language pair, its raw corpus was only released by the Autshumato Project in 2021; therefore, there are no results from [Martinus and Abbott \(2019b\)](#) and [van Biljon et al. \(2020\)](#) to compare the results of the English-to-isiNdebele Transformers in this experiment with. Table 6.3 and Table 6.4 each present an example of the translations generated by both Transformers of a sentence from the test and global test corpora, respectively.

Type	Sentence
Source	It is important to make sure that each member of the project team has a clear understanding of what they must do and how it relates to the tasks of others .
Reference	Kuqakathekile ukuqinisekisa bona ilunga elinye neline lesiqhema sephrojekthi lizwisisa kuhle okufanele kwenziwe nokobana kunobudlelwana obunjani mayelana nemisebenzi yabanye .
Prediction (BPE)	Kuqakathekile ukwenza isiqiniseko sokobana elinye neline ilunga lesiqhema sephrojekthi linokuzwisisa okucacileko kwalokho abakufanele benze begodu nokobana lihlobana bunjani emisebenzini yabanye.
Prediction (ULM)	Kuqakathekile ukwenza isiqiniseko sokobana elinye neline ilunga lesiqhema sephrojekthi lizwisise kuhle lokho okufanele bakwenze begodu nokobana lihlobana njani nemisebenzi yabanye.

Table 6.3: An English-to-isiNdebele sentence pair from the test corpus alongside the predictions of the source sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.

The low BLEU scores observed for both Transformers indicate that by and large the models do not generate identical translations to the reference translations in the test and global test corpora. However, when inspecting models' predictions, we are able to identify matching root words and shorter n -grams in the predictions and the reference translations. In some cases when the predicted translation is not identical to the reference translation, we also observe the models' capacity to generate meaningful translations that align with the intent of the source segment. The example in Table 6.3 is used to showcase these findings. We see that the predictions are able to generate very few matching 1- and 2-grams; however, there are a number of matching root words. For example, the reference contains the word ["o"+"ku"+"fanele"]; whereas the BPE Transformer generates ["aba"+"ku"+"fanele"]. We also notice that the reference translates the source "to make sure" as "ukuqinisekisa"; whereas, both predictions translate the source phrase as "ukwenza isiqiniseko". While the prediction is not identical to the reference, we can see that the model does try to generate a meaningful translation that aligns with the intent of the source phrase: "ukwenza" means "to make", and "isiqiniseko" means "sure/certainty".

In the case of the example presented in Table 6.4, we see that both predictions are only able to match a few 1-grams across the different reference translations. What we do see a lot of, is the replication of a number root words observed in the reference translations, e.g., ["lo"+"msebenzi"] in the references is predicted as ["le"+"msebenzi"] by the BPE Transformer. We also again observe the models generating a meaningful translation that aligns with intent of the source sentence: the word "client" is translated to "itlayenti" and ["ama"+"tlayenti"] (plural form of "itlayenti") in both predictions, respectively. Both words look like linguistic adaptations of the the English word "client".

Type	Sentence
Source	We want you, our client, to judge us according to the standard of service we set out in this charter.
References	1. Sifuna wena, njengekhastama lethu, bona usahlulele ngezinga lomsebenzi esiwuveza kilomtloomalungelo. 2. Sifuna wena, njengekhastama lethu, usahlulele ngezinga lomsebenzi esiwethulako emtlowenethu. 3. Sifuna wena, mthengi wethu, ukobana usahlulele ngokwezinga lomsebenzi esitole ngawo kileNcwadi yesiVumelwano. 4. Sifuna wena, ikhastama lethu, usijaje ngezinga lomsebenzi wethu esilibeke kiletjhathi.
Prediction (BPE)	Sifuna ukuveza, itlayenti lethu, ukuhlulela ngokuya kwezinga lemisebenzi esiyibekwe kiletjhadi.
Prediction (ULM)	Sifuna bona unikele amatlayenti wethu anikela ijaji ukuya ngokwezinga lomsebenzi esiwubeka kiletjhadi

Table 6.4: An English sentence from the global test corpus alongside the translations generated of the sentence in isiNdebele by four professional translators, and the predictions of the sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.

6.1.3 English-to-Xitsonga

A Transformer was trained on a corpus that was tokenized using BPE for the translation of English into Xitsonga. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 33.97 on the validation corpus. This language pair had the largest training corpus, and it achieved BLEU scores of 34.99 and 29.32 on the test and global test corpus, respectively.

A second Transformer was trained on a corpus that was tokenized using ULM for the translation of English into Xitsonga. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 33.82(-0.15) on the validation corpus, and scores of 34.81(-0.18) and 27.92(-1.40) on the test and global test corpus, respectively. The values in the brackets reflect the BLEU score difference between the two models.

Recall that the Transformer in [Martinus and Abbott \(2019b\)](#) was evaluated on a test corpus with

3000 sentence pairs, and the same global test corpus used in this study in a later paper (Martinus and Abbott, 2019a). For the translation of English into Xitsonga, it obtained BLEU scores of 49.74 and 17.98 on the test and global test corpora, respectively. Similarly to the Northern Sesotho Transformers, both English-to-Xitsonga Transformers deployed in this experiment scored higher on the global test corpus than the Transformer in Martinus and Abbott (2019b), using a smaller corpus, as a result of the additional data cleaning techniques applied in preprocessing, to train a more shallow and narrow architecture. The higher BLEU score on the test corpus in Martinus and Abbott (2019b), relative to its global test corpus score and the test scores of the Transformers in this experiment, again reveal the propensity of the larger Transformer to overfit on noisy low-resource corpora. Table 6.5 and Table 6.6 each present an example of the translations generated by both Transformers of a sentence from the test and global test corpora, respectively.

Type	Sentence
Source	This will play a central role in development cooperation .
Reference	Leswi swi ta va ntirho wa nkoka eka ntirhisano wa swa nhluvukiso .
Prediction (BPE)	Leswi swi ta tlanga xiave xa nkoka eka ntirhisano wa nhluvukiso.
Prediction (ULM)	Leswi swi ta tlanga xiave xa nkoka eka ntirhisano wa nhluvukiso.

Table 6.5: An English-to-Xitsonga sentence pair from the test corpus alongside the predictions of the source sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.

In general, both Transformers do not generate identical translations to the reference translation provided in the test and global corpora, as evidenced by the BLEU scores obtained that range between 27.92 and 34.99, out of 100. However, when inspecting the models' predictions, we are able to match n -grams in the predictions with n -grams in the reference translations. In some cases when the predicted translation is not identical to the reference translation, we also observe the models' capacity to generate meaningful translations that align with the intent of the source segment. The example in Table 6.5 is used to demonstrate these findings. We see that the predictions are able to generate a matching n -grams with the reference. We also notice that unlike the reference, both predictions generate a more direct translation of the source phrase "This will play" as "Leswi swi ta tlanga". The word "tlanga" means "play"; whereas, "ntirho" in the reference means work or function.

In the case of the example presented in Table 6.6, we see that both translations are able to replicate a number n -grams across the different reference translations.

6.1.4 Discussion

In Table 6.7, we present the BLEU scores of the Transformers deployed for each language pair. The scores are computed on each language pair's test and global test corpora. For each BLEU score, a 95% confidence interval ($\mu \pm$ margin of error) is provided. The statistics used to compute the confidence interval are estimated using the paired bootstrap resampling method. As part of this experiment, we also compare the two Transformers for each language pair. To do this, we conduct a paired significance test. The final column in Table 6.7 presents the p -values computed comparing the ULM Transformer to the BPE Transformer. The null hypothesis states that there is no difference in how the two tokenization methods impact the translation performance of the Transformer with respect to the BLEU score. Assuming that the null hypothesis is true, the p -value represents the probability that the absolute difference between the two models' BLEU scores that we computed on a corpus is equal to or more extreme. If the p -value is found to be very small, we have sufficient evidence to reject the null hypothesis, and conclude that the impact ULM has on the translation performance of the Transformer is significantly different from BPE.

When we compare the BPE Transformer with the ULM Transformer, we see that it obtained higher BLEU scores on the test and global test corpora, except for the ULM Transformer trained for the translation of English into isiNdebele. That model obtained a lower score on the test corpus. The very small p -value computed for the paired significance test comparing the ULM Transformer to the BPE Transformer on the English-to-isiNdebele test corpus, indicates that there is sufficient evidence to reject the null hypothesis, and conclude that there is a statistically significant difference

Type	Sentence
Source	We want you, our client, to judge us according to the standard of service we set out in this charter.
References	1. <i>Hi lava wena, khasimende ya hina, ku hi ahlula hi xiyimo xa vukorhokeri lexi hi xi vekeke eka xiyimo xa vukorhokeri lexi kumekaka eka tsalwa leri ra timfanelo.</i> 2. <i>Hi tsakela wena khasimende ya hina ku nyika mavonelo hi xiyimo xa vukorhokeri lebyi hi byi kombiseke eka tsalwa leri.</i> 3. <i>Hi lava leswaku n'wina, tikhasimende ta hina, mi hi avanyisa hi xiyimo xa vukorhokeri bya hina lebyi hi byi simekeke eka Nt-wanano lowu.</i> 4. <i>Hi lava wena, khasimende ya hina, u hi pima hi mipimo ya vukorhokeri leyi hi yi vekaka eka chata leyi.</i>
Prediction (BPE)	Hi lava ku khumba, tlilayente ya hina, ku hi ahlula ku ya hi mpimo wa vukorhokeri lebyi hi vekeke eka filipichati leyi.
Prediction (ULM)	Hi lava leswaku u tipfumelela ku ahlula muavanyisi wa hina ku ya hi mpimo wa vukorhokeri lowu hi wu vekaka eka chati leyi

Table 6.6: An English sentence from the global test corpus alongside the translations generated of the sentence in Xitsonga by four professional translators, and the predictions of the sentence generated by the two Transformer models that were each trained on a corpus that was tokenized using BPE and ULM, respectively.

in how the two models generate translations from the noisy test corpus. From this conclusion, it is reasonable to assume that the higher score of the ULM Transformer on the English-to-isiNdebele test corpus can be attributed to how ULM tokenizes sentences in this corpus.

While the BPE Transformers may have obtained higher BLEU scores on the English-to-isiNdebele global test corpus, as well as on all the other language pairs' test and global test corpora, the large p -values computed for the paired significance tests indicate that we do not have sufficient evidence to reject the null hypothesis. Meaning that there is reason to assume that the two tokenization algorithms do not cause the Transformer to behave differently with respect to the BLEU scores. This is in line with experiments in Kudo (2018), who illustrates in his experiments that the translation performance of NMT models that use BPE and ULM are generally comparable.

The BPE Transformer trained for the translation of English into Xitsonga is an exception. It not only achieved a higher BLEU score on the global test corpus compared the ULM Transformer; a small p -value was also computed for the paired significance test. This indicates that there is sufficient evidence to reject the null hypothesis, and conclude that there is a statistically significant difference in how the two tokenization algorithms impact how the Transformer generates translations on the global test corpus. From this conclusion, we can assume that in the case of the English-to-Xitsonga global test corpus, the way in which ULM tokenized the sentences was not very beneficial to the model's translation performance.

For both the BPE and ULM Transformers, we expected to see the BLEU scores increase, as the size of the training corpus increases. This holds for the English-to-Northern Sesotho and English-to-Xitsonga language pairs. The English-to-isiNdebele language pair has more than four times the number of sentence pairs in its training corpus than the English-to-Northern Sesotho language pair, yet both of its models obtain the lowest BLEU scores compared to all the models deployed in this experiment. The low score is potentially due to the linguistic properties specific to isiNdebele. Recall that isiNdebele is an agglutinating language, such that similarly to Northern Sesotho and Xitsonga, it encodes grammatical meaning through appending prefixes, infixes and suffixes to root/stem words resulting in morphologically dense words in a sentence; unlike English, a predominantly isolating language, characterised by lower density words (Spaull et al., 2020). IsiNdebele also follows a conjunctive writing system, unlike Northern Sesotho, Xitsonga and English; meaning that it combines grammatical components into single words, resulting in shorter sentences, where each word in the sentence is generally much longer (Prinsloo and de Schryver, 2002; Spaull et al., 2020). Therefore, the linguistic complexity of isiNdebele, similarly to other Nguni languages, make it harder to model, especially when modelling the translation into isiNdebele from a language so

linguistically distinct from it, such as English (Nyoni and Bassett, 2021).

However, I would like to posit that if the goal of the BLEU score is to provide a sophisticated count of the number of n -grams in a prediction that match with n -grams found in a reference translation (Papineni et al., 2002); then it places the evaluation of isiNdebele translations at a disadvantage, compared to Northern Sesotho and Xitsonga with their disjunctive writing systems. Disjunctive writing systems are characterised by longer sentences that use a large number of one-syllable words in-between more dense words (Prinsloo and de Schryver, 2002; Spaul et al., 2020). The Transformer has a less difficult time predicting the words with fewer letters in the them. When inspecting the predictions generated by the English-to-isiNdebele Transformers, the models demonstrate a reasonable capacity to at least translate some elements of the source sentence seen in the number of correctly predicted root words, in spite of the very low BLEU scores.

		BPE	ULM
English-to-Northern Sesotho	Test	18.16 (18.15 $\pm$ 0.71)	17.85 (17.85 $\pm$ 0.70) ($p = 0.0739$)
	Global Test	19.98 (19.96 $\pm$ 1.34)	19.78 (19.77 $\pm$ 1.27) ($p = 0.2358$)
English-to-isiNdebele	Test	10.85 (10.85 $\pm$ 0.53)	11.26 (11.26 $\pm$ 0.56) ($p = 0.0280$)*
	Global Test	10.20 (10.18 $\pm$ 1.16)	9.59 (9.56 $\pm$ 1.09) ($p = 0.1039$)
English-to-Xitsonga	Test	34.99 (35.00 $\pm$ 1.18)	34.81 (34.82 $\pm$ 1.15) ($p = 0.2158$)
	Global Test	29.32 (29.32 $\pm$ 1.51)	27.92 (27.92 $\pm$ 1.60) ($p = 0.0110$)*

Table 6.7: The BLEU scores of the Transformers deployed for each language pair in the baseline experiment. Each score includes its 95% confidence interval. The last column contains the p -value for the paired significance tests that were conducted. The asterisk indicates that based on the p -value, the two models were found to behave differently, and the **bold font** indicates that this is best score and it is statistically significant from the baseline model (in this case, the BPE Transformer is treated as the baseline model).

6.2 Experiment 2: Subword Regularization

In this experiment, both standard subword tokenization algorithms were modified to perform subword regularization. The objective of this experiment was to examine the impact that the subword regularization algorithms have on the translation performance of a Transformer-based NMT model, relative to their corresponding standard subword tokenization algorithm. For each language pair, we will present the BLEU scores for two Transformers:

- A Transformer trained on a corpus that was prepared using BPE-Dropout (Provilkov et al., 2020); and
- A Transformer trained on a corpus that was prepared using ULM with subword regularization (Kudo, 2018).

6.2.1 English-to-Northern Sesotho

A Transformer was trained on a corpus that was prepared following BPE-Dropout for the translation of English into Northern Sesotho. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 15.53(-4.07) on the validation corpus, and scores of 14.50(-3.66) and 18.65(-1.33) on the test and global test corpora. The values in the brackets reflect the BLEU score difference between the BPE-Dropout and the BPE Transformer models.

A second Transformer was trained on a corpus that was prepared following ULM with subword regularization for the translation of English into Northern Sesotho. Once training was complete,

the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 11.72(-8.12) on the validation corpus, and scores of 11.22(-6.63) and 14.86(-4.92) on the test and global test corpora. The values in the brackets reflect the BLEU score difference between the ULM with subword regularization and the ULM Transformer models.

Both Transformers deployed in this experiment obtained lower BLEU scores on the validation, test and global test corpora to their counterpart baseline models in Experiment 1. Table F.1 and Table F.2 each present an example of a translation generated by the BPE-Dropout and ULM with subword regularization Transformers of a sentence from the test and global test corpora, respectively. In both tables, we will also get to see how the predicted translations compare to the translations generated by the baseline models.

6.2.2 English-to-isiNdebele

A Transformer was trained on a corpus that was prepared following BPE-Dropout for the translation of English into isiNdebele. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 10.51(-1.08) on the validation corpus, and scores of 9.48(-1.37) and 10.81(+0.61) on the test and global test corpora. The values in the brackets reflect the BLEU score difference between the BPE-Dropout and the BPE Transformer models.

A second Transformer was trained on a corpus that was prepared following ULM tokenization with subword regularization for the translation of English into isiNdebele. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 9.08(-2.89) on the validation corpus, and scores of 8.43(-2.83) and 10.84(+1.25) on the test and global test corpora. The values in the brackets reflect the BLEU score difference between the ULM with subword regularization and the ULM Transformer models.

Both Transformers deployed in this experiment only obtained lower BLEU scores on the validation and test corpora to their counterpart baseline models in Experiment 1. Table F.3 and Table F.4 each present an example of a translation generated by the BPE-Dropout and ULM with subword regularization Transformers of a sentence from the test and global test corpora, respectively. In both tables, we will also get to see how the predicted translations compare to the translations generated by the baseline models.

6.2.3 English-to-Xitsonga

A Transformer was trained on a corpus that was prepared following BPE-Dropout for the translation of English into Xitsonga. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 31.65(-2.32) on the validation corpus, and scores of 31.63(-3.36) and 27.92(-1.40) on the test and global test corpora. The values in the brackets reflect the BLEU score difference between the BPE-Dropout and the BPE Transformer models.

A second Transformer was trained on a corpus that was prepared following ULM tokenization with subword regularization for the translation of English into Xitsonga. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 29.74(-4.08) on the validation corpus, and scores of 29.05(-5.76) and 30.19(+2.27) on the test and global test corpora. The values in the brackets reflect the BLEU score difference between the ULM with subword regularization and the ULM Transformer models.

The BPE-Dropout Transformer obtained lower BLEU scores on the validation, test and global test corpora to its counterpart baseline model in Experiment 1. Whereas, the ULM with subword regularization Transformer only obtained lower BLEU scores on the validation and test corpora to its counterpart baseline model in Experiment 1. Table F.5 and Table F.6 each present an example of a translation generated by the BPE-Dropout and ULM with subword regularization Transformers of a sentence from the test and global test corpora, respectively. In both tables, we will also get to see how the predicted translations compare to the translations generated by the baseline models.

6.2.4 Discussion

In Table 6.8, we present the BLEU scores of the Transformers deployed for each language pair. The scores are computed on each language pair’s test and global test corpora. For each BLEU score, a 95% confidence interval is provided. As part of this experiment, we compare the impact of each subword regularization algorithm on the translation performance of the Transformer for each language pair, relative to its baseline model in Experiment 1. Table 6.8 also presents the p -values computed comparing the BPE-Dropout Transformers to the baseline BPE Transformers, and the ULM with subword regularization Transformers to the baseline ULM Transformers.

When we compare BPE-Dropout Transformers to the BPE Transformers, the BPE-Dropout Transformers obtain lower BLEU scores on the test and global test corpora, except for the Transformer trained for the translation of English into isiNdebele. That BPE-Dropout Transformer model obtained a higher BLEU score on the English-to-isiNdebele global test corpus. However, the large p -value computed for the paired significance test comparing it to the BPE Transformer, indicates that there is insufficient evidence to reject the null hypothesis. Therefore, we can conclude that with respect to the BLEU score, there is insufficient evidence to prove the use of BPE-Dropout had any impact on the translation performance of the Transformer on the English-to-isiNdebele global test corpus.

Not only do the BPE-Dropout Transformers obtain lower BLEU scores on the English-to-isiNdebele test corpus and all other language pairs’ test and global test corpora; the small p -values computed for the paired significance tests comparing the BPE-Dropout Transformers to BPE Transformers indicate that there is sufficient evidence to reject the null hypotheses, and conclude that there is a difference in how BPE-Dropout affects the translation performance of the Transformers, relative to BPE. From this conclusion, it is reasonable to infer that this subword regularization algorithm did not benefit these models with respect to the BLEU score.

When we compare the ULM with subword regularization Transformers with the ULM Transformers, we see that the ULM with subword regularization Transformers obtain lower scores on the all the test corpora. The small p -values computed for the paired significance tests comparing the ULM with subword regularization Transformers to the ULM Transformers indicate that there is sufficient evidence to reject the null hypothesis, and conclude that there is a difference in how ULM with subword regularization affects the translation performance of the Transformer, relative to BPE. From this conclusion, it is reasonable to infer that the modification made via subword regularization did not benefit this model on the test corpus, with respect to the BLEU score.

When we consider the global test corpora, we see that the ULM with subword regularization Transformers obtain higher BLEU scores when compared to the ULM Transformer, except for the Transformer trained for the translation of English into Northern Sesotho. That model obtained a lower BLEU score, and a low p -value was computed for the paired significance test comparing it to the ULM Transformer. Based on this small p -value, we have evidence to suggest that the modification made via subword regularization did not benefit this model with respect to the BLEU score on the global test corpus. However, the modification made to the standard ULM tokenization algorithm did appear to benefit the Transformers for the translation of English into isiNdebele and Xitsonga, they both obtained higher BLEU scores and low p -values for the paired significance tests.

While subword regularization did not always result in higher BLEU scores compared their standard tokenization counterparts, this approach appears to increase the Transformer’s robustness to noisy training examples. In Chapter 3, we highlight the issues observed in the raw corpora that affect its quality. Since no data cleaning is applied to the test corpora, many of those issues can still be observed in the corpora. The global test corpora are of a better quality corpora, despite their sizes. Each global test corpus also has four reference translations, instead of one, for the predictions to be compared across. Both corpora come from the same domain; therefore, if the Transformer is doing a good job learning how to translate from one language into another, it is within reason to expect it obtain a BLEU score on the global test corpus that is similar to or higher than the test corpus. We observe this more often on the Transformers that are trained with subword regularization (5 out of 6 models), compared to the Transformers trained with standard subword tokenization (2 out of 6 models).

It is worth noting that this experiment did not observe the improvements to BLEU scores that are similar to [Kudo \(2018\)](#) and [Provilkov et al. \(2020\)](#) when applying subword regularization methods. However, in both of the publications, the Transformers were allowed to train to convergence. In

this experiment, our Transformers were only allowed to train for 4 epochs on the modified training corpus with 25 copies of the original corpus. This could be a potential reason for the poorer performance. It would also be worth investigating an appropriate number of copies one should make of the training corpus before training the Transformer.

		BPE-Dropout	ULM with SR
English-to-Northern Sesotho	Test	14.50 (14.49 $\pm$ 0.57) ($p = 0.0010$)*	11.22 (11.21 $\pm$ 0.51) ($p = 0.0010$)*
	Global Test	18.65 (18.66 $\pm$ 1.28) ($p = 0.0010$)*	14.86 (14.84 $\pm$ 1.10) ($p = 0.0010$)*
English-to-isiNdebele	Test	9.48 (9.48 $\pm$ 0.50) ($p = 0.0010$)*	8.43 (8.44 $\pm$ 0.49) ($p = 0.0010$)*
	Global Test	10.81 (10.77 $\pm$ 1.14) ($p = 0.0609$)	10.84 (10.81 $\pm$ 1.15) ($p = 0.0170$)*
English-to-Xitsonga	Test	31.63 (31.63 $\pm$ 1.09) ($p = 0.0010$)*	29.05 (29.05 $\pm$ 1.05) ($p = 0.0010$)*
	Global Test	27.92 (27.92 $\pm$ 1.60) ($p = 0.0110$)*	30.19 (30.19 $\pm$ 1.63) ($p = 0.0010$)*

Table 6.8: The BLEU scores of the Transformers deployed for each language pair in the baseline experiment. Each score includes its 95% confidence interval. The last column contains the p -value for the paired significance tests that were conducted. The asterisk indicates that based on the p -value, the two models were found to behave differently, and the **bold font** indicates that this is best score and it is statistically significant from the baseline model in the first experiment.

6.3 Experiment 3: Task-Specific Tokenization

In the final experiment, we train task-specific tokenizers for target languages. The task-specific tokenizer leverages the loss function of an NMT model, whose architecture resembles the NMT model used for translation, to fine-tune how ULM assigns probabilities to tokens in a vocabulary (Hiraoka et al., 2021). The objective of this experiment is to examine how the task-specific tokenizer impacts the translation performance of a Transformer-based NMT model, relative to ULM. For each language pair, we will present the BLEU scores for a Transformer that is trained on a corpus whose target sentences are prepared using the task-specific tokenizer.

6.3.1 English-to-Northern Sesotho

A Transformer was trained on a corpus whose target sentences were tokenized using a task-specific tokenizer for the translation of English into Northern Sesotho. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 19.71(-0.13) on the validation corpus, and scores of 17.92(+0.07) and 20.01(+0.23) on the test and global test corpora, respectively. The values in the brackets reflect the BLEU score difference between the task-specific and ULM Transformer models.

The Transformer deployed in this experiment only obtained a lower BLEU score on the validation corpus, relative to the ULM Transformer in Experiment 1. Table F.1 and Table F.2 each present an example of a translation generated by the task-specific Transformer of a sentence from the test and global test corpora, respectively. In both tables, we will also get to see how the predicted translations compare to the translations generated by the ULM Transformer.

6.3.2 English-to-isiNdebele

A Transformer was trained on a corpus whose target sentences were tokenized using a task-specific tokenizer for the translation of English into isiNdebele. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 11.85(-0.12) on the validation corpus, and scores of 8.43(-2.83) and 9.68(+0.09) on the test and global test corpora, respectively. The values in the brackets reflect the BLEU score difference between the task-specific and ULM Transformer models.

The Transformer deployed in this experiment obtained lower BLEU scores on the validation and test corpora, relative to the ULM Transformer in Experiment 1. Table F.3 and Table F.4 each present an example of a translation generated by the task-specific Transformer of a sentence from the test and global test corpora, respectively. In both tables, we will also get to see how the predicted translations compare to the translations generated by the ULM Transformer.

6.3.3 English-to-Xitsonga

A Transformer was trained on a corpus whose target sentences were tokenized using a task-specific tokenizer for the translation of English into Xitsonga. Once training was complete, the parameter update in the Transformer that was deployed for further evaluation obtained a BLEU score of 33.83(+0.01) on the validation corpus, and scores of 35.03(+0.22) and 28.83(+0.91) on the test and global test corpora, respectively. The values in the brackets reflect the BLEU score difference between the task-specific and ULM Transformer models.

The Transformer deployed in this experiment obtained higher BLEU scores on the validation, test and global test corpora, relative to the ULM Transformer in Experiment 1. Table F.5 and Table F.6 each present an example of a translation generated by the task-specific Transformer of a sentence from the test and global test corpora, respectively. In both tables, we will also get to see how the predicted translations compare to the translations generated by the ULM Transformer.

6.3.4 Discussion

In Table 6.9, we present the BLEU scores obtained by the Transformers in this experiment. The scores were computed on each language pair’s test and global test corpora. For each BLEU score, a 95% confidence interval is provided. As part of this experiment, we compare the impact that the task-specific tokenizer has on the translation performance of the Transformer, relative to ULM. Table 6.9 also presents the p -values computed comparing the task-specific Transformer to the ULM Transformer.

Task-Specific		
English-to-Northern Sesotho	Test	17.92 (17.93 $\pm$ 0.71) (p = 0.2438)
	Global Test	20.01 (20.01 $\pm$ 1.23) (p = 0.1968)
English-to-isiNdebele	Test	8.43 (8.44 $\pm$ 0.49) (p = 0.0010)*
	Global Test	9.68 (9.69 $\pm$ 1.17) (p = 0.3207)
English-to-Xitsonga	Test	35.03 (35.03 $\pm$ 1.21) (p = 0.1558)
	Global Test	28.83 (28.83 $\pm$ 1.63) (p = 0.0140)*

Table 6.9: The BLEU scores of the Transformers deployed for each language pair in the baseline experiment. Each score includes its 95% confidence interval. The last column contains the p -value for the paired significance tests that were conducted. The asterisk indicates that based on the p -value, the two models were found to behave differently, and the **bold font** indicates that this is best score and it is statistically significant from the baseline model in the first experiment.

When we compare the task-specific Transformers to the ULM Transformers, we see that the task-specific Transformers obtained higher BLEU scores on the test and global test corpora, except for the Transformer trained for the translation of English into isiNdebele. That model obtained a lower BLEU score on the English-to-isiNdebele test corpus, compared to the ULM Transformer. The small p -value computed for the paired significance test comparing the task-specific Transformer to the ULM Transformer indicates that there is sufficient evidence to reject the null hypothesis, and conclude that there is a statistically significant difference in how the two models generate translations on the English-to-isiNdebele test corpus. From this conclusion it is reasonable to

assume that the modification made to ULM via the task-specific tokenizer did not benefit the Transformer on the test corpus, with respect to the BLEU score.

While the task-specific Transformer obtained higher BLEU scores on the English-to-isiNdebele global test corpus, and all the other language pairs' test and global test corpora; the large p -values computed for the paired significance tests comparing the task-specific Transformers to the ULM Transformers indicate that we do not have sufficient evidence to reject the null hypotheses. Meaning that it is within reason to assume that both approaches to tokenization impact the Transformer similarly, except for the Transformer trained for the translation of English into Xitsonga. The small p -value computed for its paired significance test indicates that there is sufficient evidence to reject the null hypothesis, and conclude that there is a difference in how the two models generate translations on the English-to-Xitsonga global test corpus. From this conclusion it is reasonable to assume that relative to ULM, the task-specific tokenizer was beneficial to the Transformer for the translation of English-to-Xitsonga sentence pairs in the global test corpus.

By and large the task-specific tokenizer did not significantly improve the translation performance of the Transformer, relative to ULM. Recall that the tokenizer is pre-trained to compute the probabilities for tokens from the ULM. When we jointly train the tokenizer with a Transformer, the goal is to fine-tune how the tokenizer assigns probabilities to the tokens in the vocabulary. The joint training process is computationally expensive. Due to time and resource constraints, the models in this experiment were only allowed to run for 30 epochs. While the task-specific tokenizer did not negatively impact the translation performance of the Transformer, with respect to the BLEU score; the minor improvements suggest that the 30 epochs for joint training is not enough, or that the corpora used were too small train the large joint model, especially in the case of the English-to-Northern Sesotho and -isiNdebele corpora.

6.4 Conclusion

In this chapter we presented the results, and discussed the impact of the three experiments. The models in the second and third experiment were compared to the baseline models in the first experiment. The outcome of the experiments illustrate that while standard subword tokenization algorithms generally outperform their corresponding subword regularization methods, with respect to the BLEU score; subword regularization displays superior generalisation capabilities in the face of noisy training examples. Compared to standard subword tokenization algorithms, the performance of the task-specific tokenizer is generally comparable; however, with enough training examples the task-specific tokenizer demonstrates the capacity to improve the predictive accuracy of the model. In the following chapter, we will conclude this thesis with an outline of the key findings based on the discussions in this chapter, the limitations of the experiments, and recommendations for future work aligned with the research in this thesis.

Chapter 7

Conclusion

7.1 Introduction

Recall that the objective of this thesis is to investigate whether the NMT models deployed for the translation of English into low-resource South African languages can benefit from carefully tuning of the vocabulary representation, architecture and/or the training strategies of these models. More specifically, this thesis investigates the impact of adaptations made to standard subword tokenization algorithms on the translation performance of these models. Due to constraints with respect to time and computational resources, this thesis only considers translations from English into a subset of South African languages for its investigation, which includes isiNdebele, Northern Sesotho and Xitsonga.

7.2 Conclusions

Based on the findings and discussions in Chapter 6, it can be concluded that relative to South African languages with a mostly disjunctive writing system, NMT models perform poorly translating into South African languages with a conjunctive writing system, with respect to the BLEU score. South African languages with a conjunctive writing system are typically characterised by longer words and shorter sentences (Prinsloo and de Schryver, 2002; Spaull et al., 2020), making the modelling of such languages a difficult task, especially in low-resource data settings; as well as when evaluating these languages using a metric that only performs a count of the number of matching n -grams between a prediction and reference translation, ignoring any other meaning that the NMT model is able to capture within the words of these languages. With regards to the comparative analysis, measuring the impact of standard subword tokenization algorithms on the translation performance of text, from English into low-resource South African languages, with adaptations made these algorithms, it can be concluded that the standard subword tokenization algorithms, in the baseline experiments, generally outperform their corresponding subword regularization method, with respect to the BLEU score; however, subword regularization displays more superior generalisation capabilities in the face of noisy training examples. Recall that in Subsection 3.2.1, we describe the issues observed in the parallel corpora sourced from the Autshumato Project that negatively affect the quality of the corpora. The noisy examples refers to segments that display these issues. The data preprocessing steps, as described in Subsection 3.2.2, that were taken to clean, simplify and filter out these noisy segments in the training corpora were based on heuristic rules that were derived from analysing the corpora¹. Without a linguist and/or professional translator to inspect each and every segment pair, we cannot guarantee that the corpora are completely free from noise after preprocessing; only that the noise has been reduced. In the case of the task-specific tokenizer, while its impact on translation performance is comparable to its corresponding standard subword tokenization algorithm; with enough examples as evidenced by the English-to-Xitsonga language pair, it can improve the predictive accuracy of the model. Lastly,

¹For example, in Subsection 3.2.2 we computed source-target ratios of segments pairs in a corpus, using the length of each source segment and dividing it by the length of its corresponding target segment. Segment pairs whose ratios were found to be an outlier were then filtered out. The motivation behind this step was to try and filter out segment pairs whose source segments were aligned to incomplete, over-translations, or completely irrelevant translations on the target side.

while the experiments did not always produce the expected outcomes, they revealed promising opportunities for further work in line with the aim of this thesis: low-resource data settings are not entirely a constraint to NMT; instead, they present an opportunity to develop more domain-specific solutions that are optimised for time and resource efficiency.

7.3 Limitations

Some of the limitations in this thesis include:

1. **Computational resources and time constraints:** This limitation meant that no model was trained until convergence; therefore, when experiments did not return an expected outcome based on experiments conducted in earlier publications, this is a potential factor. This limitation also meant that no hyperparameter tuning was explored to determine a more optimal implementation of the modified approaches to standard subword tokenization. In the discussion of the results, some assumptions made about certain outcomes are attributed to languages' linguistic properties. However, due to the computational resources and time constraints, we could not evaluate whether these outcomes hold for other closely related languages, for a more effective comparison.
2. **Evaluation metric:** While the BLEU score provides a sophisticated count of n -gram matches (Papineni et al., 2002); it does not account for similar translations. The models tend to predict very rudimentary and direct translations relative to the reference translations provided which are more fluent and nuanced². While these direct translations are not ideal, they are also not always incorrect; the BLEU score does not account for this. In the case of South African languages, there is a lack of standardisation with regards to certain naming conventions, with some linguists opting to adapt existing English terms, while others opt to developing words through linguistic principles. For example, the word "charter" is translated to "kilomtlolomalungelo", "emtolwenethu", "kileNcwadi yesiVumelwano", and "kiletjhatha" by the four translators in the English-to-isiNdebele global test corpus. This lack of standardisation affects the quality of the evaluations using the BLEU score. Secondly, the typological and orthographic complexity of South African languages are disadvantaged when evaluated against this score, because the score does not take into consideration the number of components in a sentence that are sometimes captured in a single word.
3. **Absence of professional translators and linguists:** This thesis relied on data-driven approaches to clean the corpora used to develop the models, and evaluate the predictions generated. The lack of collaboration with professional translators and linguists meant that the models' translation performance is distorted by the noisy training examples, and the evaluation of predictions is limited to an automatic, language-independent metric.

7.4 Future Work

The opportunities for future work in line with this research include:

1. The architecture of the NMT model used in this thesis is based on the work of the medium-depth Transformer (van Biljon et al., 2020). The findings in van Biljon et al. (2020), Rajab (2022) and this thesis highlight the potential to use smaller models in low-resource data settings to generate domain-specific translations. However, like any other machine learning model, the effectiveness of these smaller models will be dependent on quality data. Therefore, there is an opportunity to prioritise collaborating with professional translators to craft smaller, but better quality corpora; as opposed to sourcing very large, and noisy corpora. Such efforts minimise the distortions caused by noisy training corpora, and make learning representations easier for these smaller models.
2. Similarly to Rajab (2022), there is an opportunity investigate more optimal implementations of the subword regularization algorithms, and the task-specific tokenizer for South African languages. This would involve looking into appropriate hyperparameter configurations for these methods. For the task-specific tokenizer, this thesis only pre-trains the target language tokenizer prior to training the joint model; however, similar to an experiment conducted in Hiraoka et al. (2021), it might be worthwhile to explore how to pre-train the NMT model

²See Chapter 6, we reflect on some of the example translations in the first experiment.

prior to the training of the joint model³. By doing so, we can continue to keep the number of training steps required for the joint model to a minimum; as opposed to training the joint model for significantly longer. Lastly, while searching for more optimal implementations of these methods, it would also be worthwhile to explore whether the findings hold for more closely related languages, such as all Nguni languages, Sotho-Tswana languages, etc.

3. The value of automatic evaluation metrics is that it enables developers of MT models to evaluate large amounts of predicted translations, at a much quicker and affordable rate (Papineni et al., 2002). Many of these metrics aim to be language-independent, limiting their effectiveness beyond the languages that were used to develop these metrics. Taking into consideration the linguistic properties of South African languages highlighted in Chapter 3, and the some of the limitations of the BLEU score observed for the evaluation of isiNdebele translations in Chapter 6, there is an opportunity to work with linguists and translators to develop more effective automatic evaluations for South African languages, more especially Nguni languages that we suspect also suffer from the constraints that were observed for isiNdebele.

³Hiraoka et al. (2021) train an NMT model, then apply the optimised model’s weights to the NMT model prior to training the joint model.

Appendix A

A Case for Subword Tokenization

We conducted an exploratory data analysis on the English-to-isiNdebele training parallel corpus to show empirically the linguistic differences that exist between English and isiNdebele in order to make a case for why representing segments in these languages as sequences of subword tokens is a sensible choice for NMT.

The number of segment pairs in the English-to-isiNdebele training corpus after preprocessing is 89 783. Figure A.1 presents the distributions of lengths of segments on the English and isiNdebele side of the training corpus. From the distributions, we observe that on average the length of segments in isiNdebele are shorter than the segments in English. In fact, based on the summary statistics of the data, 75% of segments in isiNdebele contain 18 or less whitespace characters; whereas only 50% of segments in English contain 17 or less whitespace characters.

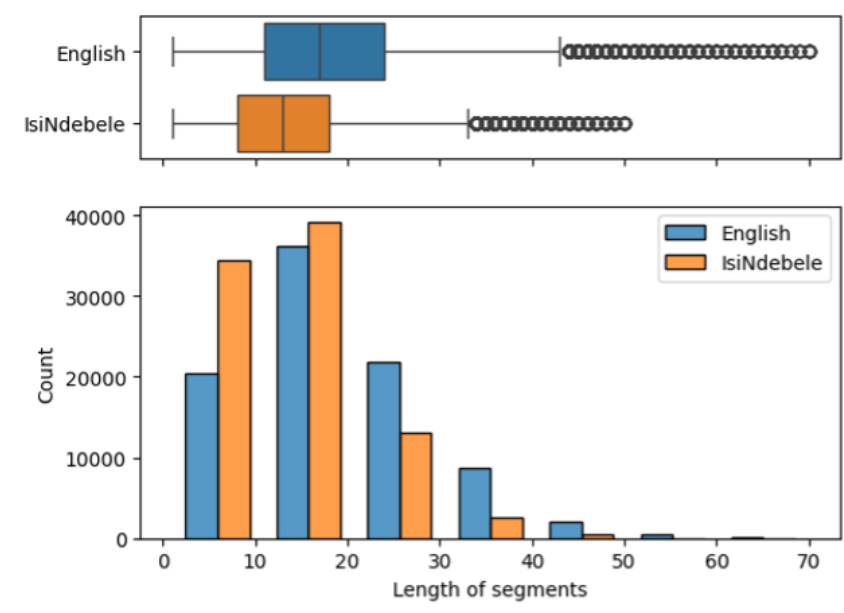


Figure A.1: Distributions of lengths of English and isiNdebele segments.

Figure A.2 presents the distribution of English-to-isiNdebele segment pair ratios that are computed by taking the length of an English segment and dividing it by the length of its corresponding isiNdebele segment. Similarly to the lengths computed in Figure A.1, the lengths for the ratio are computed with respect to the number of whitespace tokens in the segment. From the distribution in Figure A.2, we observe that the majority segment pair ratios are greater than 1. In fact, more than 75% of segment pair ratios are greater than 1.667 based the summary statistics of the data. This means that in general, English segments contain more whitespace characters than their corresponding segments in isiNdebele.

The unique whitespace characters in the training parallel corpus include words, numbers and punc-

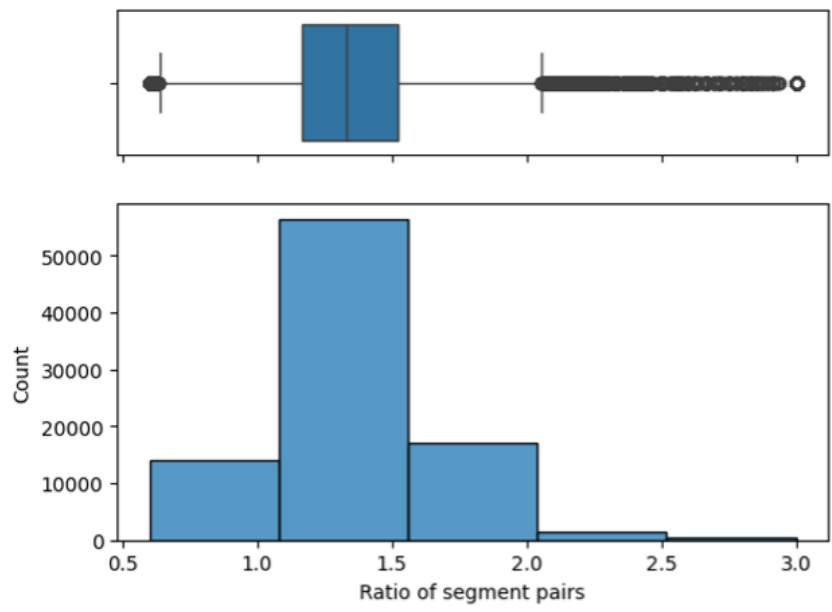


Figure A.2: Distribution of English-to-isiNdebele segment pair ratios.

tuation marks. On the English side of the corpus, there are 37 709 unique whitespace characters; whereas the isiNdebele side contains 164 887 unique whitespace characters. Figure A.3 presents the distributions of lengths of whitespace characters on the English and isiNdebele sides of the training corpus. Included in the figure is the distribution of lengths of subword tokens that were generated from the training corpus using the BPE tokenization algorithm to represent segments in both English and isiNdebele. From Figure A.3, we observe that not only are there more unique whitespace characters in isiNdebele than there are in English, but the length of these isiNdebele characters are also generally longer.

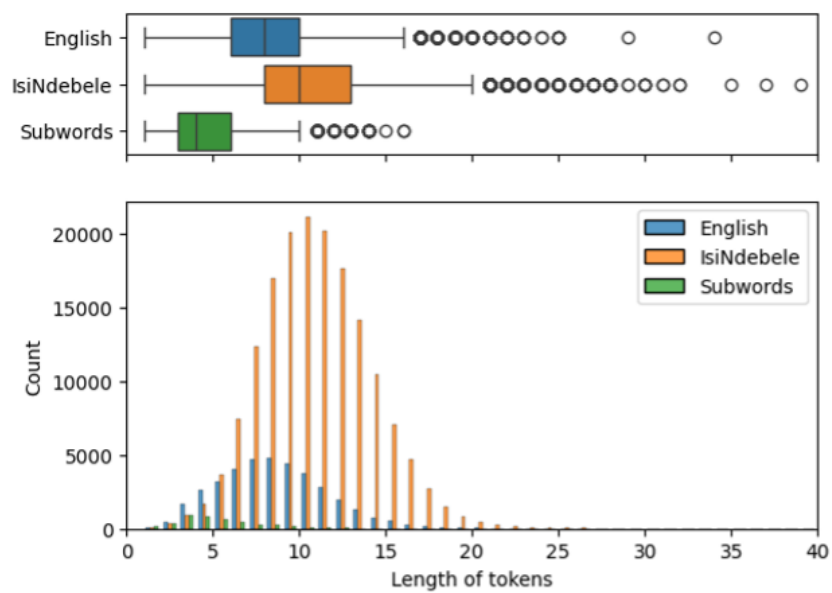


Figure A.3: Distributions of lengths of whitespace characters on the English and isiNdebele sides of the training corpus, along with the length of subword tokens from a BPE-generated vocabulary.

Subwords tokens are segments of words, meaning that they can be as short as a single-character, as long as a whole word or somewhere in-between. Looking at Figure A.3, it then follows that in general the length of subword tokens would be shorter than the whitespace tokens on the English and isiNdebele side of the training corpus. However, what is more interesting is how this figure is able to put into perspective just how small the BPE-generated vocabulary of subword tokens is relative to the number of whitespace tokens there are in English and isiNdebele. Based

on the summary statistics for the subword token data, the vocabulary contains 4 164 subword tokens, with the 50% of tokens measuring at a length of 4 or less. What we will observe in Chapter 6 is that these memory-efficient sized vocabularies generated via subword tokenization contain subword tokens that are able to efficiently encode segments on both sides of the training corpus in a way that enables NMT models to learn meaning, perform translation and generate unseen words. Figure A.4 presents a word cloud of the most frequently observed subword tokens in the training corpus from the BPE-generated vocabulary.



Figure A.4: Word cloud of subword tokens from a BPE-generated vocabulary.

Figures A.1, A.2 and A.3 all support the claims for why we should use subword tokens to represent English-to-isiNdebele segment pairs for NMT: the mapping of words between English and isiNdebele is simply not one-to-one. Not only is isiNdebele’s word formation process more complex relative to English due to isiNdebele being an agglutinating language, it also uses a conjunctive writing system. IsiNdebele’s conjunctive writing system presents itself in shorter segments and longer words in segments compared to English, with the English language’s more isolating word formation processes and intermediate writing system, as seen in all three figures. Since isiNdebele is an agglutinating this means that many of its words can be broken down into prefixes, suffixes, infixes and root words that can be mapped to morphemes, words, phrases or even whole sentences in English. The goal of subword tokenization is to extract appropriate segments of words in English and isiNdebele, with the end goal of enabling the NMT model to learn the mappings between the two languages for translation.

Similar figures to the ones presented in this appendix have been generated for the English-to-Northern Sesotho and English-to-Xitsonga training parallel corpora, and can be located in on the GitHub repository linked to this research: [tyobeka/Research](https://github.com/tyobeka/Research).

Appendix B

Expectation-Maximisation

Under the classical EM framework (Haugh, 2015), we need to define an observed variable, a latent variable and an unknown parameter vector that we wish to optimise. Let X be the observed variable, such that it represents a sentence in the training corpus; $\mathbf{X}$ be the latent variable, such that it represents a subword sequence representation of sentence X ; and $\boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{V}|}$ the parameter vector, such that each index in the vector corresponds to the probability linked to observing each token from the given vocabulary, $\mathcal{V}$.

Once we have defined all the variables, we can formulate a log-likelihood over the entire training corpus. In this case, the log-likelihood function is given by $\ell(\boldsymbol{\theta}; X, \mathbf{X})$. The goal of the EM algorithm is to find the optimal parameter vector, $\boldsymbol{\theta}$, that maximises the expectation of the log-likelihood function over all subword sequence representations of sentences in the training corpus:

1. Expectation (E-Step): Formulate the expectation of the log-likelihood function, $\ell(\boldsymbol{\theta}; X, \mathbf{X})$, over the subword sequence representations of sentences, given the sentences in the training corpus, X , and current parameter estimate, $\boldsymbol{\theta}^{(t)}$:

$$\begin{aligned} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) &= \mathbb{E} \left[\ell(\boldsymbol{\theta}; X, \mathbf{X}) | X, \boldsymbol{\theta}^{(t)} \right] \\ &= \mathbb{E} \left[\log P(X, \mathbf{X} | \boldsymbol{\theta}) | X, \boldsymbol{\theta}^{(t)} \right] \\ &= \sum_{n=1}^{2|\mathcal{D}|} \log \left(\sum_{i=1}^{|\mathcal{S}(X^{(n)})|} P(\mathbf{X}_i^{(n)}) \right), \end{aligned} \tag{B.1}$$

where, $\mathcal{D}$ represents the training corpus, with $|\mathcal{D}|$ sentence pairs; each $X^{(n)}$ represents a sentence in the training corpus; and $\mathcal{S}(X^{(n)}) = \{\mathbf{X}_1^{(n)}, \mathbf{X}_2^{(n)}, \dots\}$ is the set of all valid subword sequence representations of sentence $X^{(n)}$ that have been generated using the tokens in the vocabulary, $\mathcal{V}$.

2. Maximization (M-Step): Update the parameter vector by searching for the vector $\boldsymbol{\theta}^{t+1}$ that optimises that expectation formulated in Equation B.1:

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \tag{B.2}$$

The E-Step and M-Step are repeated until the stopping criterion is met. It is not clear what stopping criterion Kudo (2018) applies. However, in general the two steps are repeated until (i.) the parameter vector $\boldsymbol{\theta}$ converges, that is that the difference between $\boldsymbol{\theta}^{(t)}$ and $\boldsymbol{\theta}^{(t+1)}$ does not exceed a pre-defined threshold, or (ii.) a maximum number of iterations have been carried out.

Note: In the description provided by HuggingFace on how to implement ULM tokenization algorithm, the parameter vector assumed to optimise the expectation in Equation B.1 is computed such that each index in the vector corresponds the frequency of each token in the training corpus relative to the frequency of all the tokens in the vocabulary.

Appendix C

Viterbi Algorithm

The Viterbi algorithm (Viterbi, 1967) assumes that the process of tokenizing a sentence can be modelled using the Hidden Markov Model (HMM). Under the HMM framework (Jurafsky and Martin, 2024), we need to define a sequence of observations, and a set of hidden states. Let sentence X , encoded as $\{c_1, \dots, c_T\}$ where each c_i is a character in the sentence, be the sequence of observations; and the tokens in the vocabulary, $x_j \in \mathcal{V}$, be the hidden states.

The goal of the Viterbi algorithm is to find $\mathbf{X}^* = \{x_1, \dots, x_N\}$ that satisfies Equation 4.2. For each token in the vocabulary, $x_j \in \mathcal{V}$, the algorithm recursively searches for the most probable sequence of subword tokens $\{x_1, \dots, x_{n-1}\}$ that lead to subword token $x_n = x_j$, to represent the segment of the sentence, $\{c_1, \dots, c_t\}$:

1. Initialisation: At time step $t = 1$, we let the segment of the sentence equal to $\{c_1\}$, and search for the most likely token in the vocabulary to represent the segment, $x_1 = x_j$ such that $x_j \in \mathcal{V}$.
2. Recursion: At time step $t = 2, \dots, T$, we consider a segment with the next character in the sentence appended to it, that is $\{c_1, \dots, c_t\}$. For each token in the vocabulary, $x_j \in \mathcal{V}$, the algorithm searches for the most probable sequence of subword tokens $\{x_1^j, \dots, x_{n-1}^j\}$ that lead to subword token $x_n^j = x_j$, that characterise as a valid subword sequence representation the segment, $\{c_1, \dots, c_t\}$.
3. Termination: Once the recursion is complete, we will have computed the most probable path, $\{x_1^j, \dots, x_{N-1}^j\}$, to $x_N^j = x_j$, for each $x_j \in \mathcal{V}$, that characterises as a valid subword sequence representation the sentence, $\{c_1, \dots, c_T\}$. The algorithm outputs the $\{x_1^j, \dots, x_N^j\}$ that satisfies Equation 4.2.

Appendix D

Forward-Filtering and Backward-Sampling Algorithm

The FFBS algorithm (Scott, 2002) allows us to approximate the sampling from a multinomial distribution, when $l \rightarrow \infty$. Similarly to the Viterbi algorithm, FFBS is applied to tasks whose processes are governed by a HMM. Let sentence X , encoded as $\{c_1, \dots, c_t\}$ where each c_i is a character in the sentence, be a sequence of observations; and the tokens in the vocabulary, $x_j \in \mathcal{V}$, be the hidden states.

The goal of the FFBS is to use DP to efficiently sample a sequence of tokens from the vocabulary to represent the sentence (Kudo, 2018). As the name suggests, the algorithm comprises two major steps, namely a forward-filtering step and a backward-sampling step:

1. Forward-Filtering: For each token in the vocabulary, $x_j \in \mathcal{V}$, this step recursively computes a forward-filtering distribution that accounts for the probabilities of the subword sequence leading to x_j , that characterise valid subword sequence representations of the segment of a sentence, $\{c_1, \dots, c_t\}$:
 - 1.1. Initialisation: At time step $t = 1$, we let the segment equal to $\{c_1\}$, and search for the most likely token in the vocabulary to represent the segment.
 - 1.2. Recursion: At time step $t = 2, \dots, T$, we consider a segment with the next character in the sentence appended to it, that is $\{c_1, \dots, c_t\}$. For each token in the vocabulary, $x_j \in \mathcal{V}$, this step computes a forward-filtering distribution that accounts for the probabilities of all the subword sequence leading to x_j , that characterise valid subword sequence representations of the segment of a sentence, $\{c_1, \dots, c_t\}$.
2. Backward-Sampling: This step uses the forward-filtering distributions to sample a sequence of tokens to represent the sentence, X :
 - 2.1. Initialisation: Starting at the end of the sentence, this step samples a subword token in the final position of a valid subword sequence representation of the sentence, x_N .
 - 2.2. Recursion: For $n = N - 1, N - 2, \dots, 1$, it then samples the preceding token, x_n , to the subword token x_{n+1} from the forward-filtering distribution of all valid paths leading to observing token x_{n+1} . This recursive step is repeated to re-construct a valid subword sequence representation of the sentence.

Appendix E

Forward-DP and Backward-A* Algorithm

The Forward-DP Backward-A* algorithm [Nagata \(1994\)](#) allows us to search for the m -best sequence of subword tokens to represent a sentence. Similarly to the Viterbi algorithm, the Forward-DP Backward-A* algorithm is applied to tasks whose processes are governed by a HMM. Let sentence X , encoded as $\{c_1, \dots, c_t\}$ where each c_i is a character in the sentence, be a sequence of observations; and the tokens in the vocabulary, $x_j \in \mathcal{V}$, be the hidden states.

The goal of Forward-DP Backward-A* algorithm is to use DP to perform an efficient search to determine the m -best sequence representations of a sentence, $\mathbf{X}_1^*, \dots, \mathbf{X}_m^*$ that satisfy Equation 4.2. As the name suggests, the algorithm comprises two major steps: a forward-DP, and a backward-A* search.

1. Forward-DP search: Starting with a single character, and incrementally appending the next character in the sentence to it, for each time step $t = 1, \dots, T$, the forward search will keep track of sequences of tokens leading up each token in the vocabulary, $x_j \in \mathcal{V}$, that are valid representations of each segment of sentence $\{c_1, \dots, c_t\}$. These sequences are referred to as partial paths in [Nagata \(1994\)](#). Along with each partial path, the forward search also keeps track of the paths corresponding probabilities.
2. Backward-A* search: Starting with tokens that are valid representations of the end of the sentence and working backwards, the backward search recursively tries to extend the backward partial paths. A backward partial path refers to tokens representing the tail-end of a sentence. The backward partial paths are extended one token at a time based on the record of partial paths recorded in the forward search. With each recursive step, probabilities are computed of the sequences by summing the probability of the backward partial path with the best probability of the partial path leading up to that point; and the sequences are then ranked based on the top- m potential sequence representations.

Appendix F

Model Predictions

The predictions presented in this appendix have all been post-processed. All BLEU scores and paired statistical significance tests were computed using the post-processed predictions. The code to post-process a prediction is presented in Appendix G.

Type	Sentence
Source	Registered testing stations may only examine and test motor vehicles in accordance with their grading.
Reference	Diteišene tše di ngwadišitšwego di ka dira diteko tša dikoloi tša bogolo bjoo bo dumeletšwego ka gona.
BPE	Dipeistara tše di ngwadišitšwego di ka dirwa fela gomme tša rekišwa dikoloi go ya ka dikoloi go ya ka difatanaga.
ULM	Diteše tša diteko tše di ngwadišitšwego di ka hlahlobja gomme diteko tša diteko tša dikoloi go ya ka mabaka a bona
BPE-Dropout	Diteko tše di ngwadišitšwego di ka dirwa fela gomme di ka dira dikoloi go ya go ya ka dikoloi tša bona.
ULM with Subword Regularization	Diteko tša go ngwadiša diteko di ka hlahlobolwa ke dikoloi tše di ka hlahlolwago go ya ka dikoloi tša bona.
Task-Specific	Diteišene tše di ngwadišitšwego di ka hlahlobja fela gomme diteko tša dikoloi tša go ya ka mabaka a bona

Table F.1: An English-to-Northern Sesotho sentence pair from the test corpus alongside the predictions of the source sentence generated by each of the Transformers.

Type	Sentence
Source	We want you, our client, to judge us according to the standard of service we set out in this charter.
References	1. Re nyaka gore wena, modiriši wa rena, go re ahlola ka maemo a tirelo ao re a filego ka mo gare ga tšhata ye. 2. Re nyaka gore wena, moreki wa rena, o re ahlole go ya ka boemo bja tirelo bjoo re bo beago ka go lengwalotumelelo le. 3. Re rata gore wena moreki, o ahlole mošomo wa rena go ya ka maemo a tirelo ao a lego ka gare ga tšhata ye. 4. Re nyaka wena, khastama ya rena, go re ahlola go ya le ka maemo a tirelo yeo re e beyago ka go lengwalotumelelo le.
BPE	Re nyaka, klaemete ya rena, go ya ka maemo a rena go ya ka maemo a tirelo ye re e beilwego ka mo nakong ye.
ULM	Re nyaka gore o itlama badiredi ba rena bao ba nago le maswanedi a go ya ka maemo a tirelo ye re bego re lebane le yona
BPE-Dropout	Re nyaka, boso bja rena, go ya ka maemo a tirelo ye re beilwego ka mo mo .
ULM with Subword Regularization	Re nyaka, baahlodi ba rena, go ya go ya ka maemo a go ya ka maemo a go ya ka maemong a go tirelo ye.
Task-Specific	Re nyaka gore o ikopanya le moahlodi wa rena go ya ka maemo a godimo a go ya ka maemo a tirelo ye re e beilwego ka gare ga websaete ye

Table F.2: An English sentence from the global test corpus alongside the translations generated of the sentence in Northern Sesotho by four professional translators, and the predictions of the sentence generated by each of the Transformers.

Type	Sentence
Source	It is important to make sure that each member of the project team has a clear understanding of what they must do and how it relates to the tasks of others .
Reference	Kuqakathekile ukuqinisekisa bona ilunga elinye neline lesiqhema sephrojekthi lizwisisa kuhle okufanele kwenziwe nokobana kunobudlelwana obunjani mayelana nemisebenzi yabanye .
BPE	Kuqakathekile ukwenza isiqiniseko sokobana elinye neline ilunga lesiqhema sephrojekthi linokuzwisisa okucacileko kwalokho abakufanele benze begodu nokobana lihlobana bunjani emisebenzini yabanye.
ULM	Kuqakathekile ukwenza isiqiniseko sokobana elinye neline ilunga lesiqhema sephrojekthi lizwisise kuhle lokho okufanele bakwenze begodu nokobana lihlobana njani nemisebenzi yabanye.
BPE-Dropout	Kuqakathekile ukwenza isiqiniseko sokobana elinye neline ilunga lesiqhema sephrojekthi linokuzwisisa okucacileko kwalokho okufanele benze begodu bunjani imisebenzi yabanye.
ULM with Subword Regularization	Kuqakathekile ukwenza isiqiniseko sokobana ilunga elinye neline lesiqhema sephrojekthi linokuzwisisa okucacileko okufanele lenze begodu lihlobana njani nemisebenzi yabanye.
Task-Specific	Kuqakathekile ukwenza isiqiniseko sokobana elinye neline ilunga lesiqhema sephrojekthi linokuzwisisa okucacileko lokho abakwenzako begodu nokobana lihlobana njani nemisebenzi yabanye.

Table F.3: An English-to-isiNdebele sentence pair from the test corpus alongside the predictions of the source sentence generated by each of the Transformers.

Type	Sentence
Source	We want you, our client, to judge us according to the standard of service we set out in this charter.
References	1. Sifuna wena, njengekhastama lethu, bona usahlulele ngezinga lomsebenzi esiwuveza kilomtloomalungelo. 2. Sifuna wena, njengekhastama lethu, usahlulele ngezinga lomsebenzi esiwethulako emtlolewenethu. 3. Sifuna wena, mthengi wethu, ukobana usahlulele ngokwezinga lomsebenzi esitole ngawo kileNcwadi yesiVumelwano. 4. Sifuna wena, ikhastama lethu, usijaje ngezinga lomsebenzi wethu esilibeke kiletjhatha.
BPE	Sifuna ukuveza, itlayenti lethu, ukuhlulela ngokuya kwezinga lemisebenzi esiyibekwe kiletjhadi.
ULM	Sifuna bona unikele amatlayenti wethu anikela ijaji ukuya ngokwezinga lomsebenzi esiwubeka kiletjhadi
BPE-Dropout	Sifuna, itlayenti yethu, ukuhlulela ngokuya kwezinga lemisebenzi esiyibekwe kiletjhadi.
ULM with Subword Regularization	Sifuna, utlayenti yethu, ukujajisa ngokuya kwezinga lemisebenzi esibekwe etjhadini le.
Task-Specific	Sifuna ukuthokoza amatlayenti wethu ukobana sihlule ukuya ngokwezinga lomsebenzi esizibeke kiletjhadi.

Table F.4: An English sentence from the global test corpus alongside the translations generated of the sentence in isiNdebele by four professional translators, and the predictions of the sentence generated by each of the Transformers.

Type	Sentence
Source	This will play a central role in development cooperation .
Reference	Leswi swi ta va ntirho wa nkoka eka ntirhisano wa swa nhluvukiso .
BPE	Leswi swi ta tlanga xiave xa nkoka eka ntirhisano wa nhluvukiso.
ULM	Leswi swi ta tlanga xiave xa nkoka eka ntirhisano wa nhluvukiso.
BPE-Dropout	Leswi swi ta tlanga xiave xa nkoka eka ntirhisano wa nhluvukiso.
ULM with Subword Regularization	Leswi swi ta va na xiave xa nkoka eka ntirhisano wa nhluvukiso.
Task-Specific	Leswi swi ta tlanga xiave xa nkoka eka ntirhisano wa nhluvukiso.

Table F.5: An English-to-Xitsonga sentence pair from the test corpus alongside the predictions of the source sentence generated by each of the Transformers.

Type	Sentence
Source	We want you, our client, to judge us according to the standard of service we set out in this charter.
References	1. <i>Hi lava wena, khasimende ya hina, ku hi ahlula hi xiyimo xa vukorhokeri lexi hi xi vekeke eka xiyimo xa vukorhokeri lexi kumekaka eka tsalwa leri ra timfanelo.</i> 2. <i>Hi tsakela wena khasimende ya hina ku nyika mavonelo hi xiyimo xa vukorhokeri lebyi hi byi kombiseke eka tsalwa leri.</i> 3. <i>Hi lava leswaku n'wina, tikhasimende ta hina, mi hi avanyisa hi xiyimo xa vukorhokeri bya hina lebyi hi byi simekeke eka Ntwanano lowu.</i> 4. <i>Hi lava wena, khasimende ya hina, u hi pima hi mipimo ya vukorhokeri leyi hi yi vekaka eka chata leyi.</i>
BPE	Hi lava ku khumba, tlilayente ya hina, ku hi ahlula ku ya hi mpimo wa vukorhokeri lebyi hi vekeke eka filipichati leyi.
ULM	Hi lava leswaku u tipfumelela ku ahlula muavanyisi wa hina ku ya hi mpimo wa vukorhokeri lowu hi wu vekaka eka chati leyi
BPE-Dropout	Hi lava wena, tlilayimete ya hina, ku hi avanyisa hi ku ya hi xiyimo xa vukorhokeri lebyi hi byi vekeke eka filichati leyi.
ULM with Subword Regularization	Hi lava, ku ahlula hi ku landza mpimo wa vukorhokeri lebyi hi byi vekekeke eka chati leyi.
Task-Specific	Hi lava leswaku u tipfumelela hi ku tirhisa mpimo wa vukorhokeri lowu hi vekeriweke eka filichati

Table F.6: An English sentence from the global test corpus alongside the translations generated of the sentence in Xitsonga by four professional translators, and the predictions of the sentence generated by each of the Transformers.

Appendix G

Post-Processing Translations

```
1 before_punc = '!,.,;?)'
2 def remove_whitespace_before(segment, punc):
3     """
4     This function looks for punctuation in a segment and removes the whitespace
5     before the punctuation mark.
6
7     Args:
8         segment (str): segment to be processed.
9         punc (str): string of punctuation marks.
10
11     Returns:
12         The segment with whitespace removed before punctuation marks in punc.
13     """
14     punc_positions = []
15     for i, symbol in enumerate(segment):
16         if symbol in punc:
17             punc_positions.append(i)
18
19     punc_positions.reverse()
20     for pos in punc_positions:
21         if segment[pos-1] == " ":
22             segment = segment[0:pos-1] + segment[pos:]
23
24     return segment
```

Listing G.1: A code snippet of how the translations generated by the Transformer models were post-processed.

Appendix H

Code

The data and code for this research is available in the following Github repository: [tyobeka/Re-search](#)

Bibliography

- Agić, Ž. and Vulić, I. (2019), JW300: A wide-coverage parallel corpus for low-resource languages, *in* A. Korhonen, D. Traum and L. Màrquez, eds, ‘Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics’, Association for Computational Linguistics, Florence, Italy, pp. 3204–3210.
URL: <https://aclanthology.org/P19-1310/>
- Araabi, A. and Monz, C. (2020), Optimizing transformer for low-resource neural machine translation, *in* D. Scott, N. Bel and C. Zong, eds, ‘Proceedings of the 28th International Conference on Computational Linguistics’, International Committee on Computational Linguistics, Barcelona, Spain (Online), pp. 3429–3435.
URL: <https://aclanthology.org/2020.coling-main.304>
- Ba, J. L., Kiros, J. R. and Hinton, G. E. (2016), ‘Layer normalization’.
URL: <https://arxiv.org/abs/1607.06450>
- Bahdanau, D., Cho, K. and Bengio, Y. (2016), ‘Neural machine translation by jointly learning to align and translate’.
URL: <https://arxiv.org/abs/1409.0473>
- Brown, P. F., Cocke, J., Pietra, S. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L. and Roossin, P. S. (1990), ‘A statistical approach to machine translation’, *Comput. Linguistics* **16**, 79–85.
URL: <https://api.semanticscholar.org/CorpusID:14386564>
- Cho, K., van Merriënboer, B., Bahdanau, D. and Bengio, Y. (2014), ‘On the properties of neural machine translation: Encoder-decoder approaches’.
URL: <https://arxiv.org/abs/1409.1259>
- Cho, K., van Merriënboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), Learning phrase representations using rnn encoder–decoder for statistical machine translation, *in* ‘Conference on Empirical Methods in Natural Language Processing’.
URL: <https://api.semanticscholar.org/CorpusID:5590763>
- Chung, J., Cho, K. and Bengio, Y. (2016), A character-level decoder without explicit segmentation for neural machine translation, *in* K. Erk and N. A. Smith, eds, ‘Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)’, Association for Computational Linguistics, Berlin, Germany, pp. 1693–1703.
URL: <https://aclanthology.org/P16-1160/>
- Cieri, C., Maxwell, M., Strassel, S. and Tracey, J. (2016), Selection criteria for low resource language programs, *in* N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk and S. Piperidis, eds, ‘Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)’, European Language Resources Association (ELRA), Portorož, Slovenia, pp. 4543–4549.
URL: <https://aclanthology.org/L16-1720>
- Costa-jussà, M. R., Farrús, M., Mariño, J. B. and Fonollosa, J. A. R. (2010), Automatic and human evaluation study of a rule-based and a statistical Catalan-Spanish machine translation systems, *in* N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner and D. Tapias, eds, ‘Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)’, European Language Resources Association (ELRA), Valletta, Malta.
URL: <https://aclanthology.org/L10-1023/>

- ## BIBLIOGRAPHY

- He, K., Zhang, X., Ren, S. and Sun, J. (2015), ‘Deep residual learning for image recognition’.
URL: <https://arxiv.org/abs/1512.03385>
- He, X., Haffari, G. and Norouzi, M. (2020), ‘Dynamic programming encoding for subword segmentation in neural machine translation’.
URL: <https://arxiv.org/abs/2005.06606>
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. R. (2012), ‘Improving neural networks by preventing co-adaptation of feature detectors’.
URL: <https://arxiv.org/abs/1207.0580>
- Hiraoka, T., Takase, S., Uchiumi, K., Keyaki, A. and Okazaki, N. (2020), Optimizing word segmentation for downstream task, *in* T. Cohn, Y. He and Y. Liu, eds, ‘Findings of the Association for Computational Linguistics: EMNLP 2020’, Association for Computational Linguistics, Online, pp. 1341–1351.
URL: <https://aclanthology.org/2020.findings-emnlp.120/>
- Hiraoka, T., Takase, S., Uchiumi, K., Keyaki, A. and Okazaki, N. (2021), ‘Joint optimization of tokenization and downstream model’.
URL: <https://arxiv.org/abs/2105.12410>
- Hochreiter, S. and Schmidhuber, J. (1997), ‘Long short-term memory’, *Neural Computation* **9**, 1735–1780.
- Hutchins, W. J. (2001), ‘Machine translation over fifty years’, *Histoire épistémologie langage* **23**(1), 7–31.
- Ioffe, S. and Szegedy, C. (2015), ‘Batch normalization: Accelerating deep network training by reducing internal covariate shift’.
URL: <https://arxiv.org/abs/1502.03167>
- Janson, T. (1991-1992), ‘Southern bantu and makua’, *Sprache und Geschichte in Africa* **12/13**, 63–106.
- Jean, S., Cho, K., Memisevic, R. and Bengio, Y. (2015), ‘On using very large target vocabulary for neural machine translation’.
URL: <https://arxiv.org/abs/1412.2007>
- Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M. and Dean, J. (2017), ‘Google’s multilingual neural machine translation system: Enabling zero-shot translation’.
URL: <https://arxiv.org/abs/1611.04558>
- Jurafsky, D. and Martin, J. H. (2024), ‘Speech and language processing: Chapter a: Hidden markov models (draft)’.
URL: <https://web.stanford.edu/~jurafsky/slp3/A.pdf>
- Kalchbrenner, N. and Blunsom, P. (2013), Recurrent continuous translation models, *in* D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu and S. Bethard, eds, ‘Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing’, Association for Computational Linguistics, Seattle, Washington, USA, pp. 1700–1709.
URL: <https://aclanthology.org/D13-1176>
- Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A. and Kavukcuoglu, K. (2017), ‘Neural machine translation in linear time’.
URL: <https://arxiv.org/abs/1610.10099>
- Kessler, B. (1995), Computational dialectology in Irish Gaelic, *in* S. P. Abney and E. W. Hinrichs, eds, ‘Seventh Conference of the European Chapter of the Association for Computational Linguistics’, Association for Computational Linguistics, Dublin, Ireland.
URL: <https://aclanthology.org/E95-1009/>
- Kingma, D. P. and Ba, J. (2017), ‘Adam: A method for stochastic optimization’.
URL: <https://arxiv.org/abs/1412.6980>

- Koehn, P. (2004), Statistical significance tests for machine translation evaluation, *in* D. Lin and D. Wu, eds, ‘Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing’, Association for Computational Linguistics, Barcelona, Spain, pp. 388–395.
URL: <https://aclanthology.org/W04-3250/>
- Koehn, P. and Knowles, R. (2017), Six challenges for neural machine translation, *in* T. Luong, A. Birch, G. Neubig and A. Finch, eds, ‘Proceedings of the First Workshop on Neural Machine Translation’, Association for Computational Linguistics, Vancouver, pp. 28–39.
URL: <https://aclanthology.org/W17-3204>
- Kreutzer, J., Bastings, J. and Riezler, S. (2019), Joey NMT: A minimalist NMT toolkit for novices, *in* ‘Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations’, Association for Computational Linguistics, Hong Kong, China, pp. 109–114.
URL: <https://www.aclweb.org/anthology/D19-3019>
- Krzysztof, W. (2015), ‘Noisy-parallel and comparable corpora filtering methodology for the extraction of bi-lingual equivalent data at sentence level’, *Computer Science* **16**(2), 169.
URL: <http://dx.doi.org/10.7494/csci.2015.16.2.169>
- Kudo, T. (2018), ‘Subword regularization: Improving neural network translation models with multiple subword candidates’.
URL: <https://arxiv.org/abs/1804.10959>
- Kudo, T. and Richardson, J. (2018), ‘Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing’.
URL: <https://arxiv.org/abs/1808.06226>
- Lagarda, A.-L., Alabau, V., Casacuberta, F., Silva, R. and Díaz-de Liaño, E. (2009), Statistical post-editing of a rule-based machine translation system, *in* M. Ostendorf, M. Collins, S. Narayanan, D. W. Oard and L. Vanderwende, eds, ‘Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers’, Association for Computational Linguistics, Boulder, Colorado, pp. 217–220.
URL: <https://aclanthology.org/N09-2055/>
- Levenshtein, V. I. (1965), ‘Binary codes capable of correcting deletions, insertions, and reversals’, *Soviet physics. Doklady* **10**, 707–710.
URL: <https://api.semanticscholar.org/CorpusID:60827152>
- Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M. and Zettlemoyer, L. (2020), ‘Multilingual denoising pre-training for neural machine translation’.
URL: <https://arxiv.org/abs/2001.08210>
- Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O. and Zaremba, W. (2015), ‘Addressing the rare word problem in neural machine translation’.
URL: <https://arxiv.org/abs/1410.8206>
- Luong, T., Pham, H. and Manning, C. D. (2015), Effective approaches to attention-based neural machine translation, *in* L. Màrquez, C. Callison-Burch and J. Su, eds, ‘Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing’, Association for Computational Linguistics, Lisbon, Portugal, pp. 1412–1421.
URL: <https://aclanthology.org/D15-1166/>
- Magueresse, A., Carles, V. and Heetderks, E. (2020), ‘Low-resource languages: A review of past work and future challenges’.
URL: <https://arxiv.org/abs/2006.07264>
- Maho, J. F. (2009), Nogl online the online version of the new updated guthrie list, a referential classification of the bantu languages.
URL: <https://api.semanticscholar.org/CorpusID:63998402>
- Mao, L. (2019), ‘Byte Pair Encoding’.
URL: <https://leimao.github.io/blog/Byte-Pair-Encoding>

- Martinus, L. and Abbott, J. Z. (2019a), ‘Benchmarking neural machine translation for southern african languages’.
URL: <https://arxiv.org/abs/1906.10511>
- Martinus, L. and Abbott, J. Z. (2019b), ‘A focus on neural machine translation for african languages’.
URL: <https://arxiv.org/abs/1906.05685>
- Martinus, L., Webster, J., Moonsamy, J., Jnr, M. S., Moosa, R. and Fairon, R. (2020), ‘Neural machine translation for south africa’s official languages’.
URL: <https://arxiv.org/abs/2005.06609>
- McKellar, C. A. and Puttkammer, M. J. (2020), ‘Dataset for comparable evaluation of machine translation between 11 south african languages’, *Data in Brief* **29**, 105146.
URL: <https://www.sciencedirect.com/science/article/pii/S2352340920300408>
- Meyer, F. and Buys, J. (2023), ‘Subword segmental machine translation: Unifying segmentation and target sentence generation’.
URL: <https://arxiv.org/abs/2305.07005>
- Mojapelo, M., Buys, J. and Mojapelo, M. (2023), Data augmentation for low resource neural machine translation for sotho-tswana languages (full paper), in A. Pillay, E. Jembere and A. Gerber, eds, ‘Proceedings of the Fourth Southern African Conference for Artificial Intelligence Research’.
- Nagao, M. (1984), A framework of a mechanical translation between japanese and english by analogy principle.
URL: <https://api.semanticscholar.org/CorpusID:18366233>
- Nagata, M. (1994), A stochastic Japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm, in ‘COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics’, Kyoto, Japan.
URL: <https://aclanthology.org/C94-1032>
- Nekoto, W., Marivate, V., Matsila, T., Fasubaa, T., Kolawole, T., Fagbohunge, T., Akinola, S. O., Muhammad, S. H., Kabongo, S., Osei, S., Freshia, S., Niyongabo, R. A., Macharm, R., Ogayo, P., Ahia, O., Meressa, M., Adeyemi, M., Mokgesi-Seling, M., Okegbemi, L., Martinus, L. J., Tajudeen, K., Degila, K., Ogueji, K., Siminyu, K., Kreutzer, J., Webster, J., Ali, J. T., Abbott, J., Orife, I., Ezeani, I., Dangana, I. A., Kamper, H., Elshahar, H., Duru, G., Kioko, G., Murhabazi, E., van Biljon, E., Whitenack, D., Onyefuluchi, C., Emezue, C., Dossou, B., Sibanda, B., Bassey, B. I., Olabiyi, A., Ramkilowan, A., Öktem, A., Akinfaderin, A. and Bashir, A. (2020), ‘Participatory research for low-resourced machine translation: A case study in african languages’.
URL: <https://arxiv.org/abs/2010.02353>
- Nielsen, M. A. (2015), *Neural Networks and Deep Learning*, Determination Press.
 Chapter 2.
URL: <http://neuralnetworksanddeeplearning.com/chap2.html>
- Nirenburg, S. (1989), ‘Knowledge-based machine translation’, *Machine Translation* **4**(1), 5–24.
URL: <http://www.jstor.org/stable/40008396>
- Nong, G., Zhang, S. and Chan, W. H. (2009), ‘Linear suffix array construction by almost pure induced-sorting’, *2009 Data Compression Conference* pp. 193–202.
URL: <https://api.semanticscholar.org/CorpusID:14123069>
- Nyoni, E. and Bassett, B. A. (2021), ‘Low-resource neural machine translation for southern african languages’.
URL: <https://arxiv.org/abs/2104.00366>
- Orife, I., Kreutzer, J., Sibanda, B., Whitenack, D., Siminyu, K., Martinus, L., Ali, J. T., Abbott, J., Marivate, V., Kabongo, S., Meressa, M., Murhabazi, E., Ahia, O., van Biljon, E., Ramkilowan, A., Akinfaderin, A., Öktem, A., Akin, W., Kioko, G., Degila, K., Kamper, H., Dossou, B., Emezue, C., Ogueji, K. and Bashir, A. (2020), ‘Masakhane – machine translation for africa’.
URL: <https://arxiv.org/abs/2003.11529>

- Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D. and Auli, M. (2019), ‘fairseq: A fast, extensible toolkit for sequence modeling’.
URL: <https://arxiv.org/abs/1904.01038>
- Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2002), Bleu: a method for automatic evaluation of machine translation, in ‘Proceedings of the 40th Annual Meeting on Association for Computational Linguistics’, ACL ’02, Association for Computational Linguistics, USA, p. 311–318.
URL: <https://doi.org/10.3115/1073083.1073135>
- Post, M. (2018), A call for clarity in reporting BLEU scores, in ‘Proceedings of the Third Conference on Machine Translation: Research Papers’, Association for Computational Linguistics, Belgium, Brussels, pp. 186–191.
URL: <https://www.aclweb.org/anthology/W18-6319>
- Press, O. and Wolf, L. (2017), ‘Using the output embedding to improve language models’.
URL: <https://arxiv.org/abs/1608.05859>
- Prinsloo, D. J. and de Schryver, G.-M. (2002), ‘Towards an 11 x 11 array for the degree of conjunctivism / disjunctivism of the south african languages’, *Nordic Journal of African Studies* **11**, 249–265.
URL: <https://api.semanticscholar.org/CorpusID:10999691>
- Provilkov, I., Emelianenko, D. and Voita, E. (2020), ‘Bpe-dropout: Simple and effective subword regularization’.
- Rajab, J. (2022), Effect of tokenisation strategies for low-resourced southern african languages, in ‘3rd Workshop on African Natural Language Processing’.
URL: <https://openreview.net/forum?id=SpMeq5M48W9>
- Rikters, M. (2018), Impact of corpora quality on neural machine translation, in ‘Human language technologies–The Baltic perspective’, IOS Press, pp. 126–133.
- Schuster, M. and Nakajima, K. (2012), Japanese and korean voice search, in ‘International Conference on Acoustics, Speech and Signal Processing’, pp. 5149–5152.
- Scott, S. (2002), ‘Bayesian methods for hidden markov models’, *Journal of the American Statistical Association* **97**, 337–351.
- Sennrich, R., Birch, A., Currey, A., Hermann, U., Haddow, B., Heafield, K., Barone, A. V. M. and Williams, P. (2017), ‘The university of edinburgh’s neural mt systems for wmt17’.
URL: <https://arxiv.org/abs/1708.00726>
- Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokry, J. and Nădejde, M. (2017), Nematus: a toolkit for neural machine translation, in A. Martins and A. Peñas, eds, ‘Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics’, Association for Computational Linguistics, Valencia, Spain, pp. 65–68.
URL: <https://aclanthology.org/E17-3017>
- Sennrich, R., Haddow, B. and Birch, A. (2016a), Improving neural machine translation models with monolingual data, in K. Erk and N. A. Smith, eds, ‘Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)’, Association for Computational Linguistics, Berlin, Germany, pp. 86–96.
URL: <https://aclanthology.org/P16-1009>
- Sennrich, R., Haddow, B. and Birch, A. (2016b), ‘Neural machine translation of rare words with subword units’.
URL: <https://arxiv.org/abs/1508.07909>
- Sennrich, R. and Zhang, B. (2019), ‘Revisiting low-resource neural machine translation: A case study’.
URL: <https://arxiv.org/abs/1905.11901>
- Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G. and Dean, J. (2017), ‘Outrageously large neural networks: The sparsely-gated mixture-of-experts layer’.
URL: <https://arxiv.org/abs/1701.06538>

- Spaull, N., Pretorius, E. and Mohohlwane, N. (2020), ‘Investigating the comprehension iceberg: Developing empirical benchmarks for early-grade reading in agglutinating african languages’, *South African Journal of Childhood Education* **10**.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), ‘Dropout: A simple way to prevent neural networks from overfitting’, *Journal of Machine Learning Research* **15**(56), 1929–1958.
URL: <http://jmlr.org/papers/v15/srivastava14a.html>
- Sutskever, I., Vinyals, O. and Le, Q. V. (2014), ‘Sequence to sequence learning with neural networks’.
URL: <https://arxiv.org/abs/1409.3215>
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2015), ‘Rethinking the inception architecture for computer vision’.
URL: <https://arxiv.org/abs/1512.00567>
- Tang, Y., Tran, C., Li, X., Chen, P.-J., Goyal, N., Chaudhary, V., Gu, J. and Fan, A. (2021), Multilingual translation from denoising pre-training, in C. Zong, F. Xia, W. Li and R. Navigli, eds, ‘Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021’, Association for Computational Linguistics, Online, pp. 3450–3466.
URL: <https://aclanthology.org/2021.findings-acl.304>
- Team, N., Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, A., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Barrault, L., Gonzalez, G. M., Hansanti, P., Hoffman, J., Jarrett, S., Sadagopan, K. R., Rowe, D., Spruit, S., Tran, C., Andrews, P., Ayan, N. F., Bhosale, S., Edunov, S., Fan, A., Gao, C., Goswami, V., Guzmán, F., Koehn, P., Mourachko, A., Ropers, C., Saleem, S., Schwenk, H. and Wang, J. (2022), ‘No language left behind: Scaling human-centered machine translation’.
URL: <https://arxiv.org/abs/2207.04672>
- Thompson, B. and Koehn, P. (2019), Vecalign: Improved sentence alignment in linear time and space, in K. Inui, J. Jiang, V. Ng and X. Wan, eds, ‘Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)’, Association for Computational Linguistics, Hong Kong, China, pp. 1342–1348.
URL: <https://aclanthology.org/D19-1136/>
- Tiedemann, J. (2012), Parallel data, tools and interfaces in OPUS, in N. Calzolari, K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk and S. Piperidis, eds, ‘Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)’, European Language Resources Association (ELRA), Istanbul, Turkey, pp. 2214–2218.
URL: <https://aclanthology.org/L12-1246/>
- Torrey, L. and Shavlik, J. (2009), Transfer learning, in E. Soria, J. Martin, R. Magdalena, M. Martinez and A. Serrano, eds, ‘Handbook of Research on Machine Learning Applications’, IGI Global, pp. 242–264.
- van Biljon, E., Pretorius, A. and Kreutzer, J. (2020), ‘On optimal transformer depth for low-resource language translation’.
URL: <https://arxiv.org/abs/2004.04418>
- Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Łukasz Kaiser, Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N. and Uszkoreit, J. (2018), ‘Tensor2tensor for neural machine translation’.
URL: <https://arxiv.org/abs/1803.07416>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2023), ‘Attention is all you need’.
URL: <https://arxiv.org/abs/1706.03762>
- Viterbi, A. (1967), ‘Error bounds for convolutional codes and an asymptotically optimum decoding algorithm’, *IEEE Transactions on Information Theory* **13**(2), 260–269.

- Williams, P., Sennrich, R., Nădejde, M., Huck, M., Haddow, B. and Bojar, O. (2016), Edinburgh’s statistical machine translation systems for WMT16, *in* O. Bojar, C. Buck, R. Chatterjee, C. Federmann, L. Guillou, B. Haddow, M. Huck, A. J. Yepes, A. Névél, M. Neves, P. Pecina, M. Popel, P. Koehn, C. Monz, M. Negri, M. Post, L. Specia, K. Verspoor, J. Tiedemann and M. Turchi, eds, ‘Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers’, Association for Computational Linguistics, Berlin, Germany, pp. 399–410.
URL: <https://aclanthology.org/W16-2327>
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. and Dean, J. (2016), ‘Google’s neural machine translation system: Bridging the gap between human and machine translation’.
URL: <https://arxiv.org/abs/1609.08144>
- Xu, H. and Koehn, P. (2017), Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora, *in* M. Palmer, R. Hwa and S. Riedel, eds, ‘Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing’, Association for Computational Linguistics, Copenhagen, Denmark, pp. 2945–2950.
URL: <https://aclanthology.org/D17-1319>
- Zhou, J., Cao, Y., Wang, X., Li, P. and Xu, W. (2016), ‘Deep recurrent models with fast-forward connections for neural machine translation’.
URL: <https://arxiv.org/abs/1606.04199>
- Zoph, B., Yuret, D., May, J. and Knight, K. (2016), ‘Transfer learning for low-resource neural machine translation’.
URL: <https://arxiv.org/abs/1604.02201>