Machine Translation System for Low Resource Languages

Kurt Abela

Supervisor: Dr Claudia Borg

September, 2022

Submitted in partial fulfilment of the requirements for the degree of Master Of Science.

L-Università ta' Malta
Faculty of Information & Communication Technology

Acknowledgements

I could not have undertaken this project without my supervisor, Dr. Claudia Borg who has provided assistance and expertise. Many thanks goes to Kurt Micallef and my other course-mates for general advice given. I thank all of my lecturers throughout this journey who have helped me through this course at the University Of Malta. I would also like to thank my partner and my family for their constant support.
Thanks also goes to MDIA[1] and UM[2] for helping fund this project.

[1]https://mdia.gov.mt/
[2]https://www.um.edu.mt/

Abstract

Within this study, it was aimed to explore further the translation of low-resource languages, by using techniques that have been shown to work well for high-resource languages and seeing how well they apply to low-resource languages. This study also produced the first open source Neural Machine Translation (NMT) system for Maltese - English translation. Apart from this, NMT systems were also produced for the Icelandic - English language pair. In the case of Maltese-English, a dataset was compiled and published from publicly available sources. In the case of Icelandic - English, the data was retrieved from the dataset compiled for the Workshop on Machine Translation (WMT) 2021 shared task.

This paper proposes five distinct machine translation techniques, two of which are baseline models, while the other three focus on pruning, robustness and multi-sense word translation. Some models were built on the machine translation framework Fairseq, while some models were built using a lower-level API, Tensorflow, thus the need for two baseline models. For the Pruning model, magnitude pruning is used to prune 25% of the neural network and quantization is used to reduce the size of the model further. The Robustness model was trained to recognize and rectify noise in the input data whenever possible. The Multi-Sense Words Improvement model consists of a technique to identify and mine homonyms and their likely translations, and then backtranslate the monolingual data to translate each multi-sense word correctly, according to their context.

The models trained on Fairseq produced much better results than others. The robustness models performed the best in most cases, getting the highest BLEU scores in three of the four directions. Pruning and quantizing models seemed to perform well on Maltese, even surpassing the baseline by 0.4 BLEU in the case of English -> Maltese, however the same can't be said for the English - Icelandic pair, where the BLEU score was less than 1 in both directions. The model for multi-sense word translation seemed to perform slightly better than the baseline as well, however 83% of the backtranslated predictions did not leave space for the multi-sense word to be translated using the implemented technique, so this technique had a low impact. Furthermore, all the code is published to assist future research in this area and can be found on Github[3].

[3] https://github.com/kurtabela/MSc-Thesis

Contents

List of Figures

List of Tables

List of Abbreviations

BART	Bidirectional Auto-Regressive Transformers	11
BLEU	Bilingual Evaluation Understudy	3
BPE	Byte-Pair Encodings	13
CHRF	Character N-gram F-score	24
CLI	Command Line Interface	38
CPU	Central Processing Unit	43
DBSCAN	Density-Based Spatial Clustering of Applications with Noise	46
ELMO	Embeddings from Language Models	10
GPT2	Generative Pre-trained Transformer	10
LM	Language Model	11
MLM	Masked Language Model	11
BERT	Bidirectional Encoder Representations from Transformers	11
mBERT	Multilingual Bidirectional Encoder Representations from Transformers	26
MT	Machine Translation	1
NMT	Neural Machine Translation	iii
OOV	Out of Vocabulary	12
RNN	Recurrent Neural Network	8
SECOCO	Self-Correcting Encoding for Neural Machine Translation	18
SMT	Statistical Machine Translation	6
WMT	Workshop on Machine Translation	iii

Chapter 1

Introduction

Translation between languages have always been fundamental to communicate between societies from different parts of the world. As the novelist and educator George Steiner said, "Without translation, we would be living in provinces bordering on silence."[1]

In the contemporary scene, Machine Translation (MT) is widely utilised by several people, including but not limited to; learners of languages, foreigners in communicating with locals, and to translate content such as social media posts and comments online. Computers can be used to bring people closer together, irrespective of distance and culture, and being able to automatically translate different languages is an important part of this. Indeed, they are already very effective at translating high-resource languages (Popel et al., 2020).

However, translating low-resource languages is more challenging due to the lack of parallel data. Maltese, like Icelandic, is generally considered to be a low-resource language, however when it comes to MT, the situation is improving since it was recognized as an official language of the European Union (EU)[2]. This resulted in all European Parliament proceedings and all official documentation to be translated to Maltese, thus increasing the amount of parallel data available to train MT systems. The caveat is that this data is very specific to the business of the EU, which could result in a high performance of the MT system on this type of jargon but not performing equally well on vernacular that might be used by the general public. For this research, it is intended to use data from multiple sources to cover a wide range of domains.

Besides the limitation regarding lack of data to efficiently train the system, there are still questions over how well techniques adapt when they are trained on high resource languages compared to low resource languages. This research also aims to focus on pruning

[1]https://www.trustedtranslations.com/blog/great-quotes-on-translation
[2]https://european-union.europa.eu/principles-countries-history/languages_en

techniques to limit the size of the neural networks without sacrificing accuracy. This is done with the intention of making this study more accessible and thus able to be replicated and improved. Overall, the main aims of this study are to highlight specific issues that are faced when training a neural MT model on low-resource languages and to identify potential solutions or mitigations.

1.1 Motivation

With the current improvement of performance levels of MT, the gap between human and machine translation is constantly diminishing (Popel et al., 2020). As reported by Taira et al. (2021), even in fields where there is generally less data available, such as in the field of medicine, it is seen that translation accuracy is improving. Taira et al., a team of doctors, reported that Google-generated Spanish translations of medical instructions had an accuracy of 94%. However, they also stated that the results were as low as 55% for lower-resource languages such as Armenian. Here, one can see the discrepancy between high-resource languages such as Spanish and lower resource ones, where the performance can vary quite drastically. Therefore, the motivation is to use the latest technologies and architecture to attempt to bridge the gap between the performance in these languages, with a specific focus on the Maltese language.

1.1.1 Aims and Objectives

The main goal of this thesis is to create multiple translation systems that can be used to translate words, sentences and documents for low resource languages, namely the Maltese - English and Icelandic - English language pairs. The main focus will be on the Maltese - English pair of which, to the best of our knowledge, does not have any neural network MT systems trained on a publicly available dataset that we can use to compare our systems with. There are a number of commercial systems trained on this language pair, however they are a *black box*, where we can't compare architectures or datasets. Hence, the intention is to explore different techniques and create different systems to see which systems appear to work best and should be explored further in future research.

In order to achieve this aim, the following objectives are defined:

- **O1** Putting together a dataset from varied sources available to experiment with MT performance when there is more data variety.
- **O2** Creating a baseline MT system built using a standard transformer architecture.

- **O3** Creating different MT systems with different techniques to analyze their performance in comparison with the baseline MT system.
- **O4** Evaluating the system and comparing it with similar systems trained on low-resource languages.
- **O5** Publishing the source code of this study, as well as the compilation of publicly available datasets to assist further research in this area.

1.1.2 Why the project is non-trivial

Machine translation is generally regarded as an unsolved problem, especially for low-resource languages as it is not a realistic assumption to acquire the large amounts of parallel data necessary to train a decent MT system (Ranathunga et al., 2021). This study is non-trivial due to the fact that Maltese has a lack of digital data, with most native speakers choosing to use Maltese to speak and English to write, meaning most of the content online, such as on social media posts and comments is in English[3]. In fact, this presents a challenge for the data collection stage.

It has also been observed that most of the Maltese content written in social media is written without using the Maltese font and syntax, and often written in an informal manner, with little regard to the syntax of the sentence. This means that the data collection will have to be a manual process without the ability to scrape a lot of content directly from, for example, social media websites. Generally, posts or articles from well known websites or newspapers however are usually written in the right syntax, but this limits our available pool of data, as well as the fact that a number of local websites write exclusively in English.

It is expected to find issues that are not commonly found when working on high-resource languages. These include added noise in our data since we have limited datasets to choose from, as well as that words that have multiple senses (known as homonyms) are usually incorrectly translated due to the lack of representative data of the different senses.

Another issue is with the evaluation of the system, with the most standard metric, Bilingual Evaluation Understudy (BLEU) being susceptible to impeding the development of improved models that are solely evaluated on BLEU (Kocmi et al., 2021). However, most evaluation systems are nowadays scored via the BLEU metric, so we must also use BLEU scores to be able to compare systems.

[3] `https://timesofmalta.com/articles/view/maltese-vs-english.4604`

Since there are very limited high quality parallel Maltese datasets publicly available, to the best of our knowledge, there is also no NMT system to directly compare and evaluate our system with. This presents a challenge since there is no way to analyse the architecture with an existing one, making it impossible to see which solutions improved performance and which direction future research should focus on. This does not mean that there are not any datasets in Maltese, however most of them are not manually created or verified by human translators, as seen in Section 3.1.

1.2 Main contributions of the work

The main contributions of this project are the following:

- Presenting a summary of the most important literature in regards to MT.
- Compiling a list of available datasets for MT for the Maltese - English language pair. This also includes scoring random samples from each dataset, to give an indication of which datasets appear to be of the highest quality.
- Creating baseline techniques on this dataset, using different libraries with different backends. Different libraries may have slightly different optimizations in the backend. These baselines will be important for future researchers to have a starting point to refer to.
- Creating a model that prunes the neural network by reducing the number of weights, while still retaining the same performance level as the baseline model. The model is furthermore quantized so as to reduce the size of the model by decreasing the precision of the weights.
- Creating a model that is trained to be robust to noise, especially due to the fact that most low-resource languages don't have a lot of public datasets available. This implies that the quality of the datasets used might be lower than those used in higher-resource languages, where more options may be present.
- Creating a model to better predict multi-sense words in low-resource languages. For this model, we used a clustering technique to detect multi-sense words. Then, we mine the translations of these homonyms using two language models, of the source and the target language. Finally, monolingual data is backtranslated using the mined translations for the multi-sense words.

- Reporting the results of five different MT systems, including the baseline models and others built on top of the baseline models. Since there are two language pairs and each model is trained in both directions, this results to 20 distinct models. On our dataset, a model trained on robustness appeared to perform the best in almost all directions.
- Future researchers can analyze and build on the proposed solutions as the source code of this study will be published and freely available.

1.3 Document Layout

This chapter presented an introduction to the topic at hand, while the following section, Section 2, goes over the literature review of MT, including the different techniques used for MT as well as a historical analysis of MT in the Maltese context. Section 3 then goes over the methodology, including the different MT models developed for a number of different problems. Next is Section 4, that goes over the results of each model and in which the evaluation metrics considered are discussed along with a comparison of our results with other published research. Finally, Section 5 contains some concluding remarks as well as the potential directions for future work.

Chapter 2

Background & Literature Overview

In this chapter, different extant MT approaches proposed in literature are reviewed. The different systems are compared by investigating the results obtained, datasets used to train and evaluate the system on, as well as architecture and methodology used.

MT systems were theorized all the way back in 1949, when Warren Weaver thought about using cryptographic techniques, that were used in the Second World War, to translate real-life languages (Weaver, 1952). These early systems were primitive and rule-based and heavily based on statistics. Nowadays, in MT there are 2 main approaches, namely Statistical Machine Translation (SMT) & NMT.

This chapter will be investigating how the results in this field improved throughout the years, particularly during the transition period from SMT to NMT.

2.1 Statistical Machine Translation

Before 2014, SMT systems was the main approach to translate a sequence. This was introduced by Brown et al. (1990), who developed the basis of SMT: every sentence S in a source language has a possible translation T in a target language. Building upon this view, one can claim that there is a probability $P(T|S)$ of T being the correct target translation of S. $P(T)$ is known as the language model, and is necessary to get a well-formed sentence. SMT can be further split in two sections, namely Phrase-based SMT and Word-based SMT.

2.1.1 Word-based SMT

According to Brown et al. (1988), translation needs to be based on a complex glossary of the corresponding target words and phrases that are placed in fixed locations (in vector space). The authors proposed a three step approach to solve this task:

1. Place the source text into vector space (known as the set of fixed locations).
2. Use the glossary combined with contextual information to select the target text, based on the corresponding set of fixed locations in the target language.
3. Place the words of the target fixed location into a coherent sequence to form the translation.

Vogel et al. (1996) expanded on Brown et al.'s work and proposed a new alignment model based on a hidden markov model (HMM). This alignment is used to train the system to take into consideration the localization effect between specific languages. This generally results in increased performance for languages that are similar to each other in some way, such as Indo-European languages.

Following this, alignment models were heavily researched to have sentences that were grammatically correct and coherent. Och and Ney (2000) proposed an annotation scheme, that made it possible to explicitly annotate which were the alignments the system is most confident of and which were the less confident (ambiguous) ones. This meant that human annotators can see at a glance which phrases the system is not confident about and arrange them quickly.

2.1.2 Phrase-based SMT

Word-based SMT systems were revolutionary and in many ways were the first step forward towards achieving machine translation systems as we know them today. However, they had multiple issues especially when it came to coherency of the translated sentence. The results of the translations did help the reader get an idea of the original sentence, but the systems failed to deal with case, gender and homonymy (words that have different meanings but are pronounced the same or spelled the same or both[1]). In most cases the translations were word-for-word translations, which caused these issues.

Phrase-based SMT systems aimed to solve these issues, by removing the restriction of translating a source sequence into the target sequence by translating each word individually (Koehn et al., 2003). Instead, lexical units are introduced that may be a sequence of words (that may have any length) as opposed to single words in word-based SMT systems. Each unit is paired with another (one has to be from the source language and the other has to be from the target language) to get a score/weight associated with it.

[1]`https://dictionary.cambridge.org/dictionary/english/homonym`

Although phrase-based SMT systems improved word-based models, they still had issues of their own, including the inability to reorder source sequences when long distances are involved. The field of MT continued to evolve alongside advances in the field of Natural Language Processing in general. As a consequence, MT techniques also began to shift towards neural approaches. In the following section, NMT systems will be reviewed.

2.2 Neural Machine Translation

In 2014, NMT started to emerge as a popular alternative to SMT (Cho et al., 2014b; Sutskever et al., 2014), as it allowed engineers to build a single, deep neural network that was able to simply receive a sentence as input and return a translation back.

2.2.1 Attention

Prior to Bahdanau et al. (2014), standard seq2seq models worked by having an Recurrent Neural Network (RNN) as an encoder. This encoder works by creating a set of hidden vectors, where each hidden state is determined by the current word and the previous hidden state. The output of the encoder is known as the context vector, which in this case used to be determined as the final hidden vector. Unfortunately this meant that earlier tokens end up being forgotten by the time the final word token is reached.

Bahdanau et al. (2014) introduced an extension to the popular encoder-decoder model, nowadays known as the 'attention' mechanism. They proposed that firstly, a context vector should be created at every step of the translation sequence. Secondly, the context vector is not one of the hidden states, but rather a weighted combination of all hidden states. This architecture is able to align and translate sentences jointly, allowing the model further context when generating the target words.

These mechanisms are used to learn soft word alignments between language pairs. Nowadays, these attention mechanisms have become a staple in MT models, especially in Transformers (Vaswani et al., 2017). Transformers have a critical component called the Self-Attention Network (SAN), which is a very similar concept to regular attention. This component allows the inputs to interact with each other to decide which token should each input token pay more 'attention' to.

Nonetheless, it has been observed that the self-attention models contain a lot of redundant information that can be pruned, as well as that Transformer models in general are over-parameterized (Correia et al., 2019; Michel et al., 2019a; Sanh et al., 2020; Voita et al., 2019). Additionally, these papers suggest possible performance improvements if

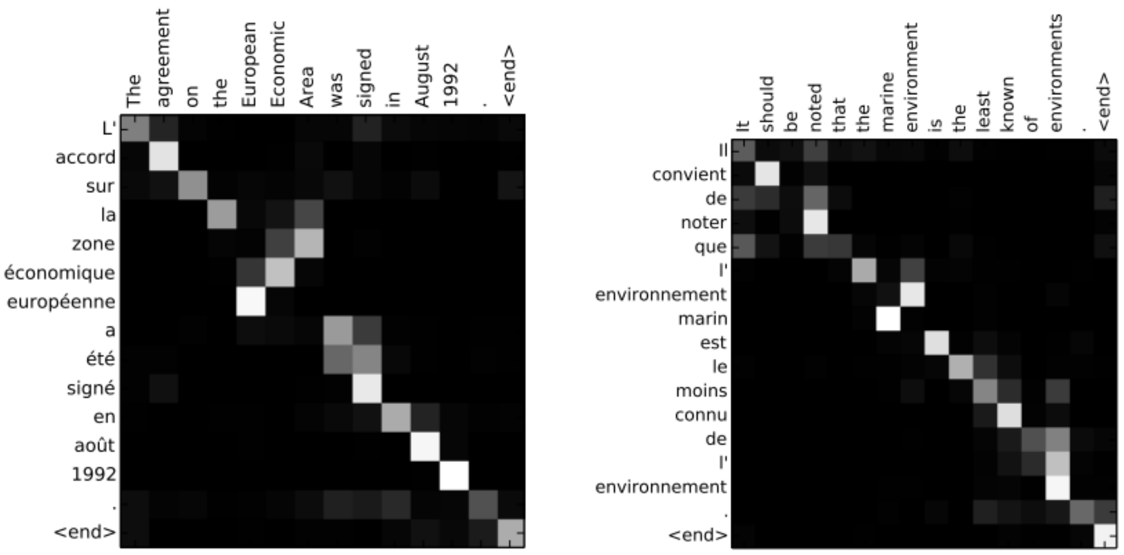


Figure 2.1: Two examples of the attention mechanism working and aligning words from different languages (Bahdanau et al., 2014)

techniques are developed to decrease the computation necessary for SAN, as well as potentially replacing it completely with fixed or learnable global attention patterns.

Tay et al. (2021) propose a method to learn synthetic attention weights without token-to-token interactions. This method, called Synthetic Attention, generates the alignment matrices by using a set of parameterized functions for synthesizing attention matrices. Essentially, this means that the attention matrices are trainable and are trained with the rest of the model parameters. The authors show that on multiple tasks (including machine translation) the proposed Synthetic Attention technique achieved very competitive results when compared to a vanilla self-attention model. In addition to this, the researchers noted that they did not manage to replace cross-attention with simpler techniques in most cases, making this an interesting area for future work.

Zeng et al. (2021) also came up with their own proposal to fix this issue by exploring a substitute for self-attention, named Recurrent Attention (RAN). RAN can learn the attention weights without any token-to-token interaction. Essentially, RAN takes the initial attention matrix and updates the weight via a feed-forward network known as the Recurrent Transition Module. They explain that this is important since recent research shows that perhaps the self-attention in Transformers mainly learn simple positional patterns, and don't understand the deeper meaning of the sentence. In fact, Xiao et al. (2019) and He et al. (2021) observe that attention weights seem to show a regular pattern and have certain correlation between layers. Their results showed that indeed there is an increase of about 0.5 BLEU on 6 translation tasks when RAN is applied to the decoder of a Transformer, and even increased by 1.0 BLEU on the Turkish-English translation task. Moreover, Zeng et al. state that future work could be done to apply RAN to cross-attention too.

2.2.2 Pre-trained Language Models

By 2018, the main focus was placed on gathering and collecting data for NMT. Devlin et al. (2019) noted that to apply pre-trained language representations to downstream tasks such as machine translation, there are two solutions, namely using a feature-based or a fine-tuning approach.

An example of a feature-based approach is Embeddings from Language Models (ELMO) (Peters et al., 2018). Feature-based approaches use the pre-trained representations as additional features and instead rely on architectures that are designed specifically for the task in mind. Pre-trained word embeddings are nowadays an integral part of NLP, since they save a lot of time over training embeddings from scratch (Turian et al., 2010). ELMO works by extracting the context sensitive features from two language models, one of which goes from left to right and the other goes from right to left. It is important to note that this architecture is still feature based and not deeply bidirectional.

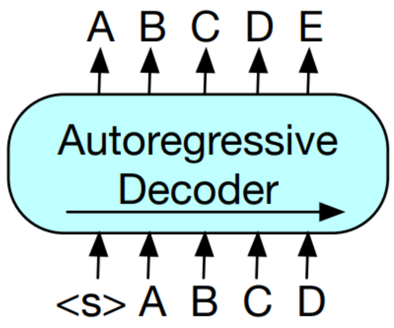


Figure 2.2: Overview of GPT, where an auto-regressive decoder is used but the text can only be processed in one direction (Lewis et al., 2019)

On the other hand, fine-tuning architectures like the Generative Pre-trained Transformer (GPT2) (Radford et al., 2018) simply fine-tune the pre-trained representations for their tasks and add a small number of task-specific parameters if necessary. These are mainly used in sentence or document encoders that produce contextual representations of these sentences or documents. These systems are pretrained on unlabeled text and then fine-tuned on the specific task that is being worked on (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). Figure 2.2 shows an overview of GPT2.

While these techniques are a step in the right direction, Devlin et al. argue that both of these limit power of the pre-trained representations due to the fact that these standard language models are unidirectional, and thus the architectures have to be unidirectional too. This means that tokens can only attend to previous tokens and tokens on the other side. To solve this issue, they propose an architecture called Bidirectional Encoder Representations from Transformers (BERT), that removes the previous constraint that a Language Model (LM) can only read left to right (or vice versa) by using a Masked Language Model (MLM). This MLM works by masking some of the original tokens and then trained to predict these tokens just by their context. This enables the representations to fuse left and right contexts, which makes it possible to pretrain a bidirectional model. This architecture is widely considered a landmark in the field of NLP, as it also achieved new state-of-the-art benchmarks in eleven tasks. An overview of BERT can be seen in Figure 2.3

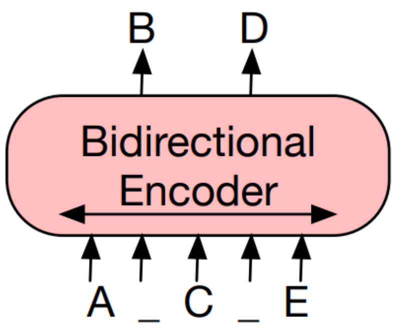


Figure 2.3: Overview of BERT, where a bidirectional encoder replaces tokens randomly with masks (Lewis et al., 2019)

In 2019, an alternative to BERT was introduced, called Bidirectional Auto-Regressive Transformers (BART) (Lewis et al., 2019). As the authors write, BART is meant to generalize BERT and GPT2 (Radford et al., 2019). They state that while BERT is useful for tasks where the prediction at a certain point in the sequence can use the rest of the sequence, it's performance diminishes for predictive tasks like text generation where a prediction at a certain point in the sequence can only be made from the inputs before the current point. On the other hand, GPT2 is good at predictive tasks such as text generation and is not as good as BERT for tasks where the whole sequence (except for the token to be predicted) is available as input. BART is intended to incorporate features so that it can be used in both cases, by having an encoder with an attention mask being fully visible (like BERT's) and a decoder with an attention mask being casual (like GPT2's). These are then connected via cross-attention, where the final hidden state of the encoder has attention applied on it by the decoder. This can be seen in Figure 2.4.

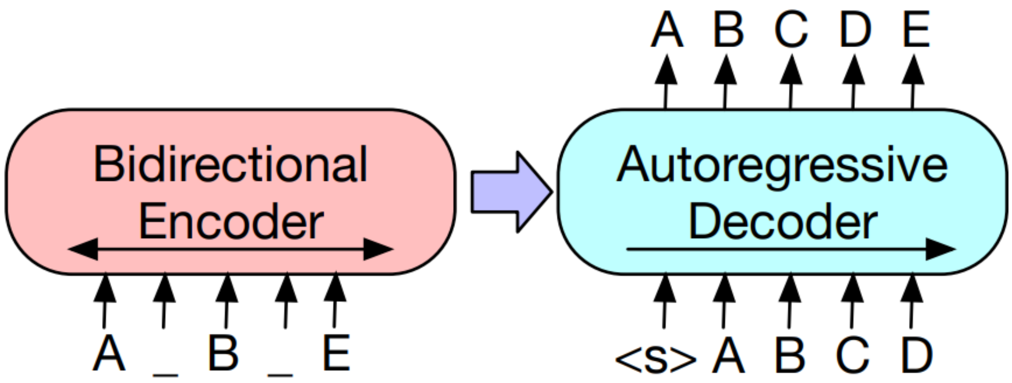


Figure 2.4: Overview of BART, which combines the architectures of BERT and GPT (Lewis et al., 2019)

2.2.3 Tokenizers

Before sentences can be passed to the NMT model, they need to be preprocessed. This generally includes padding the sequences so they are all equal lengths, and tokenizing them. Tokenization is necessary in all computational text analysis and involves splitting the text into chunks. NLP systems process the raw text from these tokens and thus it is important to choose the right way to tokenize the source text. In the case of machine translation, these tokens are used to help determine the vocabulary of the corpus, and anything that is not in the vocabulary is treated as Out of Vocabulary (OOV). There are three main techniques to do this, namely word-level tokenizers, sentence-level tokenizers, and -word level tokenizers.

Word-level tokenizers split the text into individual words, but this is not always possible in certain languages such as in Mandarin, where there is no real clear distinction of words. It is however one of the most popular technique due to its simplicity in languages where words are separated via a space. However, this can also lead to issues such as where a contraction is used (e.g. “don’t"). If the words are split based on space then the phrase is treated as an individual word in the corpus, completely separate from “do" and “not". A similar issue occurs with punctuation at the end of the word, where the word “right?" is treated as an individual token again, completely independent from “right". Another drawback of this technique is that if the input data contains noise such as misspellings then once again multiple tokens might be created when in reality they all refer to the same word.

Character-based tokens is an alternative technique that aimed to solve the issues mentioned above by creating a token of each individual character rather than a token of each word. This also results in a much smaller vocabulary to keep track of since there is a limited number of characters in a language. Using character based tokens implies that there will be no (or few) OOV words since it can generate words it has never seen before from the list of characters seen. It can also gain valuable information from noisy inputs such as misspellings. This type of tokenization is very useful in certain languages especially that contain a lot of information in each character. In English and Maltese this is not so much the case as the characters we use don't usually contain a lot of information on their own. Apart from this, having character tokenization leads to large tokenized sequences since each word is split into each character.

An improvement over these traditional tokenization techniques is known as subword tokenization. The idea is that this is a compromise between word and character tokenizations and thus solves the issues of both techniques. There are two main principals when it comes to subword tokenization:

1. Frequently used words should be left as they are, without splitting them into smaller words.

2. Rare words should be split into smaller words where appropriate.

These principles imply that words like "girls" should be split into "girl" and "s". It would also allow the system to learn more about suffixes, such as "standardization" and "tokenization" have the same suffixes ("ization") and as such, the model could learn where these suffixes are commonly utilized, even on words it has not seen during training. This makes subword tokenizers very powerful. Nowadays, models come shipped with subword tokenizers, such as Byte-Pair Encodings (BPE) (Sennrich et al., 2015b) which is used by GPT-2 (Radford et al., 2019) and RoBERTa (Liu et al., 2019), and BERT (Devlin et al., 2019) which uses WordPiece (Wu et al., 2016). Figure 2.5 shows how BERT tokenizes it's inputs. In reality, three different embeddings are used, namely token embeddings (WordPiece), segment embeddings (to distinguish between sentences if there are multiple) and position embeddings (to preserve information regarding the position of the tokens).

Given that this project will mainly focus on the Maltese, Icelandic and English languages, it does not make sense to utilize character-level embeddings due to the fact that in these languages, there is not a lot of information stored at character level. Given the issues with word-level tokenization above, it makes the most sense to use a subword tokenizer for our use case. This especially makes sense in Maltese, where a word can have different information conjugated together at different points in a sentence. For example, the Maltese word 'Nagħtihomlu' can be split in five subwords, namely 'agħti' which means 'to give', 'n' before this which implies that 'I' will be giving, 'hom' which means 'them' and 'lu' which means 'to him'. Therefore, this one word means that "I will give them to him",

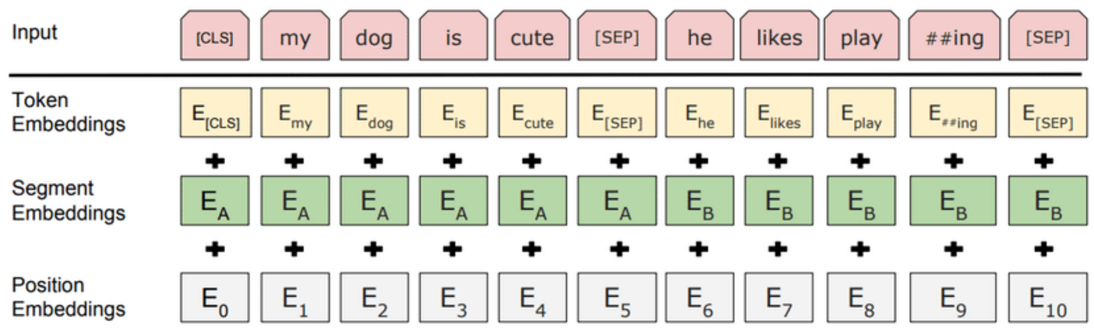

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.5: WordPiece as used in BERT (Devlin et al., 2019)

which would be ideal for a subword tokenizer to figure out and capture these details in similar words.

An important hyperparameter to be passed is the vocabulary size. Setting the vocabulary size to be too big could cause memory issues since the number of dimensions of the embedding increases too. Setting it too low would mean that not enough information will be learnt, thus an ideal value has to be found that is not too low and not too high.

2.2.4 Monolingual Data

Until 2015, NMT was performed by using only parallel sentences and documents for training. This proved an issue for low-resource languages that did not have a large number of parallel data available. Sennrich et al. (2015a) and Gulcehre et al. (2015) argued that target-side monolingual data could still play an important part to improve fluency for NMT, which is important since usually the amount of monolingual data is typically a lot more than the amount of parallel data. Gulcehre et al. (2015) managed to increase the BLEU score by almost 2 on Turkish-English translations, which at the time was considered as a low resource language pair, as well as Chinese-English translations. The architecture used is the same as the one used in Sennrich et al. (2015a), both based on Chorowski et al. (2015), but they state that their approach is not specific to this architecture. The NMT system used is a standard encoder-decoder network with recurrent neural networks (RNNs). In summary, the encoder is a bidirectional neural network that includes gated recurrent units (GRU) (Cho et al., 2014a), whereas the decoder consists of a RNN.

Gulcehre et al. trained the monolingual LMs separately, and then integrated them in the decoding process. This is done by extending the decoder to also contain a hidden state from the LM. Then, the magnitude of the LM signal is controlled by an additional controller mechanism called 'deep fusion'. The parameters of the controller are further tuned from more parallel training data, where the LM parameters are still fixed at this stage. An alternative to this is known as 'shallow fusion', that is to integrate the states through the rescoring of the beam when using the beam search algorithm. Both of these techniques are used to integrate the separate monolingual language models into the decoder stage. As can be seen in Table 2.1, both techniques outperformed the baseline NMT model on the WMT'15 dataset on different languages.

Results of Shallow & Deep Fusion against the WMT'15 Baselines		
Model	**German - English**	**Czech - English**
NMT Baseline	23.61	21.89
Shallow Fusion	23.69	22.18
Deep Fusion	24.00	22.36

Table 2.1: Results of Shallow & Deep Fusion against the WMT'15 Baselines as reported by (Gulcehre et al., 2015)

Sennrich et al. (2015a) go into detailed technical explanation of the system, explaining clearly the difference between their system and previous systems that also focused on using monolingual data, including Gulcehre et al. (2015). The authors delve deeper into the 2 main techniques used to integrate the monolingual data with the parallel data, namely by providing the data with an accompanying dummy (or empty) source sentence or a synthetic source sentence obtained directly using back-translation. The former technique implies that the system has to fully rely on the previous target words to predict the translation. This also implies that the system is training for two tasks at the same time, namely machine translation and language modelling (in the case where the source side is empty). This is known as multi-task learning. Interestingly, the ratio of monolingual data to parallel data was kept 1-to-1 during training. The second technique is used in conjunction with the first to ensure that the output layer still pays attention to the source side. The source side sequences are instead obtained via back-translation, where the monolingual target text is automatically translated into the source language.

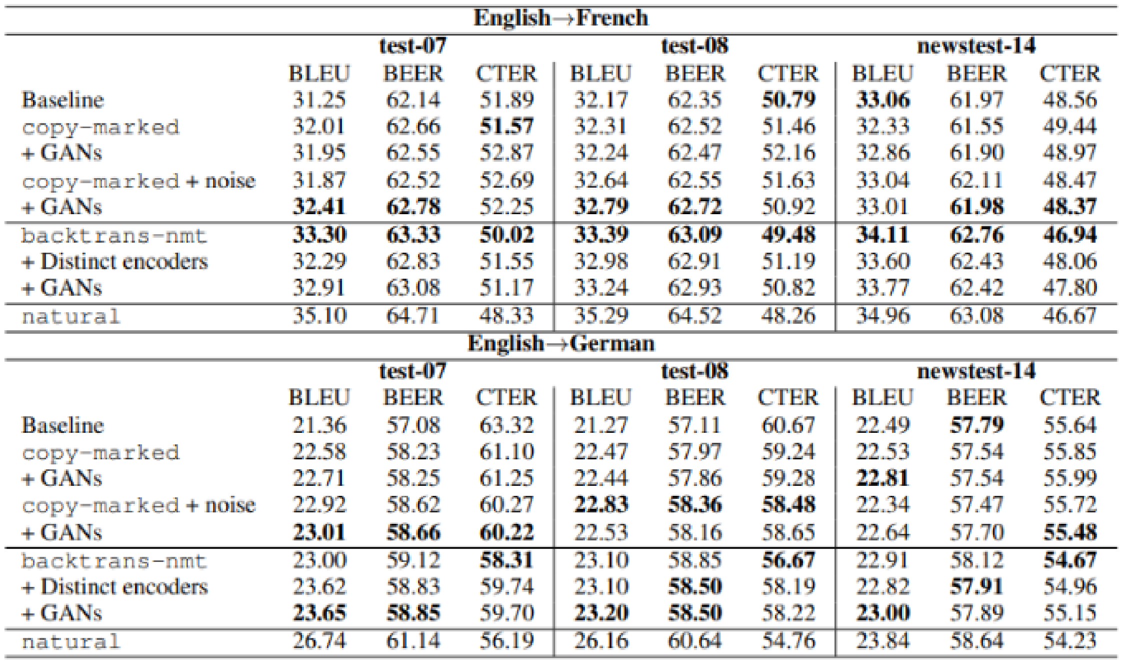
Unfortunately, the official code was not shared or published. It would have been easier to investigate the results and methodology had this been available too.

The authors evaluate their results on different translation pairs, including English to German and vice versa, as well as Turkish to English. The BLEU scores improve by over 20% when compared to the best model of Gulcehre et al. (2015) on all translation pairs.

Although Gulcehre et al. (2015) were the first to use monolingual data in theirMT system, Sennrich et al. propose a novel technique to use monolingual data to train the main NMT model, unlike Gulcehre et al. who train separate models and then incorporate them together. This makes the process simpler as only one model needs to be trained.

It is important to note that although using monolingual data helps improve the translation system, Sennrich et al. noted that at some point, the translation performance actually decreases when the size of the synthetic data is too large compared to the size of the original parallel data.

More recently, a systematic study was published (Burlot and Yvon, 2019), that showed that even today, backtranslation using an NMT seems to work the best in most cases. The results of this study can be seen in Figure2.6



English→French									
	test-07			**test-08**			**newstest-14**		
	BLEU	BEER	CTER	BLEU	BEER	CTER	BLEU	BEER	CTER
Baseline	31.25	62.14	51.89	32.17	62.35	**50.79**	**33.06**	61.97	48.56
copy-marked	32.01	62.66	**51.57**	32.31	62.52	51.46	32.33	61.55	49.44
+ GANs	31.95	62.55	52.87	32.24	62.47	52.16	32.86	61.90	48.97
copy-marked + noise	31.87	62.52	52.69	32.64	62.55	51.63	33.04	62.11	48.47
+ GANs	**32.41**	**62.78**	52.25	**32.79**	**62.72**	50.92	33.01	**61.98**	**48.37**
backtrans-nmt	**33.30**	**63.33**	**50.02**	**33.39**	**63.09**	**49.48**	**34.11**	**62.76**	**46.94**
+ Distinct encoders	32.29	62.83	51.55	32.98	62.91	51.19	33.60	62.43	48.06
+ GANs	32.91	63.08	51.17	33.24	62.93	50.82	33.77	62.42	47.80
natural	35.10	64.71	48.33	35.29	64.52	48.26	34.96	63.08	46.67
English→German									
	test-07			**test-08**			**newstest-14**		
	BLEU	BEER	CTER	BLEU	BEER	CTER	BLEU	BEER	CTER
Baseline	21.36	57.08	63.32	21.27	57.11	60.67	22.49	**57.79**	55.64
copy-marked	22.58	58.23	61.10	22.47	57.97	59.24	22.53	57.54	55.85
+ GANs	22.71	58.25	61.25	22.44	57.86	59.28	**22.81**	57.54	55.99
copy-marked + noise	22.92	58.62	60.27	**22.83**	**58.36**	**58.48**	22.34	57.47	55.72
+ GANs	**23.01**	**58.66**	**60.22**	22.53	58.16	58.65	22.64	57.70	**55.48**
backtrans-nmt	23.00	59.12	**58.31**	23.10	58.85	**56.67**	22.91	58.12	**54.67**
+ Distinct encoders	23.62	58.83	59.74	23.10	**58.50**	58.19	22.82	**57.91**	54.96
+ GANs	**23.65**	**58.85**	59.70	**23.20**	**58.50**	58.22	**23.00**	57.89	55.15
natural	26.74	61.14	56.19	26.16	60.64	54.76	23.84	58.64	54.23

Figure 2.6: Results of different techniques to utilize monolingual data (Burlot and Yvon, 2019)

2.2.5 Robustness

Research on robustness in NMT can be classified into two main categories. The first category is using adversarial examples that can be generated using either black-box or white-box methods. These are combined with the original training data for adversarial training (Ebrahimi et al., 2018; Michel et al., 2019b). Karpukhin et al. (2019) generated adversarial examples using synthetic noise and studied Transformers' behaviour in the presence of noise. The second category is research that focuses on Fine-tuning (FT). FT is when noisy versions of input tokens are used as part of the training set, so that the decoder learns to output the correct translations despite the intentional noise. Helcl et al.

(2019) and Dabre and Sumita (2019) both work on FT to study its impact and how best to utilize it.

Passban et al. (2020) wrote a paper revisiting robustness for neural machine translation, which is the study of how noise can break down Transformers specifically. The authors introduce 3 concepts to help Transformers be more resistant to noise, to not only identify noisy inputs but also to rectify the noise. The first concept they introduce is Target Augmented Fine-Tuning (TAFT), which is a learning-rate scheduler of fine-tuning, where noisy versions of input tokens are fed to the system. This is a data augmentation technique. The second concept is Controlled Denoising (CD), where during training, noise is added to the source sequences and the encoder is trained to fix these noisy inputs before actually feeding them to the decoder. This concept is similar to the adversarial training and can be seen in Figure 2.7.

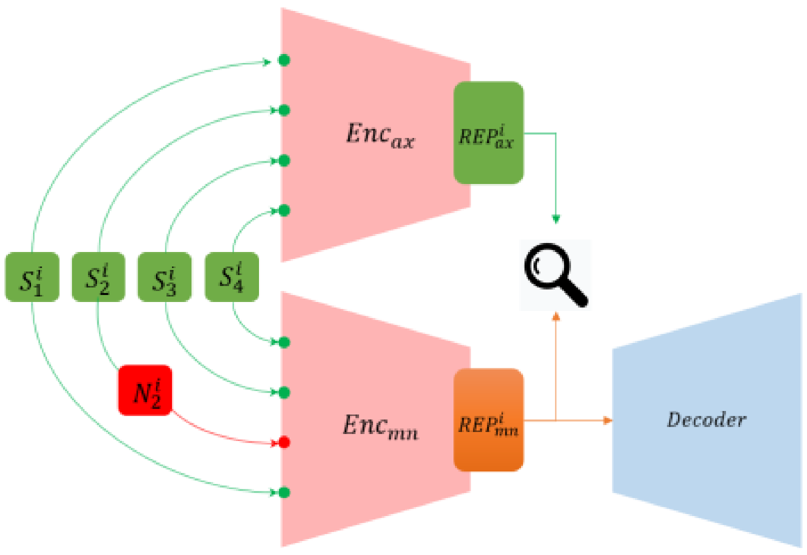


Figure 2.7: Controlled Denoising as implemented by Passban et al. (2020), where four tokens that are part of sequence *i* are given to the main and auxiliary encoders, yet one encoder is given a noisy version of the second token. To ensure that the main encoder can handle potentially noisy inputs, the two representations are compared via a loss function.

The final contribution is Dual-Channel Decoding (DCD), which is a dual-channel decoder that can select target outputs and correct the noisy inputs simultaneously. Overall, these results improved the performance of their systems, especially multi-tasking since the transformers seemed to benefit a lot when the decoder is informed about the noise from the source sequence.

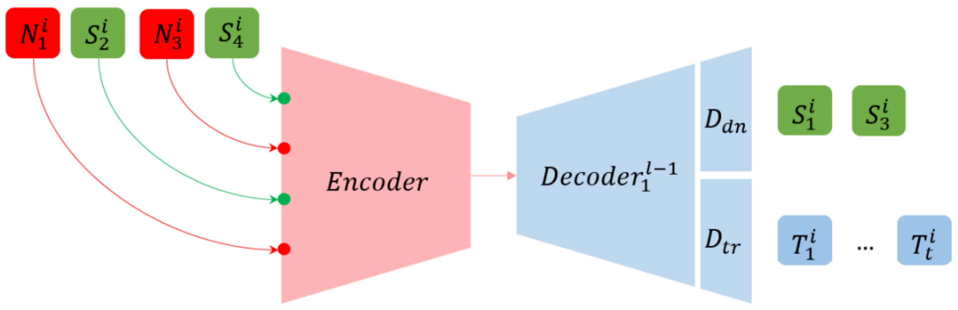


Figure 2.8: Dual-Channel Decoding as implemented by Passban et al. (2020). One can see how the decoder is a dual-channel decoder where one decoder (D_{dn}) generates the (fixed) noisy tokens detected and the other decoder (D_{tr}) actually translates the input sequence.

Sánchez-Cartagena et al. (2021) evaluated data augmentation techniques for MT, which ties in with the TAFT component introduce by Passban et al. However, Passban et al. only used natural noise for their data augmentation, whereas Sánchez-Cartagena et al. expanded on synthetic noise as well. The authors mention 6 methods to create synthetic noise. These include randomly swapping words in the sentence, removing words and replacing them with a special 'UNK' token, copying the target sentence and making it the source sentence, reversing the target sentence and using that output as the source sentence, aligning the meaning of the words in the source with the meaning of the words in the target language (so the target side becomes non-fluent) and replacing words by random entries from a bilingual lexicon (obtained from the training corpus). The authors state that these techniques should help the system reduce the reliance on the decoder and strengthen the encoder, which forces the decoder to pay more attention to the encoder representation.

Wang et al. (2021) proposed a framework called Self-Correcting Encoding for Neural Machine Translation (SECOCO). This framework works on making NMT models more robust by introducing self-correcting predictors to deal with input noise. SECOCO not only detects noisy inputs, but it also has the option to rectify the noise while simultaneously translating the original sequence without the noise in it. This makes this project different to the approaches proposed by Sánchez-Cartagena et al. (2021) and Passban et al. (2020) since SECOCO enables NMT to explicitly correct noisy inputs and delete errors. The authors propose two predictors to learn noise correction, namely an insertion predictor and a deletion predictor. These were trained by randomly deleting and inserting tokens to original clean data. The architecture can be seen in Figure 2.9, where 'x' is the source sentence that has had noise added and some tokens removed, to train the predictors. During the inference stage, two different variants are proposed. The first being an end-to-end approach where the encoder is trained to self-correct the input sequence. The second variant follows an iterative process, by iteratively editing the input. The translation only occurs when the input stops being edited. SECOCO has two main advantages that makes it useful to incorporate a similar system into our project, namely being that it provides a direct and explicit way to model the process of correcting noise, as well as that it enables an interpretable translation process.

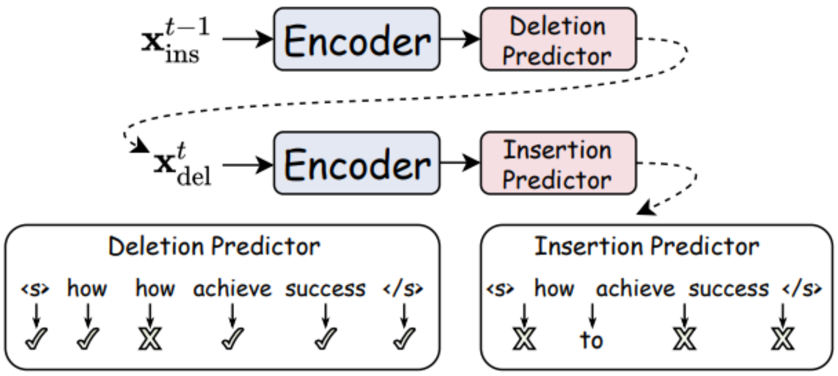

$\mathbf{x}_{\text{ins}}^{t-1}$
Encoder
Deletion Predictor
$\mathbf{x}_{\text{del}}^{t}$
Encoder
Insertion Predictor
Deletion Predictor
<s> how how achieve success </s>
Insertion Predictor
<s> how achieve success </s>
to

Figure 2.9: Overview of SECOCO (Wang et al., 2021)

2.2.6 Pruning

Thompson et al. (2020) wrote about how the recent successes of deep learning seem to require performance-intensive architectures. The authors report that in five of the studied applications, one of which is machine translation, progress in the area is strongly reliant on increases in computing power. Moving forward, this is not economically, technically or environmentally feasibly, as seen in Figure 2.10.

Nowadays, to combat this problem, more research is being put in pruning techniques. Gale et al. (2019) compressed a ResNet-50 model using magnitude pruning (Zhu and Gupta, 2017). Magnitude pruning is a way to remove the least important weights. The importance of the weights are estimated by taking the absolute weight value and removing the weights with the least importance at specific intervals. The intervals are predetermined by a schedule that dictates the training steps and frequency between the begin and end step across which sparsity is introduced. This allows the developers to change the rate of sparseness and investigate by trial and error which sparsity percentage yields the best results. This technique consistently achieves comparable or better results than the standard state-of-the-art architectures, as demonstrated by Gale et al.

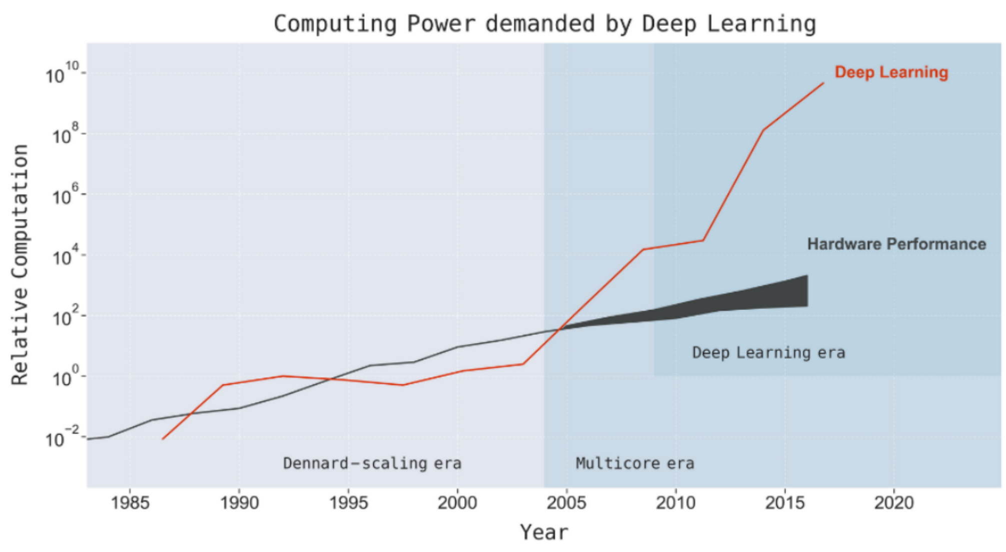


Figure 2.10: Graph showing the relative computation power required through the years as measured by Thompson et al. (2020)

(2019). The authors managed to achieve comparable results in the fields of computer vision and language models. Similar research was done by various authors on different large scale datasets (Li et al., 2020; Narang et al., 2017; Zhang et al., 2017), however all of these works focused mainly on large scale datasets.

See et al. (2016) provided a proof of concept using a RNN to translate Vietnamese to English and vice-versa. This dataset is much smaller than the ones previously used, which proved to be a new challenge since there is a limited amount of data. The authors experimented with three magnitude-based pruning techniques namely class-blind, class-uniform and class-distribution. Each parameter can be attributed to a class of weights, including softmax weights, attention weights, source embedding weights, target embedding weights as well as source and target hidden layers. The class-blind technique takes all the parameters, sorts them by magnitude and finally prunes the *x*% of parameters with the smallest magnitude, regardless of the weight class. The class-uniform technique removes the *x*% of parameters with the smallest magnitude in each class, so each class has *x*% parameters removed. The final technique, class-distribution works by pruning *x*% parameters in total, but the weights are removed in accordance to the standard deviation of the class. Therefore, for each class the weights with a magnitude less than the standard deviation of that class multiplied by λ, where λ is a universal parameter chosen so that *x*% parameters are removed in total. See et al. showed that by pruning 40% of the network, the results only decreased minimally as seen in Figure 2.11. The figure shows the BLEU score of a system trained on the WMT 2014 dataset for the English-German language

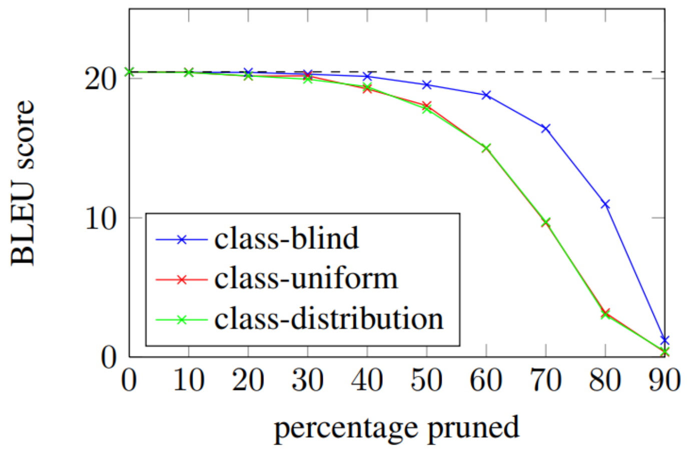


Figure 2.11: BLEU score of an English-German NMT system using different pruning schemes with different sparsity levels (See et al., 2016)

pair. This meant that although the computing resources needed decreased substantially, the system had negligible performance loss.

Ahia et al. (2021) evaluated pruning techniques in data-limited regimes, as well as taking into consideration computing resources constraints. They coined the term 'low-resource double bind', which means that they are working in a data-limited regime as well as working with compute resource constraints. They explained how their pruning process involved training a dense model normally, progressively removing weights that are deemed unimportant, and finally continuing to train the pruned model to recoup performance. Magnitude pruning (Zhu and Gupta, 2017) was used as it is easy to use and achieves better results than other state of the art approaches (Gale et al., 2019). This works by estimating the weight importance by the actual value of the weight. The closer it is to 0, then the more likely it is that it is unimportant. Finally, this method removes these unimportant weights at scheduled intervals. The authors showed that pruning deep neural networks had a positive impact on frequent sentences, but a negative impact on less frequent ones. They also showed that models can generally tolerate high levels of sparsity and retain their BLEU performance and human-judged translation quality.

2.2.7 Multi-Sense Words

Chang et al. (2021) proposed a novel embedding method for multi-sense text sequences that may be longer than a single word. The idea is that a sequence can be represented by multiple distinct multi-mode codebook embeddings, to best capture the different senses that the sequence may represent. These embeddings are simply cluster centers that are summaries of the distributions of the multi-sense sequences. The idea is that during training, the distance between the words with the same meaning and the predicted cluster centers are minimized whereas during inference the actual cluster centers are predicted. This can be seen in Figure 2.12. The authors originally developed this technique for use in unsupervised sentence similarity and extractive summarization tasks, but the basic idea can be utilized for MT systems to solve the problems of words and phrases having multiple senses.

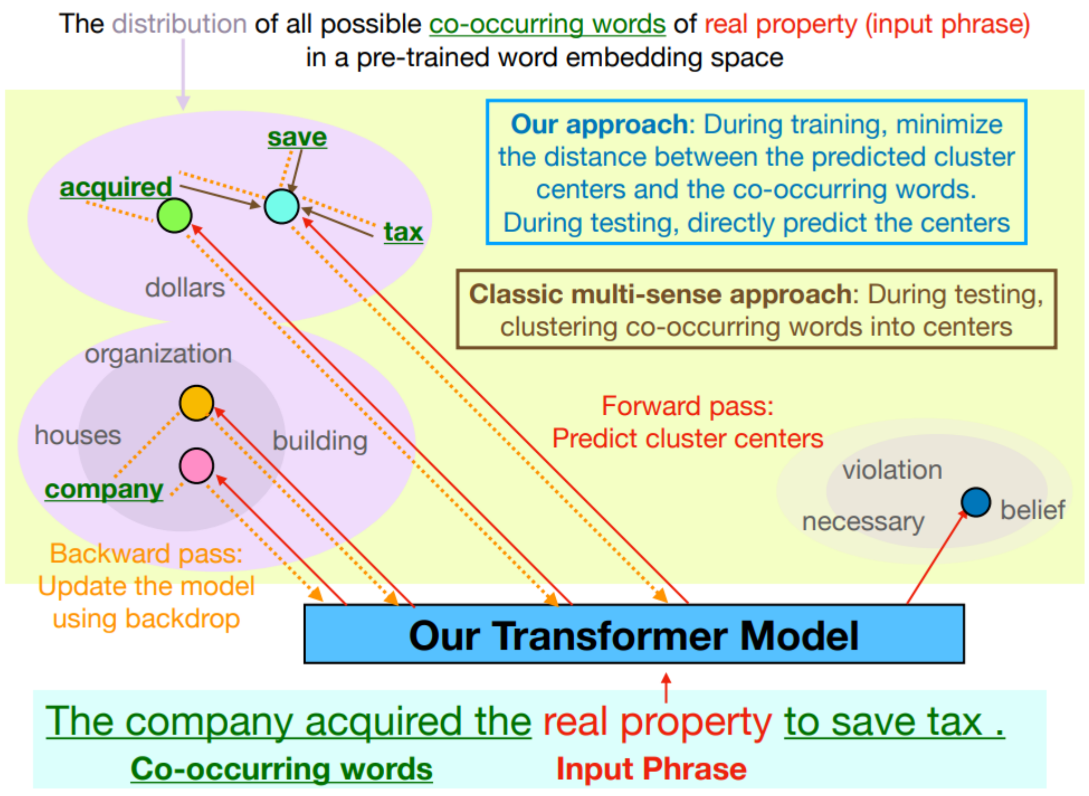

The distribution of all possible co-occurring words of real property (input phrase) in a pre-trained word embedding space
save
acquired
tax
dollars
organization
houses
building
company
Our approach: During training, minimize the distance between the predicted cluster centers and the co-occurring words. During testing, directly predict the centers
Classic multi-sense approach: During testing, clustering co-occurring words into centers
Forward pass: Predict cluster centers
violation
necessary
belief
Backward pass: Update the model using backdrop
Our Transformer Model
The company acquired the real property to save tax .
Co-occurring words
Input Phrase

Figure 2.12: System overview of Chang et al. (2021)

Hangya et al. (2021) aimed to improve the context of words that have multiple senses. The authors claim that such words, known as multi-sense words, usually just have their most frequent sense appear in parallel corpora and thus the system can not fully understand the different meanings of these words. They introduce CMBT (Contextually-mined Back-Translation), which aims to solve this issue by using pre-trained cross-lingual contextual word representations (CCWRs) to get the context in which the word appears in to derive its meaning. To detect multi-sense words, BabelNet synsets (Navigli and Ponzetto, 2012) are used. It is also interesting to note that they have a list of multi-sense words for Maltese too, although as a native speaker, it appears that this list is not very comprehensive. Finally, back-translation is used to train the NMT system, where the original multi-sense words may be masked to ensure that this word is contained in the back-translation. The system overview can be seen in Figure 2.13

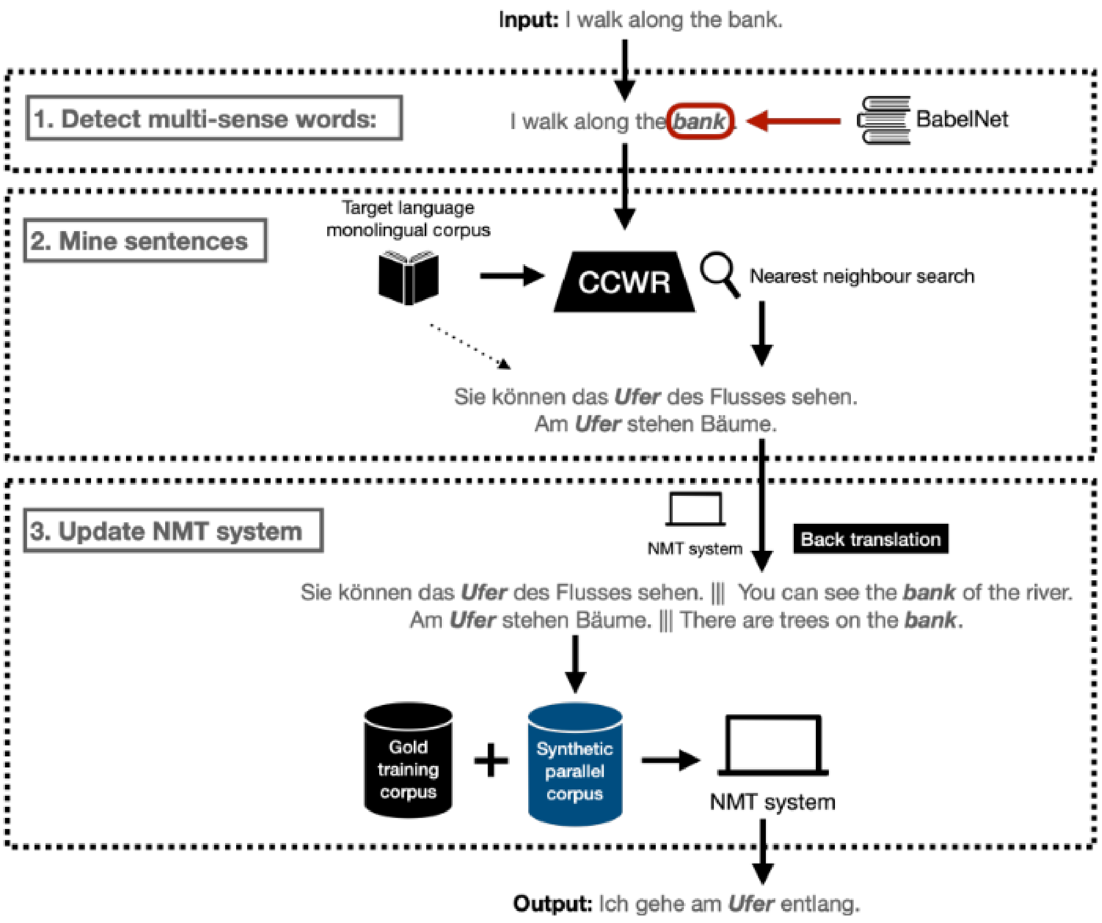

Input: I walk along the bank.
1. Detect multi-sense words:
I walk along the bank.
BabelNet
2. Mine sentences
Target language monolingual corpus
CCWR
Nearest neighbour search
Sie können das Ufer des Flusses sehen.
Am Ufer stehen Bäume.
3. Update NMT system
NMT system
Back translation
Sie können das Ufer des Flusses sehen. ||| You can see the bank of the river.
Am Ufer stehen Bäume. ||| There are trees on the bank.
Gold training corpus
Synthetic parallel corpus
NMT system
Output: Ich gehe am Ufer entlang.

Figure 2.13: System overview of (Hangya et al., 2021)

One of the downsides of this system is that getting a list of homonyms is sometimes not as simple as getting it from Babelnet. As a potential alternative to this system, that can be used for all languages (not just the ones already supported by Babelnet), a clustering algorithm may be used to detect the multi-sense words. Sato and Heffernan (2020) propose this and show a case study for Japanese. This is done by using contextualized embeddings (using BERT (Devlin et al., 2019)) to cluster individual tokens, with the aim being that individual tokens of the same meaning will be mapped in vector space closer together, in one cluster. On the other hand, embeddings mapped far away from each other are likely to be homonyms. Their experiments consisted of using BERT to create the contextualized embeddings, and then used a Gaussian Mixture Model for the clustering due to its simplicity and performance. However, the authors note that there are other algorithms that can be used as well. To halt the system by checking that the number of clusters is likely to be correct, Gap Statistic (Tibshirani et al., 2001) is used.

This technique is a statistical testing method, that takes the output from the clustering algorithm and compares the change of dispersion within each cluster against the null hypothesis. Essentially, it tries different values of the number of clusters to see which performs the best against a random setup created with uniform distribution (null hypothesis). This is a minimization problem, where the algorithm tries to find the minimum change between the result of splitting the data in *k*-clusters and the result of splitting the data in *k+1*-clusters. The clustering method can stop once the standard deviation between the values is smaller than a specific threshold, because noise can play a part beyond this part.

2.2.8 MT Evaluation

The most popular metrics to evaluate an MT system are BLEU and CHRF. BLEU works by using n-grams in both the candidate and reference sequences, regardless of the actual word order. The more matches BLEU can identify, the better the score is. The implementation takes into consideration certain pitfalls, such as the case where the model predicts the same reference word multiple times, resulting in a much higher score than one would expect. To overcome this scenario, BLEU will consider a reference word as complete once a match from the predicted side is found, and it won't be possible to match the same reference word again, resulting in a low score as one would expect.

Character N-gram F-score (CHRF) also presents a score based on n-gram precisions, however this uses the F-score of character n-grams rather than word n-grams. Variations of this metric exist, such as CHRF2, where instead of using the F1-Score, the F2-Score is used instead. In general, CHRF is calculated as follows:

$$\text{CHRF}\beta = (1+\beta^2)\frac{CHRP \cdot CHRR}{\beta^2 \cdot (CHRP + CHRR)}$$

Where CHRP stands for the percentage of character n-grams in the hypothesis which have a corresponding character in the reference (precision) and CHRR stands for the percentage of character n-grams in the reference, which also have a corresponding character in the hypothesis (recall). The value of β that is used in WMT shared tasks is CHRF2, so that will be the metric used to evaluate our work.

Nowadays other neural-based metrics exist, where the quality of the translation is estimated, such as COMET (Rei et al., 2020) and Prism Thompson and Post (2020). Unfortunately, they are based on pre-trained models that have been trained on XLM-R, which has not yet been tested on Maltese therefore it's accuracy on Maltese is unknown.

Evaluation is an issue that is discussed by Kocmi et al. (2021). The authors evaluate different automatic evaluation techniques, including the most popular metric to evaluateMT systems, namely BLEU. They argue that the best metric to use is a pre-trained neural metric such as COMET or Prism Thompson and Post (2020), and a string-based metric such as CHRF should be used as the secondary metric. The results of the correlation between human judgement and the metrics can be seen in Figure 2.14. They also state that for the best evaluation, a paired significance test should be done to reduce metric misjudgement. Kocmi et al. also showed that the sole use of BLEU may have hindered development in the area, as papers were optimised to get a high BLEU score, which may not always correlate directly with a high qualityMT system. This is exactly what Freitag et al. (2020) stated too, that string-based metrics such as BLEU may be biased against modelling techniques that generally improve quality when judged by humans. This could mean that past research may have actually discarded high quality MT systems because it did not result in high BLEU scores.

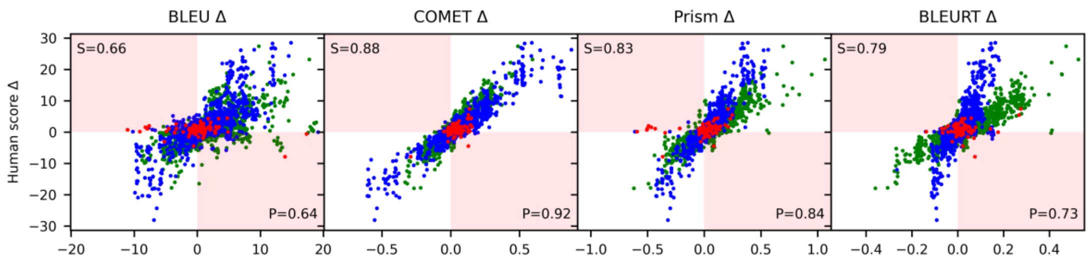


Figure 2.14: Correlation of human judgement against the judgement of the automatic metrics, including BLEU, COMET, Prism and BLEURT. One can see how COMET achieved the best score, followed by Prism. The different colours represent different languages.

Although Kocmi et al. provided a step in the right direction by suggesting COMET and CHRF, they still provide point estimate scores. This results in limited information at segment level, especially considering that the gold-standard references usually contain noise and are not perfect in any way. Thus, these point estimate scores could be unreliable, since the gold-standard reference could be unreliable too. Glushkova et al. (2021) discuss this issue in detail, and propose *uncertainty-aware* MT evaluation, that outputs a range of quality scores rather than a single point estimate. The authors use the COMET framework like Kocmi et al. (2021), and combine it with two uncertainty estimation techniques (Monte Carlo (MC) dropout (Gal and Ghahramani, 2016) and deep ensembles (Lakshminarayanan et al., 2016)), to estimate the trustworthiness of the gold-standard reference. MC dropout obtains a set of quality scores by running a number of forward passes and dropping some units based on a probability. Deep ensembles train a number of separate models in parallel to obtain a set of quality scores. Once the set of quality scores are obtained, a confidence interval is used to represent this set, ranging from the lowest to the highest score. This can be used in a number of scenarios, such as potentially detecting noisy inputs and detecting critical translation mistakes, that is useful in use cases such as when assisting human translators in professional settings.

Both Glushkova et al. (2021) and Kocmi et al. (2021) use evaluation methods that require evaluation based on ground-truth/gold-reference data. However, given that we are working in a data-limited regime, it may be more beneficial if we had to use more data to train the system and diminish the percentage of data used to evaluate. To compensate for this, Quality Estimation (QE) techniques may be utilized to evaluate the system without gold-standard references. These methods are very popular in post-editing, reducing human efforts as the system would be able to tell which parts of the output is likely high quality and which are low quality and would thus need human editing. Tuan et al. (2021) proposed a novel technique that trains an unsupervised QE model based on synthetic data. This results in not having to rely on costly examples provided by real humans. The authors create synthetic data by using a masked language model (Multilingual Bidirectional Encoder Representations from Transformers (mBERT) (Devlin et al., 2019)) and pre-trained NMT models to create translations. Quality annotations are then generated by using the TER tool (Snover et al., 2009) to compare the synthetic sentences with the references.

However, it has to be noted that the technique proposed by Tuan et al. may produce biased noise, and is complex to train since it relies on extra models to create the synthetic sentences. Zheng et al. (2021) proposed an alternative to this, which is a self-supervised technique that is able to estimate the quality both on a sentence level as well as on a word level. The system works by masking some outputs that the machine generated. If these masked outputs can be correctly predicted again based on the context, then the target word is correct. This would allow the model to get a word-level quality estimation, which can then lead to sentence-level quality estimation by summarizing the word-level predictions.

2.3 Maltese Machine Translation

Limited research exists in the context of Maltese machine translation. Rosner and Bajada (2007) worked on an SMT system, specifically targeting the issue that the translations resulted in word-for-word outputs. The authors came up with an algorithm to extract

Subject Field	Terminology	Fidelity	Intelligibility	Wellformedness	Total
Legal	5	4	2	2	13
Financial	4	4	4	4	16
Medical	4	4	4	3	15
Technical	3	1	1	1	6
Literary	2	3	2	1	8
Total	18	16	13	11	58

Table 2.2: Google Translate results scored from a scale of 0 to 5 on each methodology chosen

Subject Field	Terminology	Fidelity	Intelligibility	Wellformedness	Total
Legal	5	4	2	2	13
Financial	2	4	3	2	11
Medical	4	4	4	3	15
Technical	2	2	1	1	6
Literary	1	0	0	1	2
Total	14	14	10	9	47

Table 2.3: Bing Translate results scored from a scale of 0 to 5 on each methodology chosen

phrases as well as evaluation on a set of bilingual texts.

In 2015, Azzopardi (2015) researched the performance of available translation technologies for Maltese. Large-scale evaluation is carried out on various technologies including dictionaries, corpora, software such as spell-checking, translation memory, terminology memory, terminology management and machine translation. The evaluation of Google and Bing translators are of particular interest. The results can be seen in Tables 2.2 and 2.3. Overall, the results show that Google Translate is overall better than Bing Translator by 11 points as rated by human translators. One does have to note that these results are from 2015, so one would expect that the translation systems improved throughout the years.

2.4 Dataset Quality

The quality of the dataset to be used will affect the trained model. Data containing a lot of noise implies that the model will be less accurate. Unfortunately, when working in low-resource languages in particular, the amount of available datasets tends to be very limited and thus one does not have the luxury to pick and choose the best datasets. In fact, the internet provides a large number of available corpora, however these come with little guarantee over their accuracy and legitimacy.

Denkowski et al. (2012) proposed a system that filters noise by utilizing language models and word alignments to score each sentence pair, where pairs that have a higher score are likely to be correct translations of each other. Xu and Koehn (2017) proposed Zipporah, which is a system that selects the best sentences in a dataset. This is done by training a logistical regression model on known, clean data and synthetic noisy data. This way, each parallel sentence is scored and thus an average score of the quality of the dataset can be retrieved. A comparison of Zipporah and the system proposed by Denkowski et al. can be seen in Figure 2.15. Unfortunately, Zipporah requires data that is known to be clean, which in our case can only be achieved by manually cleaning these datasets.

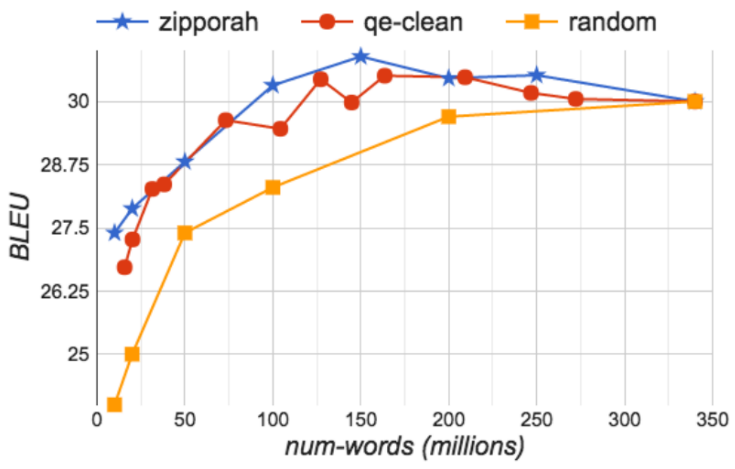


Figure 2.15: Results of Xu and Koehn (2017) on French - English TED talks against Denkowski et al. (2012)

Xu et al. (2019) filter out noise from synthetic data by computing a semantic similarity score using bilingual word embeddings to learn the mappings. This is done by taking the cosine similarities of the source and target sentence vectors after learning the bilingual linear mappings. This work was done on the Korean-English language pair. The data involved to create the embeddings in their case was created by a bilingual speaker who manually translated the top 4500 English words (excluding stop words and function words) into Korean. This would take a considerable amount of time for it to be done in other languages, however it is possible to reuse language models such as mBERTu (Micallef et al., 2022) for our use case to get the semantic similarity scores without having to manually create and learn the embeddings.

Muischnek and Müürisep (2019) examimed the most common issues in parallel and monolingual corpora and how these affect the translation system. Unlike the systems proposed by Xu and Koehn (2017) and Xu et al. (2019), the system proposed by Muischnek et al. is rule based and thus does not require any extra data. Instead, they propose filters for the following issues:

1. Duplicate parallel sentences (such as different corpora crawling the same document)
2. The same sentence having multiple corresponding sentences
3. Sentences with more than half of the characters being non-alphabetical and sentences with more than triple the number of non-alphabetical characters on one side than the other
4. Repeating tokens (usually from some form of MT, either obtained via back-translation or the original source document was automatically translated)
5. Sentences written in a different language

They showed that in some cases, up to 65% of the corpus was removed yet the BLEU score increased by up to 1.6%. Arora et al. (2021) performed a similar study with similar filters as done by Muischnek and Müürisep (2019). However, they also added a number of important filters for different types of noise, such as:

1. Sentences containing long words (these are usually words that were mistyped and a space was forgotten, for example)
2. Very short or very long sentences (such as those having up to 3 words or more than 80 words)
3. Pairs that actually contained multiple sentences, even though they were supposed to be aligned as sentence level. If one side has multiple sentences and the other side does not, it is likely to be a misalignment error when the dataset was being crawled.

Arora et al. (2021) showed that their NMT system's BLEU score also increased when it was trained on a clean corpus over the original corpus, despite the reduction in size. It was shown that the BLEU score increased by 1.39.

2.5 Conclusion

In conclusion, multiple interesting techniques have been researched that are vital for a decent performance of the implementation. It is clear that most MT systems are now using neural techniques rather than statistical ones. The attention mechanism is nowadays a staple of the transformer architecture, which achieved great results Bahdanau et al. (2014). Pre-trained language models can be used as part of our implementations due to the linguistic information that they contain over the language. Tokenizers are necessary to feed our raw data to our implementation, and there are multiple different techniques for tokenization as well. It appears that in our case, a subwords tokenizer such as BPE Sennrich et al. (2015b) would work best, as discussed in Section 2.2.3. Our datasets contain a lot of monolingual data as well, and thus the technique detailed in Sennrich et al. (2015a) can be used to utilize this data. Robustness is another important technique that is given priority due to the fact that not all the datasets collected are manually translated and verified, so it is expected that some input noise is in the dataset, and thus a Self-Correcting Encoding framework (Wang et al., 2021) is promising. Pruning is another interesting technique that may allow the implemented solutions to be stored and used completely offline, making them faster to use and lighter to store, without sacrificing too much performance. Multi-sense words may be improved as shown in Hangya et al. (2021), where the authors state that most multi sense words are underrepresented and thus, only the most frequent sense of that word is returned in most cases. Section 2.2.8 shows the issues in MT evaluation, and how careful one must be when evaluating using only BLEU, as is often the case. This section also delved into the local context in Section 2.3, where the literature available for Maltese MT is presented. Finally, Arora et al. (2021) studied the most common types of noise in datasets and came up with a technique to filter out noise by assigning each parallel sentence a score. The following section goes over the implemented solutions that are based on these techniques.

Chapter 3

Implementation

As stated in Section 1.1, this project implements various techniques and compares the results to a baseline model, with the aim of seeing which individual techniques are promising to assist future researchers in this area. This is because, as seen in Section 4, techniques impact different datasets differently. Thus, since we will explore a number of techniques, this section is split in a number of subsections, with a subsection dedicated to each model.

In the previous section, the latest research was presented. Therefore, following this research, it is aimed that a number of implementations are developed on the two language pairs in scope, namely Maltese - English and Icelandic - English. Following the information found in Section 2.2.6, where See et al. (2016) found minimal performance impact by pruning 40% of the neural network, the Pruning Model is created, following the techniques used by Ahia et al. (2021). Then, in Section 3.5, robustness was discussed where Wang et al. (2021) aim to eliminate the negative effects of noise on the NMT models and as such, the Robustness Model is therefore created. Finally, in Section 3.6, Hangya et al. (2021) aim to improve the translations of multi-sense words and this technique is modified and adjusted to the needs of the language pairs in scope, and therefore the Multi-Sense Words Improvement Model is created.

Apart from the aforementioned papers, other techniques listed in the previous section are also taken into account. These include the Attention mechanism in Section 2.2.1 that is included in all the models and is a standard part in NMT models nowadays, as well as the subwords tokenizer, which as seen in Section 2.2.3, appears to be the ideal tokenizer to use in this case. The monolingual data is also added to the systems following the results of the systematic study published by Burlot and Yvon (2019), as seen in Section 2.2.4.

As can be seen in Figure 3.1, Models 1 and 2 are baseline models that are built using different backend libraries, with Model 1 using Fairseq and Model 2 using Tensorflow. Models 3 and 5 build on Model 2 since they all use Tensorflow, whereas Model 4 is built on Model 1 as it uses Fairseq.

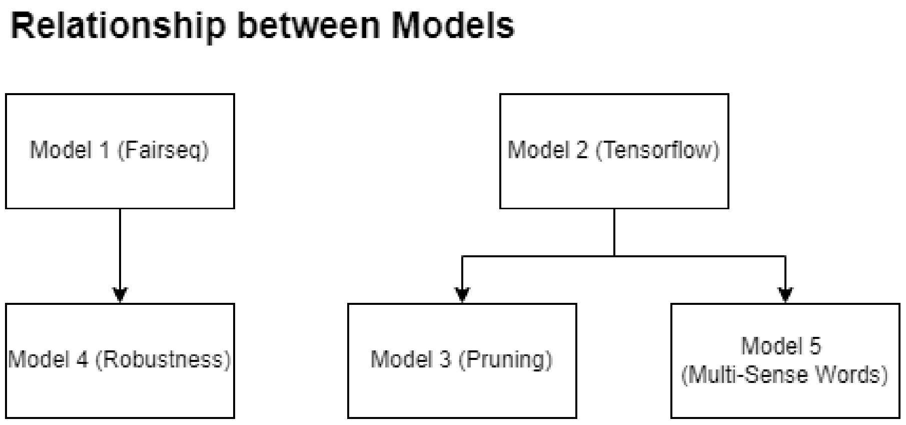


Figure 3.1: Schematic diagram of the models, showing which models are built on top of each other.

Section 3.1 lists the datasets used as well as the necessary preprocessing performed. Sections 3.2, 3.3, 3.4, 3.5 and 3.6 include detailed descriptions of each model, including the core features of each model. Section 3.7 details how the monolingual data was utilized, and Section 3.8 details how each individual dataset was scored, for further information on the datasets collected.

3.1 Dataset

In total, this research will make use of two large datasets, one for each language pair. Each dataset is compiled from various online sources. The sources for the Maltese - English language pair are the following:

3.1.1 English - Maltese

- Arab-Acquis by Habash et al. (2017) contains Maltese - English translations, translated by professional translators.
- Bilingual corpus from the European Vaccination Portal[1], which is a bilingual corpus retrieved from `https://vaccination-info.eu`

[1]`https://bit.ly/3dLbGX9`

- Bilingual corpus from the Publications Office of the EU on the medical domain [2], which is a bilingual corpus retrieved from the Publications Office of the EU on the medical domain.
- Bilingual corpus made out of PDF documents from the European Medicines Agency, (EMEA) [3], which is a bilingual corpus.
- COVID-19 ANTIBIOTIC dataset [4], which is a bilingual corpus retrieved from `https://antibiotic.ecdc.europa.eu/`.
- COVID-19 EC-EUROPA dataset [5], which is a bilingual corpus retrieved from `https://ec.europa.eu/*coronavirus-response`.
- COVID-19 EU presscorner V2 dataset [6], which is a bilingual corpus acquired from `https://ec.europa.eu/commission/presscorner/`.
- COVID-19 EUROPARL v2 dataset [7], which is a bilingual corpus retrieved from `https://www.europarl.europa.eu/`.
- Digital Corpus of the European Parliament(Hajlaoui et al., 2014), which is a multilingual corpus covering a wide range of domains from the European Parliament's official website.
- DGT-Acquis(Steinberger et al., 2014), which is a multilingual corpus extracted from the Official Journal of the European Union.
- ELRC [8], the European Language Resource Coordination, who published a bilingual dataset from multilingual sites. Preprocessing such as normalization, cleaning and deduplication was performed on this dataset.
- English - Maltese Parallel corpus from Tatoeba project [9], where Tatoeba`https://tatoeba.org/` is a collaborative website, holding a collection of sentences and translations.
- OPUS (Tiedemann, 2012b), which contains a large parallel dataset for Maltese - English that has not been manually reviewed.

[2] `https://bit.ly/3R2G5OH`
[3] `https://bit.ly/3QWIjPM`
[4] `https://bit.ly/3pBCg7u`
[5] `https://bit.ly/3AcjIzR`
[6] `https://bit.ly/3wmCyTD`
[7] `https://bit.ly/3wl3brZ`
[8] `https://www.lr-coordination.eu/node/2`
[9] `https://bit.ly/3cejoIU`

- EUIPO - Trade mark Guidelines [10], taken from `https://euipo.europa.eu/` is a multilingual small corpus manually translated in 22 EU languages by professional translators.
- Malta Government Gazette [11], processed by the European Language Resource Coordination.
- Laws of Malta [12], processed by the European Language Resource Coordination.
- MaCoCu (Bañón et al., 2022), which produced a large corpus of monolingual data for Maltese.

3.1.1.1 Number of Sentence Pairs per Dataset

The datasets above are listed below with further information for each dataset, including their size in GBs and the number of sentence pairs:

[10] `https://bit.ly/3AB01Tr`
[11] `https://bit.ly/3QDXm1a`
[12] `https://bit.ly/3ceekE7`

Detailed Maltese - English Dataset Information	
Dataset Name	Number of Sentence Pairs
Arab-Acquis	20,539
European Vaccination Portal	472
Publications Office of the EU on medical domain	3,093
EMEA	410,809
COVID-19 ANTIBIOTIC	391
COVID-19 EC-EUROPA	554
COVID-19 EU Press Corner	1,708
COVID-19 EUROPARL	141
Digital Corpus of the European Parliament	3,509,151
DGT-Acquis	3,146,610
ELRC	26,622
Tatoeba	382
EUIPO	17,330
Malta Government Gazette	5,391
Laws of Malta	9,185
MaCoCu (monolingual)	25,512,299
Total (parallel)	7,152,378
Total (monolingual)	25,512,299
Total	**32,664,677**

3.1.2 English - Icelandic

The datasets used for the Icelandic - English pair are the same ones used in the WMT 2021 shared task (Akhbardeh et al., 2021), with the exception of ParIce as they released an updated version with 1,769,000 new parallel sentences since the shared task was created. The sources are the following:

- Paracrawl (Bañón et al., 2020), which contains a parallel dataset automatically crawled from various online sources.
- ParIce (Barkarson and Steingrímsson, 2019), which is a parallel corpus consisting of various subcorpora, such as religious documents, EEA documents, etc...
- Newscrawl (monolingual) & WikiTitles (Akhbardeh et al., 2021), provided by WMT for the WMT 2021 shared task.

- The Icelandic Gigaword Corpus (Steingrímsson et al., 2018) which is a large monolingual corpus.

Detailed Icelandic - English Dataset Information	
Dataset Name	Number of Sentence Pairs
Paracrawl	2,392,422
ParIce	5,329,651
WikiTitles	50,183
Newscrawl (monolingual)	24,631,545
Icelandic Gigaword Corpus (monolingual)	8,547,672
Total (parallel)	7,772,256
Total (monolingual)	33,179,217
Total	**40,951,473**

3.1.3 Preprocessing

Most of these files are TMX files that needed preprocessing. However some files were also tab delimited or formatted in a similar way. A sample sentence pair in a TMX file can be seen below:

```
1  <tu tuid="1">
2      <prop type="lengthRatio">1.2</prop>
3      <tuv xml:lang="en">
4          <seg>See also:</seg>
5      </tuv>
6  
7      <tuv xml:lang="mt">
8          <seg>Ara wkoll:</seg>
9      </tuv>
10 </tu>
```

As one can see, each sentence pair in TMX format contains an ID (line 1), potentially some additional properties such as the lengthRatio (that are irrelevant for our purposes and can be removed), the language of the individual sentence, and finally the sentence itself.

Our system accepts two separate files with data from each language pair. The data was split in two separate files according to the language, as well as another file containing monolingual data, that will be processed later on. Each TMX file is processed with one sentence pair at a time. Certain pairs had empty corresponding translations for some lines, so these were added to the monolingual dataset instead. The rest of the lines are added to the parallel data files, keeping the alignment between the files in the language pair.

Preprocessing was also performed on files that had a tab delimitation, where each file was read line by line, with each line being split by the delimiter and stored in the respective file according to the language.

Apart from this, the final dataset was checked for any duplicates, in which case they were removed. Duplicates are considered to be the exact same sentence pair. If there are multiple entries for the same source text, but the target text is different, then they are kept, since there may be multiple equally correct translations of the same source sentence.

The monolingual dataset is then backtranslated using a baseline model discussed in Section 3.2 to achieve further parallel sentences synthetically.

Both datasets are shuffled and split into a training set, validation set and test set. The training set is used to train the models on representative data, the validation set is used to periodically test the model's performance after each epoch, to get an idea of whether the model is overfitting or underfitting. It is also used to determine whether the model actually improved between an epoch and another, so we can save the model with the best performance on the validation set. The test set is then used as part of the evaluation stage to determine the model's final performance. The dataset is split with 70% used for the training set, 10% used for the validation set and the final 20% used for the test set.

3.2 Model 1: Fairseq Baseline Implementation

The first model is a baseline implementation, meant to see the effectiveness of a quickly developed automatic translation system, trained on our datasets. It is built and trained using Fairseq (Ott et al., 2019). Fairseq is a library that allows for easy implementation of a machine translation system through Command Line Interface (CLI) commands, meaning minimal code is needed to create a fully working machine translation system. The architecture used is built on Mager et al. (2021), that is publicly available[13]. This however was further modified to achieve slightly better performance, mainly in relation to its performance. The flow of the model can be seen in Figure 3.2

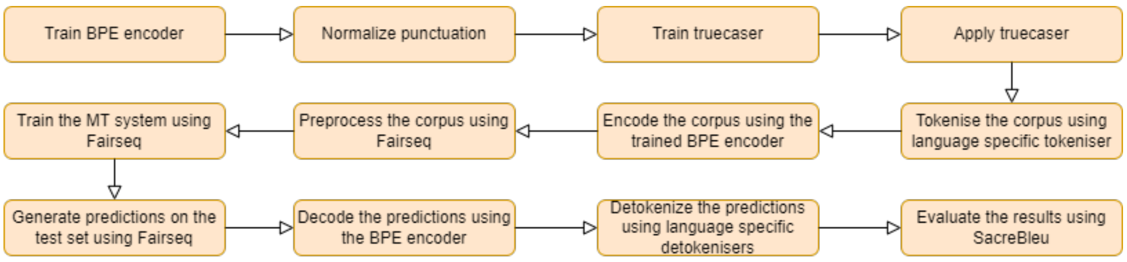

Train BPE encoder
Normalize punctuation
Train truecaser
Apply truecaser
Tokenise the corpus using language specific tokeniser
Encode the corpus using the trained BPE encoder
Preprocess the corpus using Fairseq
Train the MT system using Fairseq
Generate predictions on the test set using Fairseq
Decode the predictions using the BPE encoder
Detokenize the predictions using language specific detokenisers
Evaluate the results using SacreBleu

Figure 3.2: Overview of the flow for Model 1

Firstly, a BPE tokenizer is trained on the training set only. The ideal vocab size after trial and error turned out to be in the region of 32,000. This is important to note as having it too high or too low will cause issues as detailed in Section 2.2.3. After training this tokenizer, the model and the vocabulary list are stored. The model will be used to encode and decode the inputs and outputs of the machine translation system, whereas the vocabulary list will be used in the preprocessing stage.

Next, the MosesDecoder[14] package is used to preprocess the dataset. Firstly, the package is used to normalize punctuation in all training, validation and test sets, where inconsistencies are removed and punctuation is normalized, such as by removing extra spaces and removing punctuation. Following this, a truecaser is trained. Truecasing is the process used to learn the automatically casing of certain words, and intuitively learns the names of proper nouns and words that should be capitalized, which helps reduce data sparsity issues. Only the training set is used to train the truecaser, and once this is trained, the files in the validation and test set are truecased along with the training set.

In the case of Maltese data, a tokenizer specifically designed for Maltese was used as it was seen that the regular English tokenizer does not tokenize everything correctly. For

[13]https://github.com/AmericasNLP/americasnlp2021
[14]https://www.statmt.org/moses/

this, the tokenizer from MLRS [15] was used, which utilizes regular expressions to tokenize linguistic expressions that are specific to Maltese, such as certain prefixes and articles.

Once the preprocessing with MosesDecoder is done, the dataset is encoding using the previously trained BPE encoder. Following this, the vocabularies are built and the training data is binarized using Fairseq, via the fairseq-preprocess command. It was determined that words appearing only once should be converted to an 'unknown' token, since if a word appears only once there is a high likelihood that the word is a mistake, or at worst, a very uncommon word.

The architecture used is a transformer architecture with five encoder layers and five decoder layers with 512 dimensions each. There are two attention heads for both the encoders and decoders, with 2048 dimensions for each. The dropout probability is 0.4, whereas the attention and ReLu dropout are set to 0.2 each. Finally, the learning rate is set to $1e^{-3}$, but the initial learning rate is actually smaller, at $1e^{-7}$ and increases using a learning rate scheduler to linearly increase the rate after 4000 steps. Once the learning rate reaches $1e^{-3}$, the rate is then decayed by the inverse square route of the update number. The model is trained for 10 epochs in total. These hyperparameters were based off of the original Transformer paper (Vaswani et al., 2017), yet some were slightly modified after some improvements were noted on our datasets. Specifically, we used five encoder and decoder layers to make our system a bit lighter and faster, and a dropout of 0.4 is used instead of 0.1.

After the model is trained, the next step is to generate the translations of the test set and get it ready to be used for the evaluation stage. The Fairseq command fairseq-generate is used to generate the translations. The beam size is set to 5, so that the 5 best translations are chosen according to their respective probabilities. A length penalty is also set to 1.2, since longer sentences are slightly more preferred compared to shorter ones. Since the penalty is greater than 1, slightly longer outputs will be returned. Once the translations are produced, they are decoded using the previously trained BPE encoder.

[15] `https://mlrs.research.um.edu.mt/`

3.3 Model 2: Tensorflow Transformer Baseline Implementation

The second model is another baseline implementation, with the main difference between this model and Model 1 is that this one uses the Tensorflow (Abadi et al., 2015) library, which makes it very simple to modify for the following iterations due to its low level functionalities, unlike Model 1 where things certain features and optimizations are abstracted by Fairseq (Ott et al., 2019). Like Model 1, this model also uses the transformer architecture. Indeed, the implementation is a transformer model that uses self-attention to handle variable sized inputs. The overall architecture can be seen in Figure 3.3.

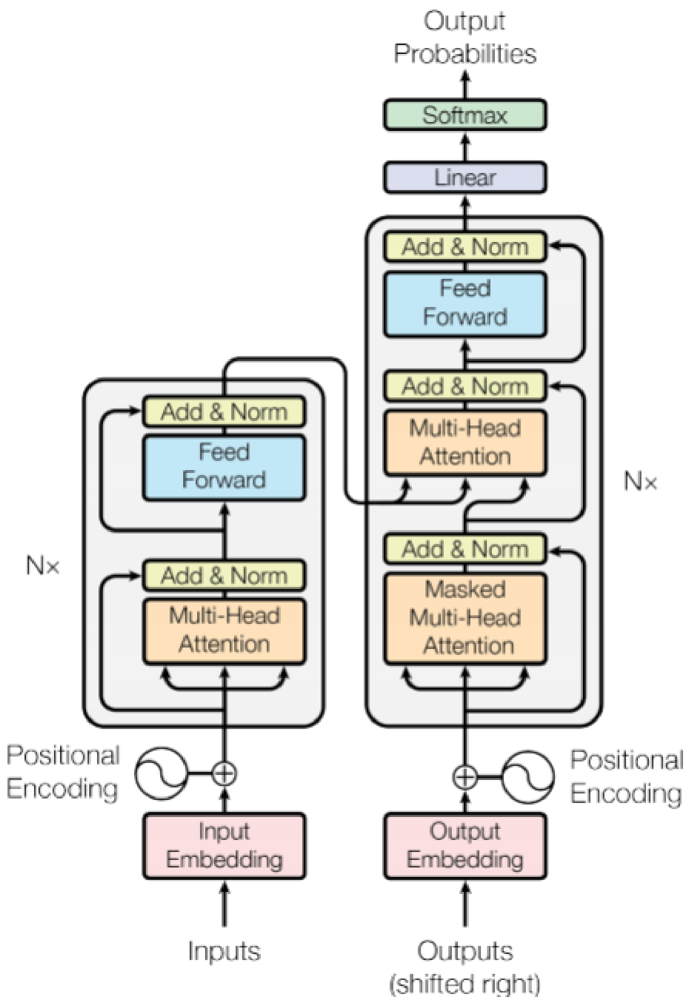


Figure 3.3: Overview of the Transformer architecture (Vaswani et al., 2017)

Firstly, the dataset was converted to a data structure known as a *Tensorflow Dataset*.

This data structure stores the raw data in a structured way, by allowing the data to be split between training, validation and testing set, as well as structuring the data into the source language and target language, for easy accessing within the code. This data structure is also particular useful for this model since it is created by the Tensorflow library, to be easily used in Tensorflow code.

Following this, a subwords tokenizer was trained to be used in the tokenization and detokenization processes. The tokenizer is built on the Wordpiece algorithm, which is the algorithm used in the original BERT (Devlin et al., 2019) paper. In this algorithm, initially all the characters in the corpus and placed in a vocabulary. A language model is then internally trained on the training data using this vocabulary. Following this, new word units are added by combining two word units from all the word units. The new one that is generated is the one that, when added to the model, improves the probability of the training data the most. This process is repeated until the vocabulary limit is reached or the probability stops increasing by a certain amount.

This model does not feature any recurrent or convolutional layers and instead uses a stack of self-attention layers. These layers do not recognize any sequential order in their inputs as they are a set of vectors. For this reason, a vector called the ‘positional encoding vector’ is added so that the model is still able to note some information regarding the relative positions of each token in the input sequence. This vector is added specifically to the embedding vector. Generally, embeddings represent a token in vector space based on their meaning, so tokens with similar meanings will be closer to each other. Note that in this case, a token is usually a subword rather than a complete word. Thus, they are mapped based on the context that the subwords appear in. Once this positional encoding vector is added, the tokens will be mapped based on their meaning and positioning, so tokens that that are similar and closer to each other in the original input sequence will be mapped closer to each other.

The architecture starts from an encoder. This input is put through an embedding which is then summed with the positional encoding. The resultant output is then passed through a number of encoder layers that generates an output for each input token in the sequence.

Each encoder layer has a number of sublayers. The first sublayer is the multi-head attention layer. This layer consists of three linear (dense) layers, that are then passed as inputs (Q (query), K (key), V (value)) to the Scaled Dot Product Attention function. This function outputs a representation of the attention weights multiplied with the V vector. This results in the most important tokens to remain as they are as well as to flush out the irrelevant ones. The outputs of each head are then concatenated together and passed through a dense layer.

The tokenizer pads some inputs to fit a certain width. A padding mask is necessary so that the model does not treat the padding as input. Essentially, this is a separate array of the same length of the input, where if an item in the array is '1', then this means that the corresponding item in the original input is padding and should be masked. On the other hand, if the item in the masking array is '0' then it means that the corresponding item in original input is part of the input sequence. Apart from this there is also a lookahead mask so the model takes a casual approach rather than a fully visible approach. In this approach, the system predicts the second token in a sequence by looking at the first token only, and not the ones that come after the second token.

Following the multi-head attention sublayer, within the architecture there is also a point wise feed forward network that consists of two fully connected layers. In this case a ReLu activation is used in between the layers. Both of these sublayers are connected by a residual connection to help combat the vanishing gradient problem. The residual connection is then followed by a layer normalization.

The output of the encoder is then passed to the decoder along with the decoder's own input (self-attention). In the training stage, this means that the actual target is the decoder's input since the **Teacher Forcing** method is used so that even if a mistake is done in the current step, this is rectified in the following step. Naturally during training part of the target is masked, so the decoder can only see the tokens from the beginning of the sequence to the current sequence. This mimics the real life scenario, where the target sequence is not available and the decoder only has it's own predictions up until the current step. In fact, during inference, the decoder's input is simply the *[START]* token.

So overall, the decoder consists of an output embedding from the encoder, positional encoding as well as a number of decoder layers. The input is therefore the summation of the positional encoding with the embedding.

As can be seen in Figure 3.3, each decoder layer consists of a masked multi-head attention layer (that includes both a look ahead and a padding mask), a multi-head attention layer with a padding mask as well as point wise feed forward networks. Each of these sublayers are connected via a residual connection and a layer normalization. The output of the decoder is passed to the final linear layer.

The hyperparameters of the base model are based on Vaswani et al. (2017). These include having six encoder layers and six decoder layers. The outputs are also in the dimension of 512, whereas the inner layer of the point-wise feed forward network is of dimension 2048. The Adam optimizer is used with a custom learning rate scheduler as stated in Vaswani et al. (2017), and the loss metric used is sparse categorical cross entropy. The model was trained for 10 epochs, on 3 A100 GPUs.

During inference, the input sequence is encoded using the trained tokenizer, and this is passed to the encoder as input. The decoder input is then initialized to be the special token '[START]'. Following this, the padding masks and look ahead masks are calculated.The output of the encoder and the decoder's self-attention are passed to the decoder to output the predictions. Once the first token is predicted, it is then passed to be part of the next input for the next token, so this way the predictions are based on the previous ones.

3.4 Model 3: Pruned & Quantized Model

The third model is built on top of Model 2. This model focuses on pruning and quantization of the neural network, which would allow this model to be used in resource-constrained areas. These constrains may include both storage limitations and compute limitations.

The local Tensorflow Dataset and subwords tokenizer from Model 2 are used since the same dataset is used, so the tokenizers can stay the same. The maximum sentence length was set to 120, as on average the sentence length was 37.9. Thus, around three times the average sentence length was taken to eliminate the excessively long sentences. This means we still use over 97% of the dataset.

Pruning is done by using the built-in Tensorflow API *'prune_low_magnitude'* on selected layers. Not all layers were pruned, since the layers used in multi-head attention are generally regarded as vital to the performance, so they are often left unpruned[16]. The rest of the layers were pruned using a polynomial decay schedule is used to prune the model, where initially 10% of the weights are removed and over time 25% of the weights are removed until the model is trained. To do this, a callback is necessary so after each training step the pruning wrappers are updated. The model was trained for 10 epochs on 3 GPUs as well.

Once training is done, the pruning wrappers are removed so that the original model is restored with the sparse weights. This removes variables that are necessary for pruning during training, but not necessary for inference, thus lowering the model size. However, the model still has the same number of weights, but most of them are now zero. Thus, a standard compression algorithm (gzip) is used to compress the model.

Post-training quantization is also done, to further reduce the size of the model and improve Central Processing Unit (CPU) and hardware accelerator latency, without decreasing the model accuracy by much. This is done by converting the model into a TFLite model, and then converting the weights to a smaller precision. In this case, the weights were converted to 8-bit precision, which achieved a 4x reduction in the size of the model.

[16]`https://www.tensorflow.org/model_optimization/guide/pruning/comprehensive_guide#prune_some_layers_sequential_and_functional`

3.5 Model 4: Robustness & Dataset Quality Check

This model focuses on improving the AI's robustness to noise in the dataset, as well as improving the dataset by using neural network MTs and LMs.

The implementation is based on (Wang et al., 2021), where the authors implemented SECOCO. In this model, self-correcting predictors are used to deal with input noise. This is done by generating noisy data, such as adding common words to the sentence, repeating existing words and removing existing words from sentence. The model is then trained to detect and rectify this noise. Wang et al. publish their code[17], which includes a modified version of the Fairseq library (Ott et al., 2019). A copy of this modified version of Fairseq is used to train our model. The flow of the model is similar to that in Model 1, since both models use the Fairseq library. The overall flow can be seen in Figure 3.2, whereas the flow for SECOCO can be seen in Figure 2.9

A BPE tokenizer is trained on the original training set, keeping the same parameters as in Model 1. This results in the vocabulary lists, to be used later in the preprocessing stage, and the tokenizer model, to be used to encode and decode the inputs and outputs of the model.

Once again, the MosesDecoder[18] package was used to normalize punctuation and train and apply a truecaser. Following this, the whole dataset is tokenised using the BPE tokenizer, or the MLRS tokenizer for the Maltese part of the dataset.

The next step is to generate the noisy data. This is done by using the published code from SECOCO to randomly modify the input sentences. Tokens have a 2% chance to be randomly deleted, and a 2% chance to have a common token added. Of those with a common token added, they also had a 70% chance that the token was repeated between one and three more times. From this process, three files are generated that will be used for the translation training part. The first file contains the actual text with the noise added. The second file contains special tags to indicate where the changes in the original text are, and what changes have been made. This includes tags such as *<Delete-[the]>|9*. This indicates that in position 9, the token *the* has been deleted from the original text. The final file contains only these tags and nothing else. All these files are to be passed to the final preprocessing step, which utilizes Fairseq preprocess.

A modified version of the Fairseq preprocess script is used, to build the vocabularies and binarize the training data. The script was slightly modified by the authors of SECOCO so that the binarized data includes the noise generated from the previous step.

[17]`https://github.com/rgwt123/Secoco`
[18]`https://www.statmt.org/moses/`

Finally, the model is trained using the Fairseq train command. The parameters passed are the same parameters used in Model 1. The model is trained for 10 epochs on 3 A100 GPUs.

Fairseq generate was then used to predict the sentences in the test set. Once the predictions were given by the trained model, they were decoded using the BPE tokenizer. For the English to Maltese MT system, the sentences were then decoded using the MT tokenizer. For the Maltese to English MT system, the MosesDecoder tokenizer was used to decode the English sentences.

3.6 Model 5: Multi-Sense Words Improvement

The final model focuses on improving the translations of multi-sense words. It is expected that more representations of less common meanings of multi-sense words should overall improve their translations during inference. For this model, we utilize the parallel data, without using the backtranslated monolingual data as done in previous models. This model still utilizes this monolingual data, but it is backtranslated in a different manner. Following Hangya et al. (2021), we utilize monolingual data and partly backtranslate it using a machine translation model. We then use a language model to translate any multi-sense words correctly. Utilizing cheap monolingual data is vital to ensure that we have appropriate representations of the different sense of the multi-sense words present in the corpus.

The model is split into three parts. The first part is aimed at detecting the multi-sense words in the corpus. The second part is aimed at crawling the monolingual corpus and storing the embeddings and saving them to a file. The final part is to create sentence pairs for each multi-sense word and create the data to be added to the training set.

3.6.1 Creating a list of multi-sense words

Hangya et al. (2021) used Babelnet synsets to get a list of multi-sense words. Although this does officially have Maltese synsets, it was quickly apparent that there is not a lot of data for the Maltese language. An alternative method was therefore implemented to get the list of multi-sense words, that can be seen in Figure 3.4.

The monolingual datasets of both Maltese and Icelandic were crawled, so that the embedding of each word in its sentence is retrieved. The Maltese embeddings are retrieved using the pre-trained mBERTU tokenizer (Micallef et al., 2022), which is built upon the popular multilingual BERT model, mBERT (Devlin et al., 2019) and extended to have more Maltese tokens in the input data. The Icelandic tokenizer uses the original mBERT

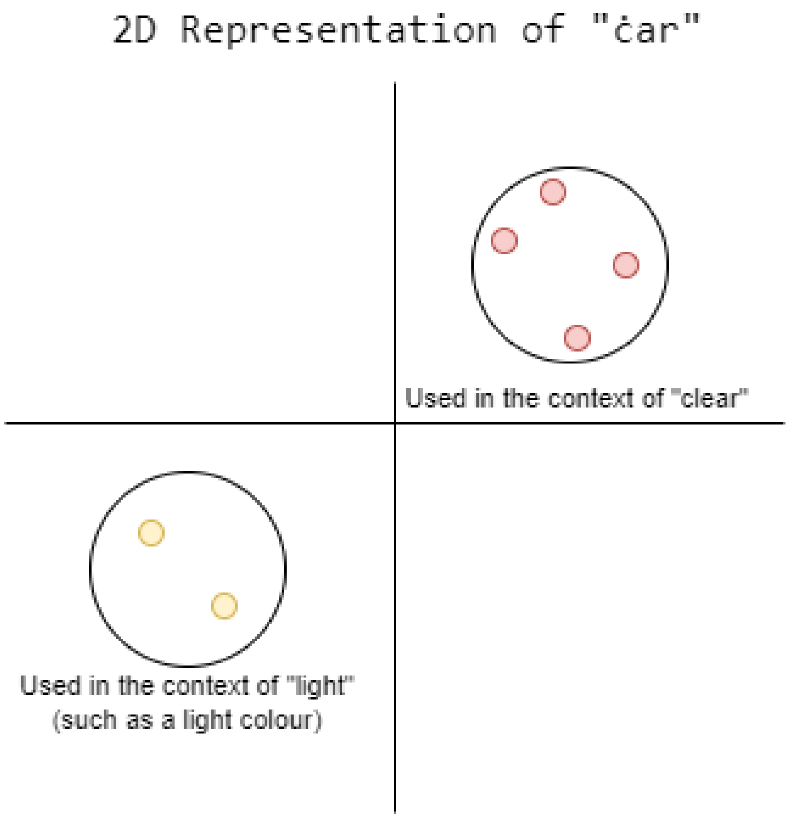

2D Representation of "ċar"
Used in the context of "clear"
Used in the context of "light"
(such as a light colour)

Figure 3.4: Simplified diagram showing how multi-sense words are detected. The words are mapped in vector space (larger than two dimensions) based on their context and clustering is performed to see if a word has multiple clusters.

implementation (Devlin et al., 2019). BERT embeddings are mapped based on the context of the word in the sentence, therefore it is expected that words with the same meaning should appear in relatively similar contexts. The embeddings are retrieved by tokenising the input sequence and removing the special tokens (including “[CLS]", “[SEP]", and “[PAD]", indicating classification, seperators and padding respectively) as well as removing symbols (such as ‘-’). A dictionary was kept with the list of embeddings and the corresponding word. Of course, most words appeared in this dictionary multiple times with slightly different embeddings. Once each word in the monolingual dataset was in this dictionary, this dictionary is looped to see which words have vastly different embeddings in different sentences. For this, the clustering algorithm Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is used to cluster the list of embeddings of the same word in the different sentences that it appeared in. DBSCAN is used over more tradition clustering algorithms such as K-Nearest Neighbours due to the fact that these algorithms expect the value of ‘K’ as a hyperparameter. In this case, we actually want to find the value of ‘K’, because ‘K’ in this scenario represents the number of different meanings of the word. If DBSCAN detects than more than 1 cluster is present, then this implies that the same word may be used to mean different things, depending on the context. DBSCAN requires two parameters, namely the number of minimum samples for the clusters to be considered independent clusters, as well as the maximum distance between two points for them to be considered a cluster. These words are then stored in a text file, indicating that these words are indeed multi-sense words.

3.6.2 Mining multi-sense words

The second step is to traverse each sentence of our monolingual corpus that have a multi-sense word. Every word in these sentences are mapped to vector space using the respective language model and tokenizer for each language. This results in having the embedding vector of each multi-sense word in the corpus based on its context. The reasoning behind this is to replace these multi-sense words in their original sentences with a special token, ‘<MASK>’, so as to not backtranslate the multi-sense word using a baseline NMT system (since it is more likely to translate it to the most common sense of the word, especially if the actual sense is a rare sense). Thus, instead of this, the closest embedding from the language models will be used instead.

To get the closest embedding from the target side (English), a target-side corpus is necessary that will be crawled so that all the embeddings of each word in this corpus are retrieved too. For this corpus, WikiDump (Foundation), a dataset containing articles from Wikipedia was used. It is a comprehensive dataset that is over 20GB in size that includes a large number of sentences in various contexts, which is important when mapping between the source languages to English.

Following this, cosine similarity is used to calculate the closest distance between the source-side multi-sense word and each word in the target-side corpus. This results in the most likely translation of the multi-sense word according to its context. Figure 3.5 illustrates this process.

3.6.3 Backtranslation & Addition of the new data into the corpus

The final step is to translate the sentences in the monolingual corpus using a baseline NMT system, except for the multi-sense words, which have been translated in Step 2.

To translate the sentences using a baseline model without translating the multi-sense words themselves, these words have been replaced by ‘<MASK>’ in Step 2. The idea is

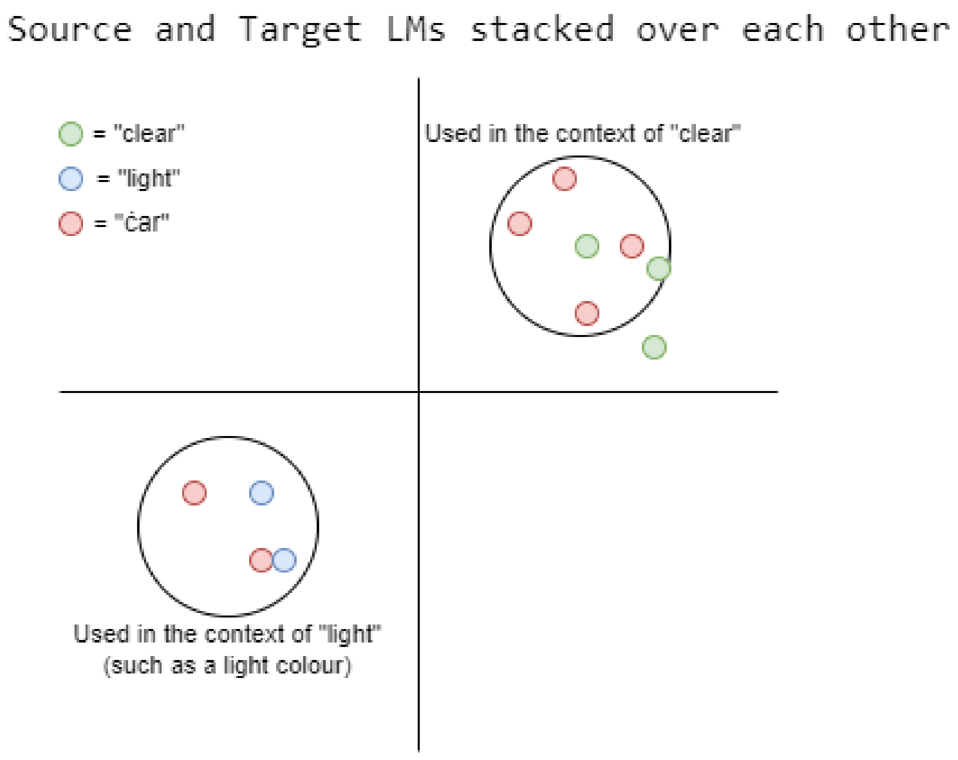

Source and Target LMs stacked over each other
= "clear"
= "light"
= "ċar"
Used in the context of "clear"
Used in the context of "light"
(such as a light colour)

Figure 3.5: Simplified diagram showing how the translations of the multi-sense words are retrieved.

to translate the whole sentence using an NMT as done in normal backtranslation, except for the multi-sense words, which have already been translated in Step 2. However, the baseline NMT system has to be trained so that it does not translate or ignore this special token. To do this, the original parallel dataset was modified so that 1% of the words are changed to '<MASK>'. The corresponding word in the target sentence of the parallel data is also changed to the same '<MASK>' token. To figure out which words correspond to the masked word, FastAlign (Dyer et al., 2013) is used, which is an unsupervised word alignment package. This package takes two sentences in different languages as inputs, and returns an output in the *i*-*j* Pharaoh Format, where the word in position *i* of the source sentence corresponds to the word in position *j* of the target sentence.

The baseline NMT model is trained on this data that contains words masked at random. The architecture used is exactly the same as the one used in Model 2.

Following this, we translate each sentence in the monolingual corpus. Those sentences that have masked words in the resulting translations are replaced with the closest embeddings from mBERTU and mBert for Maltese - English and Icelandic - English translations respectively. Therefore, we now have the corresponding target-side sentences for each sentence in the monolingual corpus. These are consequently added to the parallel dataset.

3.7 Addition of Monolingual Data

As can be seen in Section 2.2.4, backtranslating our monolingual data using an NMT baseline seems to bring the best results in most cases.

Therefore, for our models (apart from Model 5, since it uses a unique technique to backtranslate the monolingual data), backtranslation occurs. This is done by first training an NMT system from the low-resource languages (Maltese and Icelandic) to English. Model 1 is first trained on the available parallel data. Then, the monolingual data is passed to this trained model to be inferred. For this, a beam size of 3 is once again used to get the English predictions. Figure 3.6 illustrates the flow of backtranslation (Dwiastuti, 2019).

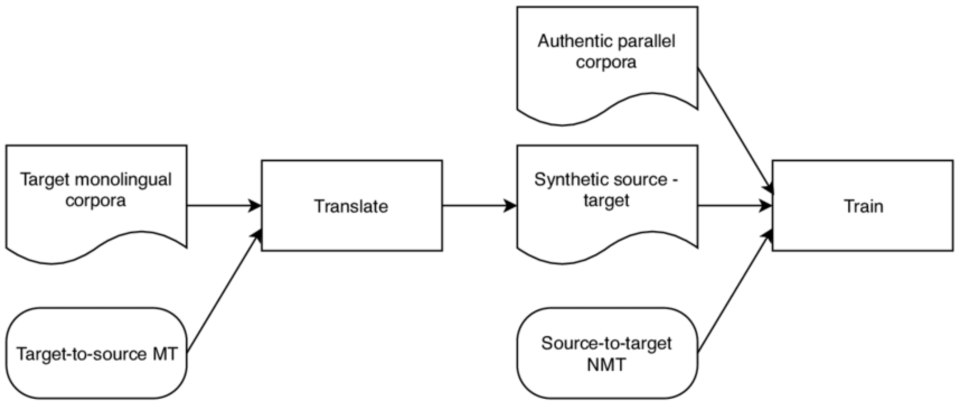

Authentic parallel corpora
Target monolingual corpora
Translate
Synthetic source - target
Train
Target-to-source MT
Source-to-target NMT

Figure 3.6: Figure by Dwiastuti (2019) showcasing how backtranslation works, by training a target-to-source MT system and translating the target monolinugal corpora. The authentic parallel corpora and the predictions of the target monolingual corpora are then used train a source-to-traget NMT.

Once the predictions have been complete, they are added to the parallel data and all the models (except Model 5) are trained on this data.

3.8 Dataset Scoring

Since our datasets are collated from publicly available datasets, it is expected that some parallel sentences are of lower quality than others. For the purposes of this project, it is interesting to evaluate a random sample of each individual dataset and get an idea of the quality of the dataset samples. This will show whether a particular dataset is better or worse than others.

A script was created based on Herold et al. (2022), where each sentence pair in the dataset is given a score by an MT system and a pre-trained language model. The language models used are BERTu for Maltese, IsRoBERTa for Icelandic and the original BERT model for English.

This script begins by looping the whole dataset and splitting it into small batches to give scores in batches rather than one by one. The first step is to use the library *langid.py* to detect the language of the sentences. If either sentence is not in the expected language, then the sentence pair is discarded.

The second step includes getting a list of the language model scores for each pair in the batch. Each pair in the batch is tokenized and passed to the tokenizer and subsequently passed to the language model that is trained on the respective language. The loss of the language model is retrieved and a softmax is applied, which results in the probability of each token appearing in the order it appeared in. Therefore, an automatically translated sentence that is unnatural should get a low score by the language model. The log probability is taken and stored for each token in each sentence in the current batch.

The third step is to get a list of the confidence scores of the MT model for each pair in each batch. In this step, two baseline MT models (that translate from the source language to the target language and vice versa) trained from Model 2 are used. The tokenized sentences are passed to the MT models. The two MT models were slightly optimized for this task, as they do not need to perform any beam search because only the probabilities of the actual known target sequence are important. Thus, since this model predicts the next token along with the probability of each token in the vocabulary, if the corresponding target sentence has x tokens, the model will need to be called to infer x times. As was done in the previous step, the loss of the MT model at each step is retrieved and softmax is applied, which results in the probability of each token. The log probability is then taken and stored for each token in each sentence in the current batch.

Following this, the scores of the language models and machine translation models are normalized (they are divided by the length of tokens) and averaged. The score will always be negative, but a higher score implies that the sentence pair is a good translation.

As shown in Section 5.3, this work can be further extended for future research by creating a language model and optimizing it for the task to not require expensive beam search and thus making it faster. As it currently is, it would take a considerable amount of time and resources to score the whole dataset, including the synthetic data in this manner, due to the fact that the language model is not optimized for this task.

Chapter 4

Results & Discussion

This section will go over the metrics used to evaluate our systems, as well as the individual results of each model.

4.1 Metrics Used

Traditionally in literature, there are two main metrics used to evaluate MT models. These include BLEU & CHRF. Both BLEU & CHRF require a reference translation, where the similarity between the predicted sequence and the reference sequence is measured. BLEU works by using n-grams in both the candidate and reference sequences, regardless of the actual word order. The more matches BLEU can identify, the better the score is. CHRF also presents a score based on n-gram precisions, however this uses the F-score of character n-grams rather than word n-grams. In our case, we will be using bigrams (CHRF-2) as it is the one commonly used to evaluate WMT scores as well.

Certain models are slower to train than others and thus require more GPUs to train within a shorter amount of time, so the resources required for each model will be presented as well in the following sections.

The following sections will present the results for each model. Section 4.7 also includes sample results of each individual dataset for Maltese as described in Section 3.8. For the purposes of this project, they have been added to Model 1, where the results of Model 1 with and without the modification of data will be compared.

Model 1 Results		
Model	**BLEU**	**CHRF2**
English -> Maltese	33.62	66.01
Maltese -> English	61.12	76.11
English -> Icelandic	37.63	60.21
Icelandic -> English	39.22	60.79

Table 4.1: BLEU and CHRF2 scores of Model 1 for each direction

4.2 Model 1: Fairseq Baseline Model

To train this model, each direction was trained for 10 epochs using 3 A100 GPUs. Despite using multiple GPUs, Fairseq was able to automatically handle distributed training. To generate the predictions, the fairseq command *fairseq-generate* was used. This takes in the tokenized and encoded test set and evaluates it automatically. It also allows the facilitation of beam search which can give better results, since it is not fair to assume that the most probable token at each iteration is always the correct one. Therefore, beam search is used to take the top 5 most probable tokens rather than the most probable one for the next iteration. Following some experiments, it was seen that the model achieved slightly better performance, at least on our dataset, if the length penalty is set to 1.2 to get slightly longer sequences than if the length penalty is set to the default (1 - where no penalty is applied).

This model achieved relatively good results in all four directions as can be seen in Table 4.1. For the English - Icelandic pair, it was shown that the BLEU score for the MT system going from English to Icelandic is 39.2, whereas the BLEU score for the MT system going from Icelandic to English is 37.6. On the otherhand, the CHRF2 score for the MT system going from English to Icelandic is 60.2 and the CHRF2 score for the MT system going from Icelandic to English is 60.8.

For the English - Maltese language pair, the BLEU score for the English -> Maltese MT system is 61.1, and 33.6 for the Maltese -> English MT system. This is interesting as there does seem to be a discrepancy between different directions on the same language pair, which seems to be prevalent in the other models as well. The CHRF2 score for the English -> Maltese MT system is 60.2 and 76.1 for the Maltese -> English MT system. The discrepancy between going from English to Maltese and vice versa is surprising, as although some difference in the scores are expected, the difference is usually much smaller. Upon closer inspection, it appears that when predicting Maltese text, the detokenization process is usually far from expected. An example that was generated by the model is as follows: "iċ- ċirku ta' barra tal- munita jiddeplora t- 12 -il stilla tal- bandiera

Ewropea.". The reference sentence is: "Iċ-ċirku estern tal-munita juri t-tnax-il (12) stilla tal-bandiera Ewropea.". Overall, the translation is very good. The main differences are 'ta' barra' instead of 'estern' (which are interchangeable in this case) and the omission of the word 'tnax' (which means twelve), yet '12' was written instead of this.

Examples of Maltese Inputs	
Input	**Prediction**
mhux anqas minn 60 % tal-piż	not less than 60 % of weight
iż-żwieġtal-Prinċep Albert u Charlene	Prince Albert' s marriage and Charlene
Ammont kumplessiv ta' għajnuna ad hoc konċessa lill-impriża: EUR 0,09 (f'miljuni)	overall amount of ad hoc aid granted to the undertaking : EUR 0,09 (in millions)
Notifika minn qabel ta' konċentrazzjoni (KażCOMP/M.6172 – Daimler/Rolls-Royce/Tognum/Bergen)	prior notification of a concentration (Case COMP / M . 6172 - Daimler / Rolls- Royce / Tognum / Bergen)

Table 4.2: Examples of sentences in Maltese and the corresponding English predictions

As can be seen in Table 4.2, overall the model was very capable of translating some sample sentences in Maltese to English. Some grammatical issues were noted, such as in the second example, where the grammatically correct translation would be "Prince Albert and Charlene's marriage", because the input suggests that the marriage is between Prince Albert and Charlene. The third example shows that the model works well for longer sentences that may not be as straightforward as well. This example contains an input that contains a phrase in Latin, that is nowadays known in all languages and one generally expects the phrase to remain as it is. It also contains an amount of money, where the currency is expected to remain in the predicted output as well as the amount. Indeed, the model predicted the correct translation. The fourth example showcases a number of OOV words and proper names that should be kept as is in the predictions and once again, the model predicted the correct translation.

For the opposite direction, the model performed noticeably worse although the translations were mostly understandable. In the first example, the system was not able to correctly predict the multi-sense word 'lots' in its expected meaning based on the context. In the context of the input sentence, 'lots' can be replaced by 'many', which in Maltese would be translated as 'ħafna'. However it was translated as 'lottijiet', which means 'bunches/-batches', and does not particularly fit in the context. Although the meaning can be derived from the translation, a native speaker can tell that it is not the correct translation. Sec-

Examples of English Inputs	
Input	**Prediction**
It is characterised by a sweet taste with a multiplicity of aromas, lots of flavour and a tender flesh.	huwa kkaratterizzat minn togħma ħelwa b' multipliċità ta' aromas , lottijiet ta' togħma u laħam tal- offerta .
The flesh is orange in colour and keeps especially well.	Il- laħam dgħif huwa oranġjo fil- kulur u jżomm speċjalment tajjeb .
the PGI logo.	il- logo tal- IĠP .
Fruit and cereals, fresh or processed	frott u ċereali , frisk jew ipproċessat
The commitments in the Decision did not include any time framework, deadline or review clause.	l- impenji fid- Deċiżjoni ma inkludew l-ebda qafas ta' żmien , skadenza jew klawsola ta' reviżjoni .

Table 4.3: Examples of sentences in English and the corresponding Maltese predictions

ondly, it also makes a mistake near the end of the sentence, where 'a tender flesh' has been incorrectly predicted to 'laħam tal- offerta', which means 'meat on offer'. This has nothing to do with the original input. In the second example, we can see a case where the model invents a new word that changes the meaning of the sentence, by saying 'Il- laħam dgħif', which means the meat that is lean, which is not mentioned anywhere in the input. There is also a mistake in that the expression 'keeps especially well' is translated literally. The third example contains a mistake where the model tried to predict an acronym 'PGI' into Maltese, which did not happen in Table 4.2. Finally, the fourth and fifth examples are correctly predicted.

4.3 Model 2: Transformer Baseline Model

Model 2 Results		
Model	**BLEU**	**CHRF2**
English -> Maltese	1.90	21.73
Maltese -> English	4.51	29.00
English -> Icelandic	2.29	16.70
Icelandic -> English	5.74	20.40

Table 4.4: BLEU and CHRF2 scores of Model 2 for each direction

To evaluate Model 2, the list of sequences to infer was split into batches and each batch is passed to a special function that handles the inference of each batch, so batch prediction can be done (to speed up the process). The best checkpoint model is loaded.

Firstly, each batch is tokenized using the corresponding tokenizer. If the size is greater than the maximum number of tokens, the input sentence is discarded. The model takes the original tokenized input and what has currently been predicted by the model and returns a list of probabilities for the next token. Beam search is used to generate the predictions once again. Following inspiration from Wu et al. (2016) and Tensor2Tensor[1], a function was created that takes the size of the beam, maximum token length, length penalty and the encoder input (the tokenized source sentence).

The beam search function loops until a termination condition is met. The loop stops if the maximum token length is reached or if the lowest scoring finished sequence has a higher score than the highest scoring sequence that is not yet done (while taking the length penalty into account). The actual loop contains three key steps:

1. Firstly, predictions are done to extend the sequences that are not yet terminated, by predicting each one and getting the next token. The scores of the sequences are retrieved by normalizing the log probabilities.

2. The next step is to calculate the scores of the current sequences and store the best ones that aren't yet finished. Each normalized log probability is divided by the length penalty, which is calculated as $(5 + len(decode)/6)^{-\alpha}$ as done in Wu et al. (2016). The amount of sequences stored will be equal to the beam size.

3. The top finished sequences are also stored, with the amount of sequences stored once again equal to the beam size.

Finally, the sequence with the highest normalized log probability is returned. This sequence is then detokenized and returned back to the user. Due to the addition of beam search, testing the whole test is a bit slow. Hence, to mitigate this issue, the test set was split up in 30 parts and 30 CPUs were used to evaluate each part. While one can evaluate on a GPU, due to the large nature of the test set, it was seen that it would be more efficient to utilize cheap CPUs and perform parallel inference on the different CPUs.

The results of this model can be seen in Table 4.4, where the model performed considerably worse than Model 1 in all directions, showcasing how powerful Fairseq is. It achieved 1.90 BLEU and 21.73 CHRF2 scores for English -> Maltese, and 4.51 BLEU and 29.00 CHRF2 for Maltese -> English translations. On English -> Icelandic translations, it achieved 2.29 BLEU and 16.70 CHRF2, whereas for Icelandic -> English translations the system resulted in a BLEU score of 5.74 and a CHRF2 score of 20.40. Upon investigating the results further, interesting properties were quickly noted. For example, the sentence

[1] `https://github.com/tensorflow/tensor2tensor`

“Decision (Hubert Weber)", was translated to “Decizjoni tal-qorti tal-gustizzja". Firstly, the input sentence is not very easy to translate because it contains a name and a surname along with a single word that can actually be translated (decision). This is concerning and shows that the dataset should be filtered more if there are a lot of sentences similar to this. Secondly, although the only word that can be translated is indeed translated correctly, the model ‘hallucinates’ and starts inventing words. Interestingly, the words invented are all words that you would expect to see in a legal/news text, as the prediction reads “Decision of the court of justice". This could imply that there is a lot of data for the legal domain but not enough data for generic day to day use.

This trend of the model not knowing where to stop seems to be prevalent in other predictions as well. For example, given “Fruit and cereals, fresh or processed", the model predicts “frott u cereali, friski jew ipprocessati u friski". Indeed the model actually translates the input sentence perfectly, however it adds "and fresh" to the end of the sentence instead of stopping.

4.4 Model 3: Pruned & Quantized Model

Model 3 Results		
Model	**BLEU**	**CHRF2**
English -> Maltese	2.36	10.79
Maltese -> English	4.24	11.44
English -> Icelandic	0.43	9.12
Icelandic -> English	0.29	7.13

Table 4.5: BLEU and CHRF2 scores of Model 3 for each direction

To infer the model, the TFLite model is first loaded from memory and the memory to accept the input tensors are allocated, since TFLite requires memory allocation prior to inference to optimize the testing speed. The input sentence is tokenized using the trained subwords tokenizer and padded to the max length of the sequence. This is the first input to the model. The second input to the model is an empty sentence with just the special ‘[START]’ token encoded in the target language’s tokenizer. The prediction is retrieved using the same beam search algorithm used in Model 2.

Inference was done using multithreading and only using CPUs, since this model is aimed to be used on any device, especially mobile devices that may not have a GPU. Therefore, inference is optimized for CPU predictions as well. As at the time of development, TFLite does not officially support Nvidia GPUs[2], thus CPUs were used to infer the

[2]https://github.com/tensorflow/tensorflow/issues/34536

test set.

It is also interesting to note the size reduction of the final model. The original model size is 1.08GB. After removing the pruned weights and converting the model to a TFLite model, the size is reduced to 487MB. This is already a 55% reduction. Following quantization of this pruned model, the final model sits at 216MB, which is just 20% of the original model. Looking at the difference in results between this model and Model 2, it looks like one can experiment a bit more with increasing the pruning percentage a bit further, which will reduce the size of the model further. However, one has to note that this model has a much slower inference time on PCs due to the fact that Nvidia GPUs are not supported yet.

Unfortunately, this model performed quite bad overall, as did Model 2, as seen in Table 4.5. The best scores were only 4.24 BLEU and 11.44 CHRF2, for the Maltese -> English direction. The BLEU score is comparable to the score achieved by Model 1, yet the CHRF2 score is almost halved. Interestingly, the BLEU score actually increases by 0.46 in the English -> Maltese direction although once again the CHRF2 score is actually worse.

Examples of Maltese Inputs	
Input	**Prediction**
Deċiżjoni tal-Kummissjoni ta ’ l-24 ta ’ Settembru 2007 dwar tilqim ta ’ emerġenza tat-tjur tar-razzett fl-Italja kontra l-influwenza avjarja b’patoġeniįtà baxxa (notifikata taħt id-dokument numru C (2007) 4393)	commission decision of 24 september 2007 on emergency european farmcatureure vaccination vaccines in italy against low pathogenic influenza (notified under document c c (2007 2007 plant 3))
(notifikata taħt id-dokument numru C (2007) 4393)	(notified under document c (2007 2007) 4393)
Id-Direttiva 2005/94 / KE tistipula l-miżuri minimi ta ’ kontroll li għandhom jiġu applikati fil-każ ta ’ tifqigħa tal-marda ta ’ l-influwenza avjarja fit-tjur tar-razzett jew tjur oħra mrobbija mill-bniedem.	directive 2005 / 94 / ec lays down minimum control measures to be applied in the case of outbreak of aviation influenza disease in farm poultry or other poultry reared by man.

Table 4.6: Examples of sentences in Maltese and the corresponding English predictions

Some sample predictions can be seen in Table 4.6. One can immediately see that although there is still some sense of legible translations, since the idea of the original sentence is still there, it is not nearly as good as the models built on Fairseq. For a start, it tried to translate certain important numbers and acronyms that meant that sentence may no longer have the expected meaning. This is evident in all examples, where a document number C((2007) 4393) was translated as cc(2007 2007 plant 3) and a directive named 2005/94/KE was translated as 2005/94/ec. Apart from these issues, some other minor phrases were translated incorrectly, such as in the first example, where there was no mention of ‘European’ in the original text, yet this was mentioned in the prediction. Another example is that the system did not correctly predict the rare word ‘avjarja’, which means ‘avian’, presumably because this word is not present enough in the training set.

It is also interesting to note that for the Icelandic models, the model performed extremely poorly. Overall, it appeared to perform ‘safe’ predictions and repeat the most likely predictions, such as a lot of repeating commas and stop words. Two such predictions include “south korea - - -" and “it ’ ’ ’ ’ ’ ’ ’ ’ ’.". In reality, no practical coherence was seen in this model in most cases.

4.5 Model 4: Robust Model

Model 4 Results		
Model	**BLEU**	**CHRF2**
English -> Maltese	42.08	71.51
Maltese -> English	58.94	77.10
English -> Icelandic	38.76	59.82
Icelandic -> English	41.11	59.52

Table 4.7: BLEU and CHRF2 scores of Model 4 for each direction

To evaluate Model 4, a slightly different approach was taken than that in Model 1. Following the approach of Wang et al. (2021), *fairseq-interactive* was used instead of *fairseq-generate*. As seen in Section 4.2, *fairseq-generate* takes the tokenized and encoded test set and generates the original input and the prediction. *Fairseq-interactive* takes in the original input sequence and then internally tokenizes and encodes the input. This is representative of the way users would use this MT system.

Therefore, the whole test set was evaluated using this method, using the same parameters passed to Model 1, including beam search with width 5. Apart from this, the path to the encoder also had to be passed so that the sentences are encoded and decoded internally. The inference predictions are then stored to a file. It is important to note however, that sentences that are too long are automatically skipped by Fairseq, therefore an issue is encountered that sentences that are too long (in terms of tokens once they are detokenized) are simply not predicted and printed to file. Thus, a script was created that quickly loops over the inputs passed to the model (that in turn were predicted successfully) and each input sequence in the original test set that is not part of the inference output file is removed from the test set. This way, the test set is a parallel corpus of the same length as the inference output file. Due to the large number of sentences in both English - Maltese and English - Icelandic test sets, multiprocessing was used to split this script into multiple threads. This is done because the script is an example of an 'Embarrassingly Parallel' problem Herlihy et al. (2020), which is an algorithm that can be split up into a number of parallel tasks, due to the fact that there is often little to no dependency and communication between the processes. This is the case here, because each line in the inference output file can be iterated on independently.

This model performed really well, as seen in Table 4.7, getting the highest BLEU score in all directions except Maltese -> English, where the baseline outperformed it by 2.18 BLEU. It is no surprise that it outperformed the rest of the models built using Tensorflow as well, since it is trained using all the underlying features that Fairseq has to offer, both in training and during inference.

Examples of Noisy Inputs in Maltese	
Input	**Prediction**
Il-paragrafu fit fit-tieni kapitlu jitkellem fuq il-ktieb	the President in the second chapter speaks on the book
Il-paragrafu fit tieni kapitlu jitkellem fuq il-ktieb	the second chapter speaks on the book
Il-paragrafu fit tieni kapitlu kapitlu jitkellem fuq il-ktieb	the second chapter speaks on the book
Il-paragrafu tieni kapitlu jitkellem fuq il-ktieb	the second chapter speaks on the book
Paragrafu fit tieni kapitlu jitkellem fuq il-ktieb	paragraph in the second chapter speaks on the book

Table 4.8: Examples of noise added to a sample sentence in Maltese

The model proved to be very effective when common errors occur. For example, consider the sentence "Il-paragrafu fit tieni kapitlu jitkellem fuq il-ktieb", which means "The paragraph in the second chapter speaks about the book". Different types of noise were added to this sentence and tested as seen in 4.8. Firstly, 'Il-paragrafu' was not translated correctly in either case, except when the article was removed. This could indicate that it is more of a tokenization error, as you would expect that the article of each word gets treated on its own as a subtoken. In the first case it actually got mistranslated as 'the President' rather than 'the paragraph'. In cases where a word was written twice, such as the first and third example, the model correctly translated the word once in the prediction. In cases where a word was omitted, such as the '-' in each article from the second example onwards, the word 'fit' in the fourth example and the whole article in the fifth example, the model also translated everything correctly besides the word 'paragraph', which was interestingly only predicted correctly when the article of the word in Maltese was removed.

Examples of Noisy Inputs in English	
Input	**Prediction**
The second page in the first chapter	it-tieni paġna fl-ewwel kapitolu
The second page iin the first chapter	the second page iin the first chapter
The second in the first chapter	it-tieni fl-ewwel kapitolu
The second page in the the first chapter	it-tieni paġna fl-ewwel kapitolu
The second page in the first first chapter	it-tieni paġna fl-ewwel kapitolu

Table 4.9: Examples of noise added to a sample sentence in English

In the opposite direction, one can see that the model performs equally well at fixing noise. Interestingly, when a different type of noise is added, such as a typo in the second example, the model seems to get confused and does not translate a single word at all. In Table 4.9 one can see that the model deals really well with additional words added as can be seen in the fourth and fifth example.

Overall, this model seemed to perform really well on detecting and fixing noise. This could be expanded to handle other types of noise, such as typos, using spellchecking technology.

4.6 Model 5: Multi-Sense Words Improvement Model

Model 5 Results		
Model	**BLEU**	**CHRF2**
English -> Maltese	1.16	23.32
Maltese -> English	5.04	24.78
English -> Icelandic	3.44	13.03
Icelandic -> English	5.79	14.36

Table 4.10: BLEU and CHRF2 scores of Model 5 for each direction

To evaluate Model 5, the exact same inference script as Model 2 is used, since the main differences between these two models are the changes to the data (specifically in regards to the multi-sense words in the data.) That is, the test set was split into 30 parts, each to be evaluated by one of the 30 CPUs available. This can be evaluated using a GPU, however due to the much larger amount of CPUs than GPUs available for use, this ended up being much faster. Each part had its input batched, so each batch can later be predicted at once. Each batch is passed to the beam search function, to get results with an overall higher probability. Finally, the resulting predictions are detokenized and stored in a file to perform BLEU and CHRF2 tests.

From the results, it appears that the model managed to get a correct list of multi-sense words. On average, it was seen that 83% of the predictions did not contain the '<MASK>' tag. This means that the token was ignored in most cases and thus the multi-sense word could not be translated using the LM. This explains why the results of the model, seen in Table 4.10, are very similar to the results achieved by the baseline model, since most of the input data is actually the same, as only 17% of the sentences that contained multi-sense words were translated using the technique described in Section 3.6.

As seen in Table 4.10, the results of this model resemble a lot the results in Table 4.4, since only 17% of the sentences with multi-sense words were translated using the technique in Section 3.6. The results of the direction of the low-resource languages to English produced the best BLEU scores, with 5.04 and 5.79 BLEU score for Maltese -> English and Icelandic -> English respectively. The Maltese models produced the best CHRF2 scores, with 23.32 and 24.78 for English -> Maltese and Maltese -> English respectively, which is an improvement of over 10 in each direction. Unfortunately, it is difficult to measure the effectiveness of the model at translating multi-sense words due to the fact that the baseline performs poorly anyway. However, from the tests performed in Maltese, it looks like this model is not much better than Model 2 at translating multi-sense words. Instead, it actually seems to ignore these words, presumably because it is not confident enough at translating them. For example, giving the model "twegiba ċara", which means "clear answer", returns "answer", and completely ignores the multi-sense word. This shows how the representations of these multi-sense words need to be improved rather than back-translated as done in the previous models. To get this model to perform better, the baseline model needs to learn to predict the '<MASK>' token, and for this, using Fairseq to train such a model this phenomena might be enough, but we leave this for future research.

4.7 Maltese Dataset Scores

As detailed in Section 3.8, the individual datasets are shuffled and 100 samples from each dataset are randomly selected. The results of these tests can be seen in Table 4.11:

Dataset Name	Average Score
Arab-Acquis	-7.92
European Vaccination Portal	-7.50
Publications Office of the EU on medical domain	-7.68
EMEA	-7.61
COVID-19 ANTIBIOTIC	-7.74
COVID-19 EC-EUROPA	-7.75
COVID-19 EU Press Corner	-7.78
COVID-19 EUROPARL	-7.56
Digital Corpus of the European Parliament	-7.89
DGT-Acquis	-8.61
MaCoCu	-7.79
Tatoeba	-6.92
EUIPO	-7.89
Malta Government Gazette	-7.73
Laws of Malta	-7.78

Table 4.11: Average Scores on 100 Samples for each Individual Dataset

From these results, it appears that the Tatoeba project has the most accurate translations. However, it must be noted that from the samples chosen, almost all of them are really short phrases that are only a couple of words long. MT and LM models appear to give a higher confidence score to short sentences and phrases as the confidence decreases as the sequences get longer.

Following this, the COVID-19 Europarl and EMEA datasets achieved a good ranking as well. DGT-Acquis achieved the lowest score at -8.61.

4.8 WMT 2021 Icelandic - English Translation

WMT 2021 was the first year that included a shared task for Icelandic - English translations and vice versa. While a direct comparison between our results and the results of the shared task can't be made due to the fact that we have two million more parallel sentences due to an updated version of one of the datasets in the task, it is still interesting to learn the techniques used and challenges faced.

The top results from the WMT 2021 shared task for the Icelandic - English language pair using only the dataset specified by WMT were achieved by Zhou et al. (2021), under the name NiuTrans. The authors managed to achieve a BLEU score of 31.2 for the English -> Icelandic MT system and 39.22 for the Icelandic -> English MT system. They also achieved a CHRF2 score of 57.48 for English -> Icelandic and 61.04 for Icelandic -> English. They found that the best architecture that achieved the best results is the ODE

Transformer (Li et al., 2022), that is an extension of the standard Transformer architecture which is an ordinary differential equations-inspired model. The advantage of using an ODE transformer over a regular transformer is that such a model can potentially gain less truncation errors, which overall improved the accuracy of the model. Backtranslation was used to utilize the monolingual data to generate pseudo-parallel sentences. Multiple models were trained on the parallel corpus, so iterative knowledge distillation can be used. This is where the information learnt by the ensemble models is passed down to a single model. This process can be repeated numerous times, but each time the gap between the teacher (ensemble) models and the new single (student) model decreases. NiuTrans decided that two iterations are sufficient for their models. They also use multiple single models and combine their probability distribution of the next token to be predicted, although this does increase training and inference time. Apart from this, they use a method called post-ensemble, which consist generating sentences using the several ensemble combinations. Then, the Levinstein distance is used to select the sentence with the smallest average distance to the other sentences. This can be seen in Figure 4.1.

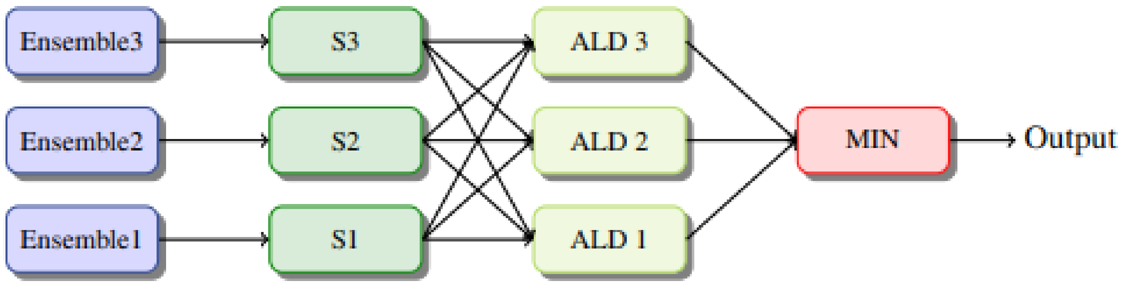


Figure 4.1: Post-ensemble technique Zhou et al. (2021). The ensembles each produce a sentence, and the average Levinstein distance is calculated for each sentence. The smallest one is used as the output.

Facebook-AI Tran et al. (2021) achieved the best results in this shared task, however they used additional data to do so. The additional data used was found online, with CC-Matrix Schwenk et al. (2019), CCAligned El-Kishky et al. (2019) and OPUS Tiedemann (2012a) being cited. Fairseq was used to train the multilingual model. Similar to what is detailed in Section 3.7, the authors also used back-translation to translate their monolingual data. They actually noted that adding back-translation improved the BLEU scores for Icelandic substantially. The authors finetuned their models with mined data that was classified to be most likely within the news domain. They averaged parameters across the last five checkpoints, so to mitigate the bias towards the most recent training data. Noisy channel re-ranking Yee et al. (2019) is also applied, where beam-search is used to give the top *n*-samples. However, instead of using the most likely sample, a more complex equation is used that takes into account the scores of the translation models that translate the source language to the target language and vice versa, as well as the language model trained on the target language. This technique is similar to the one described in Section 3.8, where two LMs and two MTs are used to score sentences in the dataset, except this time they are used to score the best sentences from the beam search. The log probabilities of the two translation models and the target side LM are taken, with a weight being applied on the target LM and the MT system that predicts the source language given the target. This weight is determined via a random search of a 1000 iterations on the validation set. One can easily adapt the code from Section 3.8 to perform a similar technique on our models, as both beam search and sentence scoring are implemented.

The authors achieved a BLEU score of 33.29 and CHRF score of 59.60 for English to Icelandic translations, and a BLEU score of 41.67 and CHRF score of 62.29 for Icelandic to English, which are the highest in this shared task.

4.9 Final Results

The final results can be seen in the Tables 4.12, 4.13, 4.15 and 4.14.

English -> Maltese MT Results		
Model	**BLEU**	**CHRF2**
Fairseq Baseline	33.6	66.0
Transformer Baseline	1.90	21.73
Pruned & Quantized	2.36	10.79
Robustness	**42.08**	**71.51**
Multi-Sense Words Improvement	1.16	23.32

Table 4.12: BLEU and CHRF2 scores of each model on English to Maltese translations.

For English -> Maltese, as shown in Table 4.12, Model 4 performed the best with 42.08 BLEU score and 71.51 CHRF score, followed by Model 1. Surprisingly, pruning performed the third best in terms of BLEU score, beating the Tensorflow Transformer model (Model 2). Despite being about 4 times smaller, it improved Model 2 by 0.46 BLEU yet the CHRF2 score decreased by 10.94. Lastly, Model 5 has the worst BLEU score and the third highest CHRF2 score, at 1.16 and 23.32 respectively.

Maltese -> English MT Results		
Model	**BLEU**	**CHRF2**
Fairseq Baseline	**61.12**	76.11
Transformer Baseline	4.51	29.00
Pruned & Quantized	4.24	11.44
Robustness	58.94	**77.10**
Multi-Sense Words Improvement	5.04	24.78

Table 4.13: BLEU and CHRF2 scores of each model on Maltese to English translations.

The results for Maltese -> English are shown in Table 4.13. Model 1, the Fairseq baseline, achieved a better BLEU score than Model 4, by 2.18, yet a slightly worse CHRF2 score by 0.99. Among the models built with Tensorflow, the multi-sense words improvement model (Model 5) achieved the best BLEU score, whereas the baseline (Model 2) achieved the best CHRF2 score. The pruning model achieved the lowest BLEU and CHRF2 score overall, however the CHRF2 score is once again less than half of the other models built with Tensorflow whereas the BLEU score is only 0.27 less than the baseline model.

English -> Icelandic MT Results		
Model	**BLEU**	**CHRF2**
Fairseq Baseline	37.63	**60.21**
Transformer Baseline	2.29	16.70
Pruned & Quantized	0.43	9.12
Robustness	**38.76**	59.82
Multi-Sense Words Improvement	3.44	13.03
NiuTrans (constrained)	30.60	57.48
Facebook-AI	33.29	59.60

Table 4.14: BLEU and CHRF2 scores of each model on English to Icelandic translations.

The results of the English -> Icelandic models can be seen in Table 4.14. For this model, Robustness and the Fairseq baseline achieved similar results, with Robustness getting 1.13 BLEU score yet 0.39 CHRF2 score less than the baseline. In Table 4.14 one can also see the results of NiuTrans (Zhou et al., 2021) and Facebook-AI (Tran et al., 2021), although the datasets are slightly different. NiuTrans used an older version of Parice, which contained slightly more than 2M less sentences than the most recent version that we used. Facebook-AI noted that they used additional datasets that were not gathered by WMT. Pruning performed the worst in terms of both BLEU and CHRF2, and as seen in Section 4.4, the results were mostly incoherent as well.

Icelandic -> English MT Results		
Model	**BLEU**	**CHRF2**
Fairseq Baseline	39.22	60.79
Transformer Baseline	5.74	20.40
Pruned & Quantized	0.29	7.13
Robustness	41.11	59.52
Multi-Sense Words Improvement	5.79	14.36
NiuTrans (constrained)	39.23	61.04
Facebook-AI	**41.66**	**62.28**

Table 4.15: BLEU and CHRF2 scores of each model on Icelandic to English translations.

Lastly for the Icelandic -> English models, as seen in Table 4.15, once again Robustness achieved the highest BLEU score whereas the Fairseq Baseline achieved the highest CHRF2 score among our models. Facebook-AI achieved the best results overall, having the highest BLEU score at 41.66 and the highest CHRF2 score at 62.28. From the TensorFlow models, the pruning model performed really poorly, with similar scores as in the opposite direction. As discussed in Section 4.6, Model 5 performed similar to the Tensorflow Transformer Baseline, with very similar BLEU scores but a moderate difference in CHRF2 scores, with the baseline model recording around 6 CHRF2 points more than the Multi-Sense Words Improvement model.

Chapter 5

Conclusions

In this study, various techniques were used to research and improve translations for low-resource languages, with a particular focus on Maltese - English and Icelandic - English language pairs. This section includes a revisit of the aims and objectives, suggestions for future work and some concluding remarks.

5.1 Revisiting the Aims and Objectives

Recall the following aims and objectives from Section 1.1:

- **O1** Putting together a dataset from varied sources available to experiment with MT performance when there is more data variety.
- **O2** Creating a baseline MT system built using a standard transformer architecture.
- **O3** Creating different MT systems with different techniques to analyze their performance in comparison with the baseline MT system.
- **O4** Evaluating the system and comparing it with similar systems trained on low-resource languages.
- **O5** Publishing the code and results of this study, as well as the compilation of publicly available datasets to assist further research in this area.

Objective O1 has been completed and detailed in Section 3.1 where the sources of both Maltese and Icelandic data has been compiled. For the Maltese dataset, 7,152,378 parallel sentences were collected and 25,512,299 monolingual sentences were used. For the Icelandic dataset we used the datasets provided by WMT, yet we used a newer version of ParIce that contained over 2M more sentences. This was done with the intention that we get a more comparable number of parallel sentences to our Maltese dataset. For this, we used 7,772,256 parallel sentences and 33,179,217 monolingual sentences.

Objective O2 has been done as part of Models 1 and 2. These two models are both baseline systems, both using a standard transformer architecture. Model 1 is done via a popular MT library Fairseq, which in itself includes a number of optimizations for translation and Model 2 is programmed using a lower level API, Tensorflow. As seen in Section 4, the models built on Fairseq achieved much better results than the models built using Tensorflow, which shows that there are great advantages to using a library that has a lot of optimizations built-in.

Objective O3 includes Models 3, 4 and 5. Three different techniques are developed, with the intention to be built on top of the baseline system. These different techniques include pruning, robustness optimizations as well as improving the translations of multi-sense words. The results showed that pruning can work really well in certain cases, yet can be detrimental to the performance of the system in others. Our results showed that the pruning worked well on the Maltese language, but performed poorly on Icelandic. Robustness performed the best from our models in almost all directions. Multi-Sense Words Improvement did not improve much over the Tensorflow Transformer Baseline model due to the fact that 83% of the multi-sense words in the dataset could not be translated according to the technique described in 3.6.

Objective O4 is completed and detailed in Section 4, where the individual results of each model are detailed as well as a discussion to compare the results. Overall, it was seen that pruning is a promising technique but greatly depends on the dataset used. Models trained on robustness can be a great addition especially when the datasets are not of the highest quality. The model trained to improve multi-sense words is promising but needs improvement to have any substantial performance increase.

Objective O5 is also done and can be accessed through Github[1]

5.2 Limitations

This study may include a number of limitations that should be taken into consideration when considering the results of this research.

The models built on Fairseq have further optimizations built-in to the framework, which explains the discrepancy in results, even between the two baseline models. Fairseq is a framework solely built for text generation tasks, which includes MT, whereas Tensor-

[1]https://github.com/kurtabela/MSc-Thesis

flow is a much lower level API that can be used for a wider variety of tasks since it can be used for any deep learning task. Some of the optimizations built in to Fairseq include:

- Efficient search algorithms built on top of beam search that may directly give better predictions (including various sampling techniques and lexically constrained decodingPost and Vilar (2018), where the models are guided to a better result after being given words that must be present in the prediction)
- Gradient accumulation, where the updates to the model may be done after a number of batches instead of after each batch, to simulate a larger batch size which can improve training speed.
- CPU offloading, so that the CPU handles the parameter and optimizer states, leaving more room in memory for input sequences during training.

Given these features, it is clear how models trained by Fairseq can perform better due to the internal optimizations. Therefore, for our conclusions, the effectiveness of techniques such as pruning and multi-sense words improvement was measured by the increase in performance over the Tensorflow Transformer baseline. One would expect that any improvement (or lack thereof) of these techniques would still be there if they were trained as Fairseq models and compared to the Fairseq baseline model, however this will have to be verified by implementing these techniques as Fairseq models.

The metrics used to present the results are BLEU and CHRF, and as seen in Section 2.2.8, may not show the full context of the results. This, it may be better to

In Section 4.7, 100 random samples were taken. These may not be representative of the whole dataset, and even if a certain dataset is indeed less than ideal, there may still be a number of sentence pairs worth keeping. As discussed in Section 5.3, this can be improved to be done on the whole dataset which leads to a better representation.

Finally, another limitation is that as a native Maltese speaker, qualitative evaluation was done on the Maltese examples, yet no qualitative evaluate could be done on the Icelandic examples due to the fact that we are not native speakers of this language.

5.3 Future Work

This work intends to indicate which techniques have the potential to improve performance on the datasets at hand to allow further research. Since the focus was not on a single model, rather multiple models with different techniques, every model has some room for further research due to the limited time and resource constraints to train multiple models.

The pruning model (Model 3) can potentially be improved by applying 'Quantization Aware Training'. This is the idea that since we are applying quantization post training, the model is trained while modelling the quantization effects at training time. Since the model is aware that it is planned to have quantization applied to it, the results could improve since the accuracy will closely resemble the test accuracy since it will be modelling the weights of the model at inference time.

It would be interesting if the Robustness Model (Model 4) can be modified so that the detection and rectification of noise can be applied at inference time as well. Currently it works like a regular NMT at inference time, with the encoder implicitly trained with noise detection and rectification. However, if this is to be done, it gets easier to evaluate the difference between explicit and implicit noise detection and rectification. It would also improve the model if typos can be detected as well, perhaps with the addition of spell-checking software before each input.

Model 5, the Multi-Sense Words Improvement Model, could be improved in multiple ways. Firstly, for step two (as described in Section 3.6) further work can be done to achieve a better mapping of mBERTU to better align the embeddings in different languages. Although mBERTU is built on mBERT, which means that all embeddings across different languages are still mapped to the same vector space, it is possible that some languages are not mapped exactly on top of each other, in which case mapping the difference will help improve with the detection of the correct translation of the multi-sense words. For step three, to improve the backtranslated sentences, instead of FastAlign, a supervised algorithm such as Sabet et al. (2020) that utilizes contextualized embeddings can be used instead if the data is available, which might make the trained baseline NMT more reliable. Another option to improve the '<MASK>' token from actually appearing in the backtranslated predictions is to perhaps train a multi-lingual model on a higher resource language so this method of returning the token back is learnt.

It would also be interesting to applying the sentence pair scoring technique in Section 4.7 on the whole dataset and set a threshold and remove the pairs below the threshold. For this, it is necessary to optimize the MT and LM inference times, or use a multi-gpu setup so that the sentences are scored in a relatively short amount of time. This can be done by optimizing a language model on the task and removing unnecessary computation, such as beam search since it is not required for this task. Similarly, this technique can also be used to get better predictions as demonstrated by Tran et al. (2021), by applying this technique following the beam search to get the top *n* predictions.

Finally, an important step forward would be to rebuild the techniques that were built using Tensorflow into Fairseq, were massive performance increases are expected and it is also vital to see that any difference in performance (effectiveness of the technique) reflect the results seen when trained as part of a Fairseq model.

5.4 Final Remarks

In total, five different models were developed for the two language pairs, each trained to go from the source language to the target language and vice versa. This results in a total of twenty individual NMT models.

The five models include a baseline model trained with Fairseq, another baseline model trained using a lower level library Tensorflow, a model optimized for size and resource constraints (a pruned model), a model that is trained to be resistant to input noise and a model trained to more accurately detect multi-sense words.

Interesting results are concluded from this study. Firstly, frameworks such as Fairseq contain a lot of underlying optimizations that facilitate MT systems over lower-level APIs such as Tensorflow. Secondly, pruning a neural network can be very promising, but it appears that this is impacted greatly by the dataset. Thirdly, robustness can be very useful, especially for low-resource languages with low-quality datasets. Finally, improving multi-sense words for low-resource languages can be done, using a clustering technique to get the multi-sense words, however more work is needed to backtranslate them properly as only 17% of the multi-sense words were able to be backtranslated using this method.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Ahia, O., Kreutzer, J., and Hooker, S. The low-resource double bind: An empirical study of pruning for low-resource machine translation. *arXiv preprint arXiv:2110.03036*, 2021.

Akhbardeh, F., Arkhangorodsky, A., Biesialska, M., Bojar, O., Chatterjee, R., Chaudhary, V., Costa-jussa, M. R., España-Bonet, C., Fan, A., Federmann, C., et al. Findings of the 2021 conference on machine translation (wmt21). In *Proceedings of the Sixth Conference on Machine Translation*, pages 1–88, 2021.

Arora, K. K., Tomar, G. S., and Agrawal, S. S. Studying the role of data quality on statistical and neural machine translation. In *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, pages 199–204. IEEE, 2021.

Azzopardi, B. S. An evaluation of the computer-aided translation tools and resources available for maltese translators. Master's thesis, University of Malta, 2015.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bañón, M., Chen, P., Haddow, B., Heafield, K., Hoang, H., Esplà-Gomis, M., Forcada, M. L., Kamran, A., Kirefu, F., Koehn, P., Ortiz Rojas, S., Pla Sempere, L., Ramírez-Sánchez, G., Sarrías, E., Strelec, M., Thompson, B., Waites, W., Wiggins, D., and Zaragoza, J. ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.417. URL `https://aclanthology.org/2020.acl-main.417`.

Bañón, M., Esplà-Gomis, M., Forcada, M. L., García-Romero, C., Kuzman, T., Ljubešić, N., van Noord, R., Sempere, L. P., Ramírez-Sánchez, G., Rupnik, P., Suchomel, V., Toral, A., van der Werff, T., and Zaragoza, J. MaCoCu: Massive collection and curation of monolingual and bilingual data: focus on under-resourced languages. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 303–304, Ghent, Belgium, June 2022. European Association for Machine Translation. URL `https://aclanthology.org/2022.eamt-1.41`.

Barkarson, S. and Steingrímsson, S. Compiling and filtering ParIce: An English-Icelandic parallel corpus. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 140–145, Turku, Finland, 2019. Linköping University Electronic Press. URL `https://www.aclweb.org/anthology/W19-6115`.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Mercer, R. L., and Roossin, P. A statistical approach to language translation. In *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*, 1988.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J., Mercer, R. L., and Roossin, P. S. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

Burlot, F. and Yvon, F. Using monolingual data in neural machine translation: a systematic study, 2019. URL https://arxiv.org/abs/1903.11437.

Chang, H.-S., Agrawal, A., and McCallum, A. Extending multi-sense word embedding to phrases and sentences for unsupervised semantic applications. *arXiv preprint arXiv:2103.15330*, 2021.

Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 06 2014a. doi: 10.3115/v1/D14-1179.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014b.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*, 2015.

Correia, G. M., Niculae, V., and Martins, A. F. T. Adaptively sparse transformers. 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)*, pages 2174–2184, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1223. URL https://aclanthology.org/D19-1223.

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

Dai, A. M. and Le, Q. V. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28:3079–3087, 2015.

Denkowski, M., Hanneman, G., and Lavie, A. The cmu-avenue french-english translation system. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 261–266, 2012.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Dwiastuti, M. Indonesian-english neural machine translation. 2019.

Dyer, C., Chahuneau, V., and Smith, N. A. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, 2013.

Ebrahimi, J., Lowd, D., and Dou, D. On adversarial examples for character-level neural machine translation. *arXiv preprint arXiv:1806.09030*, 2018.

El-Kishky, A., Chaudhary, V., Guzmán, F., and Koehn, P. Ccaligned: A massive collection of cross-lingual web-document pairs. *arXiv preprint arXiv:1911.06154*, 2019.

Foundation, W. Wikimedia downloads. URL https://dumps.wikimedia.org.

Freitag, M., Grangier, D., and Caswell, I. BLEU might be guilty but references are not innocent. *CoRR*, abs/2004.06063, 2020.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Glushkova, T., Zerva, C., Rei, R., and Martins, A. F. Uncertainty-aware machine translation evaluation. *arXiv preprint arXiv:2109.06352*, 2021.

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

Habash, N., Zalmout, N., Taji, D., Hoang, H., and Alzate, M. A parallel corpus for evaluating machine translation between arabic and european languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 235–241, 2017.

Hajlaoui, N., Kolovratnik, D., Väyrynen, J., Steinberger, R., and Varga, D. Dcep-digital corpus of the european parliament. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2014.

Hangya, V., Liu, Q., Stojanovski, D., Fraser, A., and Korhonen, A. Improving machine translation of rare and unseen word senses. In *Proceedings of the Sixth Conference on Machine Translation*, pages 614–624, 2021.

He, R., Ravula, A., Kanagal, B., and Ainslie, J. Realformer: Transformer likes residual attention, 2021.

Helcl, J., Libovickỳ, J., and Popel, M. Cuni system for the wmt19 robustness task. *arXiv preprint arXiv:1906.09246*, 2019.

Herlihy, M., Shavit, N., Luchangco, V., and Spear, M. *The art of multiprocessor programming*. Newnes, 2020.

Herold, C., Rosendahl, J., Vanvinckenroye, J., and Ney, H. Detecting various types of noise for neural machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2542–2551, 2022.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Karpukhin, V., Levy, O., Eisenstein, J., and Ghazvininejad, M. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 42–47, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5506. URL `https://aclanthology.org/D19-5506`.

Kocmi, T., Federmann, C., Grundkiewicz, R., Junczys-Dowmunt, M., Matsushita, H., and Menezes, A. To ship or not to ship: An extensive evaluation of automatic metrics for machine translation. *arXiv preprint arXiv:2107.10821*, 2021.

Koehn, P., Och, F. J., and Marcu, D. Statistical phrase-based translation. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 2003.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019. URL `https://arxiv.org/abs/1910.13461`.

Li, B., Du, Q., Zhou, T., Jing, Y., Zhou, S., Zeng, X., Xiao, T., Zhu, J., Liu, X., and Zhang, M. ODE transformer: An ordinary differential equation-inspired model for sequence generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8335–8351, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.571. URL `https://aclanthology.org/2022.acl-long.571`.

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., and Gonzalez, J. E. Train large, then compress: Rethinking model size for efficient training and inference of transformers, 2020.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Mager, M., Oncevay, A., Ebrahimi, A., Ortega, J., Rios, A., Fan, A., Gutierrez-Vasques, X., Chiruzzo, L., Giménez-Lugo, G., Ramos, R., Meza Ruiz, I. V., Coto-Solano, R., Palmer, A., Mager-Hois, E., Chaudhary, V., Neubig, G., Vu, N. T., and Kann, K.

Findings of the AmericasNLP 2021 shared task on open machine translation for indigenous languages of the Americas. In *Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas*, pages 202–217, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.americasnlp-1.23. URL `https://aclanthology.org/2021.americasnlp-1.23`.

Micallef, K., Gatt, A., Tanti, M., van der Plas, L., and Borg, C. Pre-training data quality and quantity for a low-resource language: New corpus and bert models for maltese. *arXiv preprint arXiv:2205.10517*, 2022.

Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL `https://proceedings.neurips.cc/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf`.

Michel, P., Li, X., Neubig, G., and Pino, J. M. On evaluation of adversarial perturbations for sequence-to-sequence models. *arXiv preprint arXiv:1903.06620*, 2019b.

Muischnek, K. and Müürisep, K. Impact of corpora quality on neural machine translation. In *Human Language Technologies–The Baltic Perspective: Proc. of the Eighth International Conference Baltic HLT 2018*, volume 307, page 126. IOS Press, 2019.

Narang, S., Elsen, E., Diamos, G., and Sengupta, S. Exploring sparsity in recurrent neural networks, 2017.

Navigli, R. and Ponzetto, S. P. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial intelligence*, 193:217–250, 2012.

Och, F. J. and Ney, H. Improved statistical alignment models. In *Proceedings of the 38th annual meeting of the association for computational linguistics*, pages 440–447, 2000.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Passban, P., Saladi, P. S., and Liu, Q. Revisiting robust neural machine translation: A transformer case study. *arXiv preprint arXiv:2012.15710*, 2020.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

Popel, M., Tomkova, M., Tomek, J., Kaiser, Ł., Uszkoreit, J., Bojar, O., and Žabokrtskỳ, Z. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature communications*, 11(1):1–15, 2020.

Post, M. and Vilar, D. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1119. URL `https://www.aclweb.org/anthology/N18-1119`.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Ranathunga, S., Lee, E.-S. A., Skenduli, M. P., Shekhar, R., Alam, M., and Kaur, R. Neural machine translation for low-resource languages: A survey. *arXiv preprint arXiv:2106.15115*, 2021.

Rei, R., Stewart, C., Farinha, A. C., and Lavie, A. Comet: A neural framework for mt evaluation, 2020. URL `https:`

//arxiv.org/abs/2009.09025.

Rosner, M. and Bajada, J.-A. Phrase extraction for machine translation. 2007.

Sabet, M. J., Dufter, P., Yvon, F., and Schütze, H. Simalign: High quality word alignments without parallel training data using static and contextualized embeddings. *arXiv preprint arXiv:2004.08728*, 2020.

Sánchez-Cartagena, V. M., Esplà-Gomis, M., Pérez-Ortiz, J. A., and Sánchez-Martínez, F. Rethinking data augmentation for low-resource neural machine translation: A multi-task learning approach. *CoRR*, abs/2109.03645, 2021.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

Sato, Y. and Heffernan, K. Homonym normalisation by word sense clustering: a case in japanese. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3324–3332, 2020.

Schwenk, H., Wenzek, G., Edunov, S., Grave, E., and Joulin, A. Ccmatrix: Mining billions of high-quality parallel sentences on the web. *arXiv preprint arXiv:1911.04944*, 2019.

See, A., Luong, M.-T., and Manning, C. D. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1029. URL https://aclanthology.org/K16-1029.

Sennrich, R., Haddow, B., and Birch, A. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015a.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015b.

Snover, M. G., Madnani, N., Dorr, B., and Schwartz, R. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2):117–127, 2009.

Steinberger, R., Ebrahim, M., Poulis, A., Carrasco-Benitez, M., Schlüter, P., Przybyszewski, M., and Gilbro, S. An overview of the european union's highly multilingual parallel corpora. *Language resources and evaluation*, 48(4):679–707, 2014.

Steingrímsson, S., Helgadóttir, S., Rögnvaldsson, E., Barkarson, S., and Guðnason, J. Risamálheild: A very large Icelandic text corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, 2018.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Taira, B. R., Kreger, V., Orue, A., and Diamond, L. C. A pragmatic assessment of google translate for emergency department instructions. *Journal of General Internal Medicine*, pages 1–5, 2021.

Tay, Y., Bahri, D., Metzler, D., Juan, D.-C., Zhao, Z., and Zheng, C. Synthesizer: Rethinking self-attention in transformer models, 2021.

Thompson, B. and Post, M. Paraphrase generation as zero-shot multilingual translation: Disentangling semantic similarity from lexical and syntactic diversity. In *Proceedings of the Fifth Conference on Machine Translation (Volume 1: Research Papers)*, Online, November 2020. Association for Computational Linguistics.

Thompson, N. C., Greenewald, K. H., Lee, K., and Manso, G. F. The computational limits of deep learning. *CoRR*, abs/2007.05558, 2020.

Tibshirani, R., Walther, G., and Hastie, T. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

Tiedemann, J. Parallel data, tools and interfaces in opus. In *Lrec*, volume 2012, pages 2214–2218. Citeseer, 2012a.

Tiedemann, J. Parallel data, tools and interfaces in opus. In Chair), N. C. C., Choukri, K., Declerck, T., Dogan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012b. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Tran, C., Bhosale, S., Cross, J., Koehn, P., Edunov, S., and Fan, A. Facebook ai wmt21 news translation task submission. *arXiv preprint arXiv:2108.03265*, 2021.

Tuan, Y.-L., El-Kishky, A., Renduchintala, A., Chaudhary, V., Guzmán, F., and Specia, L. Quality estimation without human-labeled data. *arXiv preprint arXiv:2102.04020*, 2021.

Turian, J., Ratinov, L., and Bengio, Y. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394, 2010.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vogel, S., Ney, H., and Tillmann, C. Hmm-based word alignment in statistical translation. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*, 1996.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

Wang, T., Zhao, C., Wang, M., Li, L., Li, H., and Xiong, D. Secoco: Self-correcting encoding for neural machine translation. *arXiv preprint arXiv:2108.12137*, 2021.

Weaver, W. Translation. In *Proceedings of the Conference on Mechanical Translation*, 1952.

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. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

Xiao, T., Li, Y., Zhu, J., Yu, Z., and Liu, T. Sharing attention weights for fast transformer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5292–5298. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/735. URL https://doi.org/10.24963/ijcai.2019/735.

Xu, G., Ko, Y., and Seo, J. Improving neural machine translation by filtering synthetic parallel data. *Entropy*, 21(12), 2019. ISSN 1099-4300. doi: 10.3390/e21121213.

Xu, H. and Koehn, P. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2945–2950, 2017.

Yee, K., Ng, N., Dauphin, Y. N., and Auli, M. Simple and effective noisy channel modeling for neural machine translation. *arXiv preprint arXiv:1908.05731*, 2019.

Zeng, J., Wu, S., Yin, Y., Jiang, Y., and Li, M. Recurrent attention for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3216–3225, 2021.

Zhang, X., Chen, W., Wang, F., Xu, S., and Xu, B. Towards compact and fast neural machine translation using a combined method. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1475–1481, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1154. URL https://aclanthology.org/D17-1154.

Zheng, Y., Tan, Z., Zhang, M., Maimaiti, M., Luan, H., Sun, M., Liu, Q., and Liu, Y. Self-supervised quality estimation for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3322–3334, 2021.

Zhou, S., Zhou, T., Wei, B., Luo, Y., Mu, Y., Zhou, Z., Wang, C., Zhou, X., Lv, C., Jing, Y., Wang, L., Zhang, J., Huang, C., Yan, Z., Hu, C., Li, B., Xiao, T., and Zhu, J. The NiuTrans machine translation systems for WMT21. In *Proceedings of the Sixth Conference on Machine Translation*, pages 265–272, Online, November 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.wmt-1.26`.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.