Doctoral Dissertation

On Language Representation for Low-Resource Neural Machine Translation

Ander Martínez

September 13, 2021

Graduate School of Science and Technology
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Ander Martínez

Thesis Committee:

Professor Taro Watanabe	(Supervisor)
Professor Satoshi Nakamura	(Co-supervisor)
Associate Professor Katsuhito Sudoh	(Co-supervisor)
Associate Professor Hiroyuki Shindo	(Co-supervisor)
Doctor Yuji Matsumoto	(RIKEN AIP)

On Language Representation for Low-Resource Neural Machine Translation*

Ander Martínez

Abstract

In recent years, machine translation systems have taken a qualitative leap. This leap is thanks to the introduction of systems based on neural networks. Neural Machine Translation (NMT) systems have not only yielded unprecedented results, which in certain cases are comparable to those of professional translators, but also come with the promise of being general solutions. However, not all language pairs are the same. If the number of parallel sentences is small, it is considered a *low-resource* pair. The most popular datasets used as benchmarks in research are large parallel corpora of related languages, such as French-English or English-German.

This thesis explores the problems of machine translation with respect to low-resource pairs. We present the key elements of a low-resource machine translation system and the critical decisions that have a great impact on the effectiveness of the system. We propose a novel approach that combines subword-level segmentation with character-level information in the form of character n-gram features to construct subword representations for standard encoder-decoder models. We use a custom algorithm to select a small number of effective binary character n-gram features. We show the benefits and characteristics of the proposed approach through extensive experimentation. The thesis ends with several concrete conclusions drawn from the experiments.

Keywords:

Low Resource, Neural Machine Translation, Word Segmentation, Feature Selection, Monolingual Data

*Doctoral Dissertation, Division of Information Science, Graduate School of Science and Technology, Nara Institute of Science and Technology, September 13, 2021.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

Language is a uniquely human trait that distinguishes us from other hominids and animals. *Ethnologue: languages of the world*, an annual reference publication on living languages, listed 7,139 languages in its 24rd edition [42]. Different societies and human groups use different languages to communicate, based on various reasons including cultural and historical background, perceived formality and/or convenience. Languages are spoken, written or signed.

Translation is the rendering from one language into another [87]. In specialized contexts, the term *translation* is usually restricted to written languages, while the act of translating between spoken or signed languages is referred as *interpretation*.

Translation has been practiced since antiquity with various purposes, such as assisting communication between different cultural groups or to import literature works produced in a different language.

Translation is a specialized job because the translator needs to know both source and target languages well. Moreover, an experienced translator will take into consideration translation theory concepts such as *fidelity* and *transparency*. *Fidelity* or *faithfulness* represents how faithful a translation is to the original, and *transparency* represents how natural the resulting text is, or how likely it is for the readers to realize that they are reading a translation. Often times, a text needs to be re-structured or reformulated in order to sound natural in the target language, and so a trade-off has to be made between *fidelity* and *transparency*. The translation of technical or specialized texts requires that the translator know the topic covered in the text and its specific vocabulary. Furthermore, translation is a very time-consuming job, especially if the volume of text to be translated is

high.

All types of public, specialized or not, may be interested in consuming translations. It may be for these reasons or others that machine translation has aroused interest from the general public. Compared to other natural language processing tasks, such as parsing, even people unfamiliar with the subject can appreciate the usefulness of this technology.

In recent years, machine translation systems have taken a qualitative leap. This leap is thanks to the introduction of systems based on neural networks. Neural Machine Translation (NMT) systems have not only yielded unprecedented results, which in certain cases are comparable to those of professional translators, but also come with the promise of being general solutions. The most recent systems can be trained to translate virtually any language pair if they are provided with a parallel corpus with little or no specific preprocessing. However, not all language pairs are the same. On the one hand, the distance between the languages must be taken into account and, on the other, the volume of the parallel corpus. If the number of parallel sentences is small, it is considered a "low-resource" pair. The most popular datasets used as benchmarks in research are large parallel corpora of related languages, such as French-English or English-German. These datasets have millions of sentence pairs.

In this thesis, we explore the problems of machine translation with respect to low-resource pairs. We present the key elements of a low-resource machine translation system and the critical decisions that have a great impact on the effectiveness of the system.

One of the main problems is identified to be in the representation of texts, and we propose and evaluate a solution that significantly improves translation results.

1.1. Machine Translation

Starting in the 1950s-60s[119], the first machine translation systems used dictionaries and rules to produce translations under limited conditions (for a specific domain and/or with a limited vocabulary). However, they soon realized the complexity of the problem. Human language has ambiguity, metaphors, and puns. The *ALPAC* (Automatic Language Processing Advisory Committee) report[31]

Japanese	**English**
犬は賢いよ。	dogs are smart.
私の本はどこ？	where's my book?
犬を見た。	I saw a dog.
犬が嫌いだ。	I hate dogs.
君の犬はどこ？	where's your dog?

Figure 1.1.: An example of parallel corpus. The data suggests that "犬" can mean "dog" or "dogs".

revealed the complexity of the problem and recommended directing resources to tools to assist human translators. The possibilities of MT systems were also limited by the hardware of that time.

At this time the problem of low-resource languages was already visible. Since the systems required large dictionaries and standards written to measure by experts, most minority languages had fewer resources.

In the 90s, the investigation into *Statistical Machine Translation* (SMT) took hold. One of the milestones of that time was the five IBM models[21]. One of the strengths of statistical machine translation is that a system can be developed with little or no knowledge of the languages involved. These systems build a translation model from parallel corpora. Figure 1.1 shows an example of a small parallel corpus that serves to illustrate how a statistical model could align the words "犬" and "dog".

The quality of statistical machine translation systems depends on the amount of data used to build them. The more data, the better the translation quality (See Figure 1.2). These systems do not only use parallel corpora, but can also use plain text or monolingual corpora to improve translations, as shown by Figure 1.2.

Going from needing dictionaries and experts to needing massive amounts of data, problems persist for low-resource languages. Although monolingual corpora may be readily available in large numbers, parallel corpora are less so.

Statistical translation systems have been refined in research until recent years, being almost entirely replaced by neural machine translation systems.

SMTs consist of various parts like translation models, language models, reordering models, etc. The translation models model the possible translations of each word in the source sentence, while language models model the probability of a sequence of words in the target language, without considering the content of the source sentence. In this way the language models can discard syntactically incorrect or improbable sequences. The systems began to integrate neural models as language models [14, 112, 88].

One can learn more about statistical translation systems by reading Jurafsky and Martin [65] (Chapter 25) or Koehn [71].

1.2. Neural Machine Translation

Neural translation systems can be seen as a continuation of statistical systems. When GPU computing made it practical to include neural language models in SMT systems, the first hybrid systems were born.

Once the language model was replaced by a neural one, the next step was to replace the translation models with neural networks, which allowed both models to be integrated into one, training at the same time and allowing co-adaptation [29].

Neural machine translation (NMT) has made remarkable progress in recent years [66, 123, 8, 131] and has become a popular approach for general machine translation tasks.

The improvement in performance and memory of the GPUs in recent years has allowed to build larger and more robust models. Section 2.4.1 contains a brief explanation of the different architectures and their evolution. Much of the technology and techniques used in NMT come from the SMT era; as an example, metrics such as BLEU, tokenizers, re-ranking etc. Some techniques have been adapted to neural systems. For example Beam Search algorithm for NMT modes is simpler than the one used in SMT systems [73].

Most recent research papers have replaced recurring networks (RNN) with self-attention, since these give better results with longer sequences. Most of the systems that participated in the last WMT challenges [18, 19, 10, 11] are based on Transformer[131]. As neural network performance improves, it is necessary to

revisit techniques and preprocessing inherited from SMT systems. NMT systems can be simplified by integrating some aspects of pre- and post-processing to conform an end-to-end system. As the entire system is based on neural networks, advances in deep learning will also be reflected in NMT systems. In Chapter 2, we discuss these issues.

According to Koehn and Knowles [72], NMT systems may need even more data than SMT systems to achieve equivalent performance. Figure 1.2 shows this trend.

In this dissertation we discuss some techniques that can improve the performance of NMT systems in low-resource situations.

1.3. Low-Resource Neural Machine Translation

The term *low-resource* has been used in the NMT context to refer to language pairs with relatively little data. However, I have not been able to find a clear definition of what an low-resource pair is. From how many sentences or words does a pair stop being low-resource?

Koehn and Knowles [72] include the plot in Figure 1.2 showing how their NMT system performed worse than their SMT system under low-resource conditions. In that specific case, compared to the SMT system, the pair stops being low-resource when it performs better than the SMT system. The NMT system outperformed the SMT at about 15 million target words. Considering an average of 27 words per sentence calculated from *Europarl v7*, that gives us about 0.5M sentences (555,555 sentences). Although in that case the definition is relative to the SMT system, the term low-resource is often used without comparison.

I have researched the use of the term in various publications. As some publications measure the size of the corpus in target sentences and other words in the target language, usually English, we consider an average of 27 words per sentence to calculate the size of the corpus.

Most publications use the term with datasets of less than 400,000 sentences [1, 34, 12, 45]. Some use it with less than 100,000 or 50,000[149, 24], and others with less than 10,000[67, 56]. For cases of less than 10,000 sentences, the term *extremely low-resource* is sometimes used.

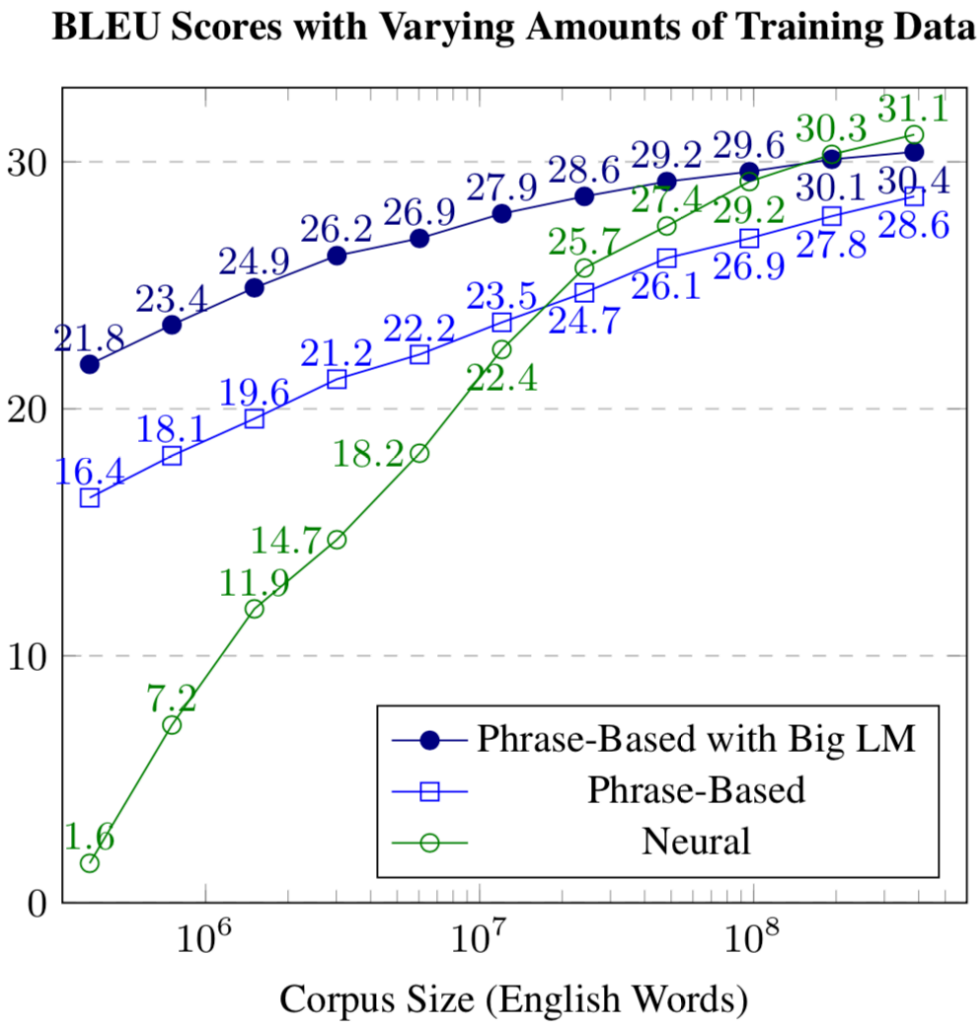

Figure 1.2.: Comparison of NMT and SMT with varying amounts of training data. Figure from Koehn and Knowles [72].

Another commonly used term is *zero-shot translation*, which consists of building MT models using absolutely zero sentences. Monolingual data or data from other languages are used for this. We discuss these models in more depth in Chapter 5.

A distinction must be made between low-resource *pairs* and low-resource *languages*. While low-resource pairs are those that have little parallel data, low-resource languages are usually considered those that have few tools. For example, according to Ethnologue [42], Mandarin Chinese and Spanish are the languages with the most native speakers in the world. However, the Spanish-Chinese pair can be considered low-resource as it has little public data available.

Thus, languages with a lack of tools such as tokenizers, morphological analyzers, parsers, etc., may have the *low-resource* label.

1.4. Vocabulary

NMT systems take a sequence of source language tokens as inputs and predict a sequence of target language tokens as corresponding translation outputs. Initially, MT systems operated with individual punctuation marks and words, delimited by spaces and punctuation marks. However, using this method results in very large vocabularies, particularly for morphologically rich languages. As neural machine translation systems are trained on GPUs and these have limited memory, the vocabularies were limited to the most frequent words in the training data, replacing the infrequent words with the *UNK* symbol, meaning *unknown*. The tokens replaced by these unknown tokens are known as *out-of-vocabulary* (OOV) words or tokens. NMT systems that operate on words are referred to as word-level NMT.

Figure 1.3 shows the English word type count for a corpus of 3M sentences (about 59.9M words). The data comes from English *Europarl v7* [70]. In a vocabulary of about 110K types, most types appear only three times or less in the entire corpus and about 35% of the types are found only once. This distribution responds to Zipf's law. If the vocabulary were limited to 64,000 types, still half of the types would appear 11 times or less. This demonstrates the vocabulary problem in which a large part of the types are under-represented. The problem is bigger with morphosyntactically complex languages.

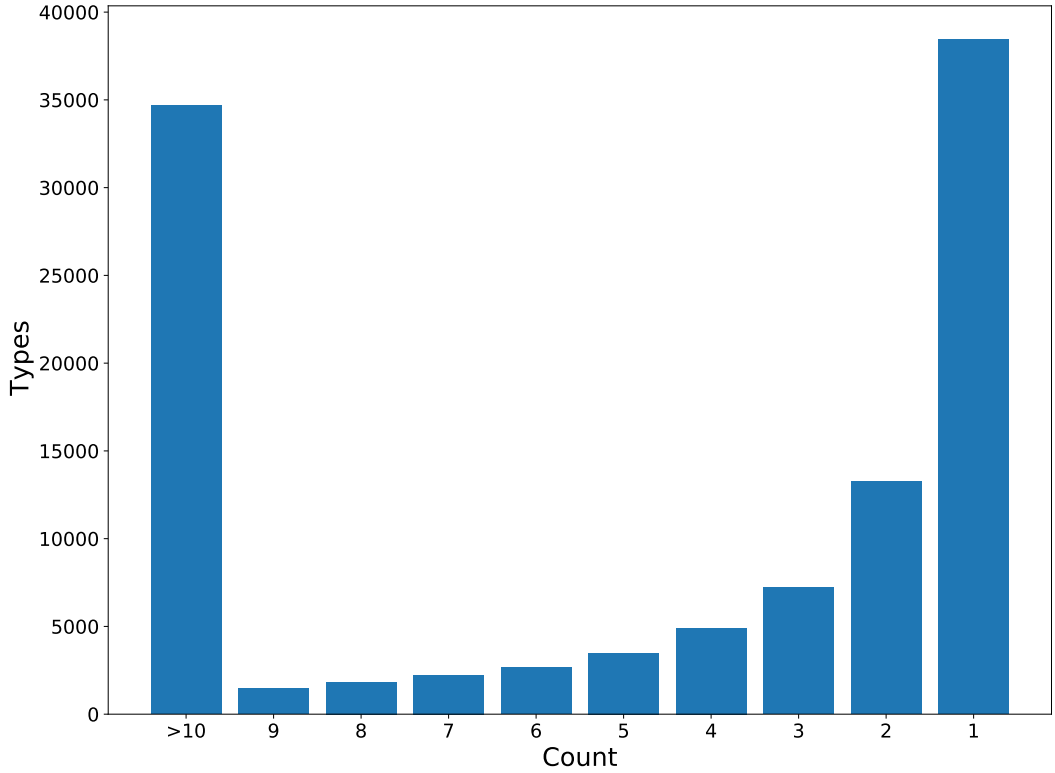

Figure 1.3.: English vocabulary type count. Most types are found only 3 times or less in a 59.9M word corpus. About 35% of the types are found only once.

There has been research on NMT systems that operate on character tokens, usually referred to as character-level NMT systems [30, 80]. The type of token on which the NMT systems operate (character, word...) is called *translation unit* [146]. Although character-level systems solve the problem of having too large vocabularies, they generate other types of problems [80]. The main one of these problems is that they greatly increase the length of the sequences to be processed. If the words of a language have an average of 7 characters, the sequences of 10 words become sequences of 70 characters and the sequences of 50 words become sequences of 350. This increase in sequence length has an effect on training and decoding latency. In addition, the distance of dependencies between words increases and it is more difficult to map the sequences of characters with

a meaning. Character-level systems require larger and deeper architectures to achieve satisfactory results [28].

The granularity of input and output tokens can be smaller than words and larger than character. Such translations units are known as *subwords* [116]. Replacing words with sequences of subwords has become a popular method. Subword segmentation methods, such as byte-pair encoding (BPE), replace infrequent words with more frequent subword segments to limit vocabulary sizes. Limiting vocabulary sizes is particularly important for morphologically complex languages because their vocabularies may contain hundreds of thousands of entries. The optimal size for a vocabulary depends on the particular training data. As indicated by Denkowski and Neubig [38], NMT models trained on smaller corpora tend to perform better with smaller BPE vocabularies, because subwords with very low frequencies are avoided.

However, BPE segmentation does not consider the context of a word or its morphemes. Subwords with similar surface forms are likely to contain the same morphemes, but if these subwords are represented by unique IDs, their similarities are not considered.

Character-level NMT can be regarded as an extreme case of subword segmentation in which each subword corresponds to a single character. Character-level systems can perform better transliteration results than subword-level systems, as shown by Sennrich [115].

Sennrich [115] compared the character-level model proposed by Lee et al. [80] and the BPE-to-character model proposed by Chung et al. [30] to their own BPE-only model [117]. They concluded that character-level decoders "perform worse at modelling morphosyntactic agreement, where information needs to be carried over long distances."

Cherry et al. [28] demonstrated that character-level models with large capacities (large numbers of layers in their encoders and decoders) can outperform BPE when their training datasets are very large (in their experiments, they considered 39.9 million sentence pairs). However, their model suffers from a disproportionate reduction in speed.

In Chapter 4, we explore a novel training method for NMT models that allows us to utilize the character n-gram information of each subword to learn corre-

sponding embeddings. This method is described in detail in Section 4.3. Our proposed models operate at the subword level and make use of short n-gram features within subwords. We refer to the character n-gram features of subwords as *sub-subword* features.

1.5. Parallel and monolingual corpora

We have already discussed the difficulty of collecting enough parallel data to create a robust NMT model. To alleviate the lack of data, one option is to use additional data, either parallel with a third auxiliary language, or monolingual data. In Chapter 5 we discuss these methods.

Some research has shown that models trained with multiple language pairs (multilingual models) can improve the performance of low-resource pairs, particularly when the target language is repeated in other pairs [63, 83, 46, 149].

Another method that works well on low-resource language pairs is *back-translation* (BT) [118, 98, 43]. BT is a simple method that uses an NMT model to translate sentences in a target language into source language sentences, allowing one to synthesize a parallel corpus.

The quality of the data can also vary greatly. Some parallel corpora are created on purpose by translators, or manually compiled and cleaned. Others are the result of aligning sentences of two parallel documents. In this case, the translations may not be exact and may have additional or missing information (under- or over-translation). Some corpora are compiled by web crawlers automatically, and published without manual verification.

The noise problem can be particularly serious in the case of manual corpora. Although some monolingual corpora are cleaned and standardized, others may come from different sources, such as social networks or old books, and may not follow standard or modern spelling, and may contain strange characters or coding errors. Furthermore, often times, these corpora are automatically compiled by web crawlers and filtered using language detection tools. These tools are fallible, causing the corpus to contain text from other languages.

As an example, the Extended Common Crawl Hausa monolingual corpus, published for the WMT 21 News Translation Task, contained large amounts of text

from Japanese song lyrics written in Latin characters. This may be the result of the language detection tool not being robust enough for low-resource languages and also not being prepared for texts written in non-standard spellings, such as Japanese written with Latin characters.

1.6. Other considerations

In this dissertation we do not discuss some methods that might be effective in LR settings. In particular we do not discuss the following ideas:

Using bilingual dictionaries

The most common way to include bilingual dictionaries is to treat them as a parallel corpus. The data is added to the training data along with the parallel sentences. However, there has been research on the optimal way to integrate this data. Some examples of this technology are Arthur et al. [6], Zhang and Zong [141] and Zhong and Chiang [147].

Copy-mechanisms or transliteration-mechanisms

As a general rule, proper nouns are not usually translated. If the source and target languages share a writing system, it may be enough to copy them. If the writing systems are different, it may be possible to transliterate the proper nouns more or less regularly. Depending on the type of text, it can contain a large number of proper nouns or named entities. Although subword segmentation methods like BPE can give good results in copying and transliteration, under low-resource conditions they might not be satisfactory. There has been research on the copy mechanisms. Some examples of this technology are Luong et al. [82], Gu et al. [54], See et al. [113] and Wang et al. [135].

Parsing methods and other linguistic characteristics

Linguistic tools can be used to improve the performance of machine translation systems. Some of these tools can be dependecy parsers or morphological analyzers. The dependency parsing information can be used in various ways like normalizing the word order or including information about the

head word. Also, other information such as part-of-speech tags, dependency tags, or lemmas can be included to *enrich* or regularize the embeddigs.

The problem with this type of method is that although they can improve low-resource translation results, it is precisely low-resource languages that usually lack this type of technology.

Some examples that use dependency parsing are Nguyen Le et al. [94], Eriguchi et al. [44], Currey and Heafield [33] and Deguchi et al. [37].

Some examples using other kind of features are Niehues and Cho [95] and Chakrabarty et al. [24]

Language-specific approaches

In addition to the examples given using dependency parsing and other tools, the knowledge of the languages of the system can in theory help make an educated selection of the hyperparameters and preprocessing. Rules can be written to generate artificial data or augment existing data according to syntactic or derivation rules.

An example of this technology is Torregrosa et al. [129].

1.7. Contributions

In this dissertation we explore different NMT techniques for low-resource systems. In particular, we focus on the method presented in Martinez et al. [84] and through multiple experiments we test how it behaves in combination with other techniques.

Our proposed subword-level NMT training method utilizes sub-subword n-gram information to regularize subword embeddings and the output layer. Sub-subword-level features are selected from all n-grams in the subword vocabulary using a custom feature selection algorithm. The set of selected sub-subword features can unambiguously identify every subword in the vocabulary. Sub-subword binary features are fed through a multilayer feed-forward network to produce subword embeddings. Because subword embeddings are produced exclusively from constant sub-subword features, the subword embeddings can be pre-computed

once the model has been trained. Therefore, our method does not need to modify baseline architectures or the numbers of parameters in trained models.

We explored how our proposed model can be combined with and auxiliary language pair and BT.

We determined the optimal architecture for applying sub-subword features to subword-level NMT and made an analysis of the produced embeddings. We determined that this method works particularly well with small training sets.

Our main contributions can be summarized as follows:

- We describe the problem of low-resource machine translation and summarize the different methods to address it (Chapter 1).
- We make an overview of a standard NMT pipeline and tools and explore their effectiveness under low-resource settings (Chapter 2).
- We propose an NMT model training approach combining subword segmentation to limit vocabulary sizes with n-gram features to construct subword embeddings using a standard multilayer feed-forward network (Subsection 4.3.1).
- We use a feature selection algorithm to select a small number of character n-gram features. Selected features should uniquely identify every word in the vocabulary (Subsection 4.3.2).
- We demonstrate that a standard multilayer feed-forward network works better than a self-attention mechanism (Subsection 4.4.2).
- We demonstrate that constructing subword embeddings strictly from n-gram features works better than using n-gram features to complement standard embeddings (Subsection 4.4.2).
- We demonstrate that our method facilitates large vocabulary sizes with small training datasets, thereby improving translation results (Subsection 4.4.3).
- We show that our proposed method can be combined with BPE dropout to improve performance (Subsection 4.5).

- We make an analysis of the embeddings created by the sub-subword feature method in combination with BPE dropout (Subsection 4.7).
- We explore how the proposed method behaves with ideographic characters and the usage of subcharacter information (Subsection 4.6).
- We demonstrate that our method works well for small training datasets (Subsection 4.4.5).
- We explore how the proposed method can be combined with an auxilary language pair (Subsection 5.1.1).
- We explore how the proposed method can be combined with BT (Subsection 5.2.4).

1.8. Thesis Outline

This thesis is organized in the following chapters:

Chapter 1: Introduction

The introduction outlined the state-of-the-art for low-resource NMT and the problems that motivated the contents of this thesis.

Chapter 2: Preliminaries on Neural Machine Translation

This chapter explains in detail the concepts of NMT necessary to understand the subsequent chapters.

Chapter 3: Subword Segmentation

This chapter explains in detail the concepts subword segmentation and preprocessing and their relation to low-resource NMT. We compare different settings for low-resource NMT through various experiments.

Chapter 4: Sub-subword n-gram features

We describe our novel NMT training method that utilizes sub-subword n-gram features [84] and examine its properties through multiple sets of experiments.

Chapter 5: External Data

We compare different methods to exploit external data, such as monolingual corpora or data from other language pairs, to improve NMT and run multiple experiments to explore how back-translation can be combined with the proposed sub-subword feature approach.

Chapter 6: Conclusion

We summarize the contents of this thesis and discuss some future research directions.

Chapter 2.

Preliminaries on Neural Machine Translation

This chapter describes various Neural Machine Translation concepts necessary to understand the rest of the dissertation. First there is a brief overview of NMT systems. The Word Embedding concept, and language models are described as a basis for better understanding of machine translation models. Next, the most common architectures are described, and how to generate or decode text from NMT models. The most common evaluation methods are described and, to conclude, a summary of the NMT systems pipeline is presented. All these concepts are presented with a focus on languages with few resources.

2.1. Overview

Machine translation systems produce a text from a given *source* text. The text generated must be the translation of the source text. Thus, machine translation systems can be considered conditional text generation systems. In the natural language processing literature, machine translation is not usually classified as text generation, reserving the term for systems that generate text from non-textual data, such as report generation or automatic subtitling. Machine Translation or Automatic Summarization (to take another example) are usually classified separately.

In any case, all these tasks are strongly related to language modeling, in that the generated text must be natural. Autoregressive language models are those

that model the probability of a word conditioned by its precedents. These types of models can be combined with search algorithms to generate a text that fits the model.

A model similar to the one described in the previous paragraph that also conditions the probability in a second text (the source text) can model the translation process. Neural Machine Translation systems follow an architecture known as *encoder-decoder*, according to which the *encoder* part is responsible for representing the information in the source text numerically and the *decoder* part is responsible for modeling the translation. The *encoder-decoder* approach was introduced by Cho et al. [29] and was later applied to other network architectures.

Figure 2.1 shows a simple diagram of an encoder-decoder model with the encoder and decoder separated by a dashed line. The embedding matrices and output-layer may or may not be included in the encoder or decoder. The different architectures that follow this design principle are described on Section 2.4.1.

2.2. Embeddings

NMT models are used to translate sequences of tokens, such as words, in a source language into corresponding sequences of tokens in a target language. The sets of all distinct tokens in the source and target language sentences are called the source and target vocabularies, respectively. Like other NLP models, NMT models operate based on closed vocabularies. The tokens in each vocabulary are assigned unique identification numbers.

Artificial Neural Networks are trained using *back-propagation* which works by computing the error gradient on the training data. In order to use these discrete variable tokens in a continuous function, each discrete value is represented as a *one-hot vector*. These one-hot vectors are then projected to dense vectors of size d by computing the dot product with an *embedding matrix*. The resulting dense vectors are called *embeddings*. As an example, for a vocabulary of size V, the word $k \in \mathbb{N}$ is represented by a vector $\mathbb{1}_V(k)$ of length V such that

$$\mathbb{1}_V(k)_i = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{if } i \neq k. \end{cases} \tag{2.1}$$

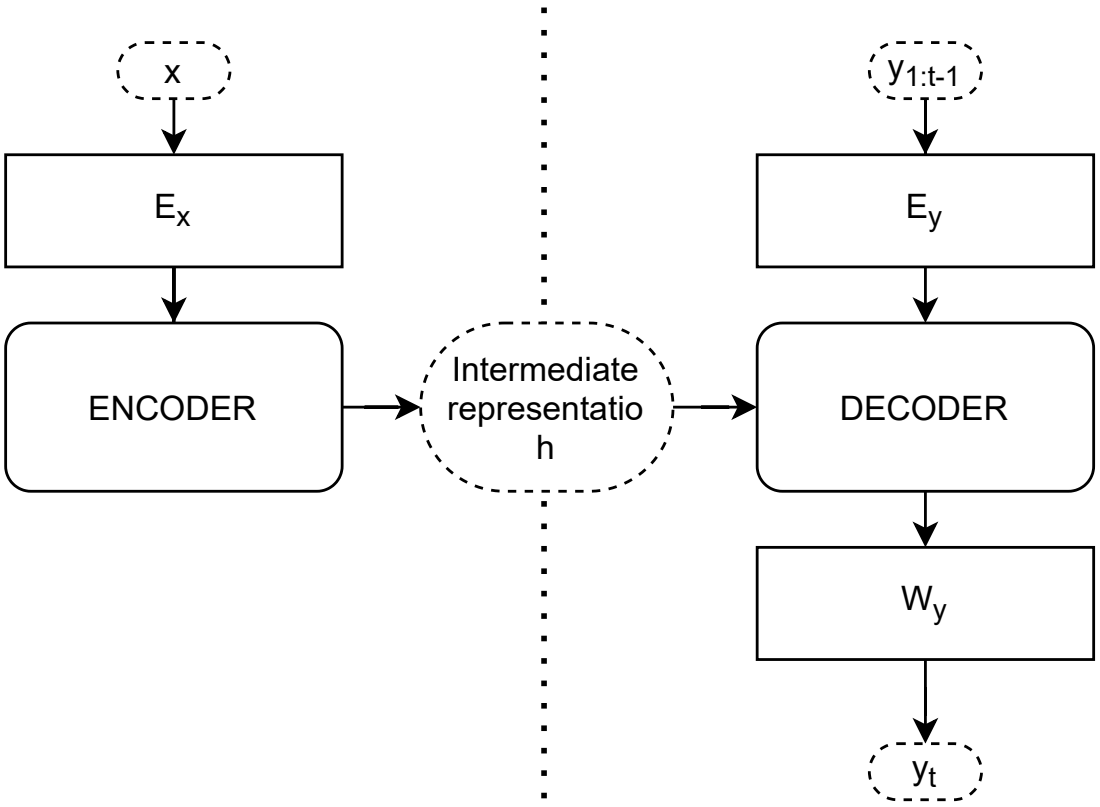

Figure 2.1.: Simple diagram showing an encoder-decoder model. The encoder- and decoder-side are separated by a dashed line. Embedding matrix E_x is sometimes considered part of the encoder, and matrices E_y and W_y part of the decoder.

The embedding $e^{(k)}$ corresponding to token k is defined with respect to embedding matrix $E \in \mathbb{R}^{d \times V}$:

$$e^{(k)} := E \cdot \mathbb{1}_V(k), \tag{2.2}$$

which is equal to extracting the kth column of E. Each embedding has an *embedding size* d.

2.3. Language Modeling

Language models (LM) estimate the probability of a sentence or piece of text. A language model trained on a given corpus learns to assign a probability for any sequence of tokens y to occur in such a corpus. If the training corpus is from a particular language and large enough, the language model approximates the probability that a sequence of tokens is written in that particular language.

To model the probability of a sequence $\mathtt{y} := (y_1, \ldots, y_T)$, the language models usually decompose the probability into the product of the probability of each of its tokens conditioned on the previous tokens:

$$p(y) := \prod_{t=1}^{T} p_t, \tag{2.3}$$

$$p_t := p(y_t | \{y_1, \ldots, y_{t-1}\}). \tag{2.4}$$

This kind of language models are called *autoregressive*.

Token sequences are usually bracketed with the special symbols `BOS` (beginning-of-sentence/sequence) and `EOS` (end-of-sentence/sequence), also referred to as `<s>` and `</s>`. Another common symbol is `OOV` (out-of-vocabulary), also referred to as `<UNK>`, which works as a catch-all symbol for the tokens not found in the vocabulary.

To train language models, no annotated data is necessary, just plain text. The models are trained to predict the token following a sequence of tokens $(y_1, \ldots, y_{t-1})$. This type of training is called *unsupervised training*. Each token in the sequence is represented in the way described on Section 2.2, so $(e_1, \ldots, e_{t-1})$ is a matrix $\mathbb{R}^{(t-1)\times d}$.

There are different types of architectures that conform to the equation

$$q_t := \mathrm{LM}(\{e_{0<j<t}\}), \tag{2.5}$$

which condenses the contextual information necessary to predict y_t into a single vector q_t. These architectures are explained in more detail on Section 2.4.1.

The vector q_t is projected to a vector the size of the vocabulary, of which each value represents the logit for the corresponding token. These values are

transformed to a probability distribution using the Softmax function:

$$p_t := \text{Softmax}(W_y q_t + b_y), \tag{2.6}$$

$$\text{Softmax}(z)_k := \frac{e^{z_k}}{\sum_l e^{z_l}}, \tag{2.7}$$

where W_y is a matrix of size $V_y \times d_o$ and b_y a bias vector of size V_y. The embedding matrix, output layer matrix W_y, and bias b_y are trained as conventional weight parameters.

An *autoregressive language model* can be used to generate natural-sounding sentences, or sequences that imitate the training data, from a prompt sequence. The details of the decoding process are explained in Section 2.5.

2.4. Neural Machine Translation

Neural Machine Translation systems work in a similar way to LMs described in Section 2.3 but conditioned on a source input sequence. The input tokens of the source sentence `x` are represented as a sequence of one-hot vectors of the corresponding tokens $\texttt{x} := (x_1, \ldots, x_{T_x})$, where T_x is the number of tokens in the sentence. To output a probability distribution p_t for a token y_t, the previous output tokens $\texttt{y} := (y_1, \ldots, y_{t-1})$, as well as the input `x`, are conditioned. These tokens are then projected into dense vectors $e_i^{(x)}$ and $e_j^{(y)}$ using embedding matrices E_x and E_y, in the way described on Section 2.2.

The source sequences are usually bracketed with `BOS` and `EOS`, same as described for the target sequence in Section 2.3.

NMT models use two embedding matrices, namely E_x for source sentence words and E_y for representing words in the target sentence. When using embeddings of d dimensions and vocabularies of sizes V_x and V_y, the embedding matrices E_x and E_y are in $\mathbb{R}^{d \times V_x}$ and $\mathbb{R}^{d \times V_y}$, respectively.

Given an input embedding sequence $e^{(x)}$ and previous translation tokens' embedding sequence $\{e^{(y)}_{0<j<t}\}$, where the *NMT* model produces a dense vector q_t of size d_o corresponding to the next token in the translation.

$$e_i^{(x)} := E_x x_i, \tag{2.8}$$

$$e_j^{(y)} := E_y y_j, \tag{2.9}$$

$$q_t := \text{NMT}(\{e_{0<i\leq T_x}^{(x)}\}, \{e_{0<j<t}^{(y)}\}), \tag{2.10}$$

where *NMT* can be any architecture described on Section 2.4.1.

The vector q_t is then projected into a vector of size V_d whence the Softmax function produces a probability distribution, as per Equations 2.6 and 2.7.

If both the source language and target language are represented by the same set of tokens, they can share an embedding matrix, resulting in $E_x = E_y$. This is a common technique used in conjunction with subword segmentation, described on Chapter 3, if the same subword segmentation model is used for both source and target languages.

Another technique called *weight tying* [103] consists in defining W_y as the transpose of E_y. If source and target embeddings are shared and weight tying is used, $E_x = E_y = W_y^\intercal$. All the experiments in this dissertation follow this approach.

There is also ongoing research on *non-autoregressive neural machine translation* (NAT) [55, 53] that work in a different way than described here. The strong point of these models is the decoding speed at the expense of a reduction in the quality of the translations. These models are outside the scope of this dissertation.

2.4.1. NMT architectures

The first NMT models used Recurrent Neural Networks (RNN), also known as *(RNN) sequence-to-sequence* models [123]. RNNs allow variable length sequences to be processed. They do this by processing each element of the sequence one by one in order. The order of processing can be from first to last or vice versa. Bidirectional RNNs run through the sequence in both directions.

In early NMT models the encoder produced a vector representing the source sequence. The decoder used the information from this vector to generate the target sequence. A problem with this method is that the vector produced is of a fixed size, in such a way that all sequences have to be compressed to this size, regardless of their length. These models had difficulty translating long sequences, since it is difficult to compress very long sequences into a reduced vector.

The solution to this problem was the attention mechanism[8]. This mechanism makes it possible to produce a single vector from a set of vectors. In *sequence-to-sequence with attention* models, the encoder produces a vector for each element of the sequence and the decoder uses an attention mechanism at each decoding step to obtain a different representation of the source each time. The attention mechanism learns to select the source vectors that contain the relevant information for each step. Figure 2.2 shows a diagram of this architecture.

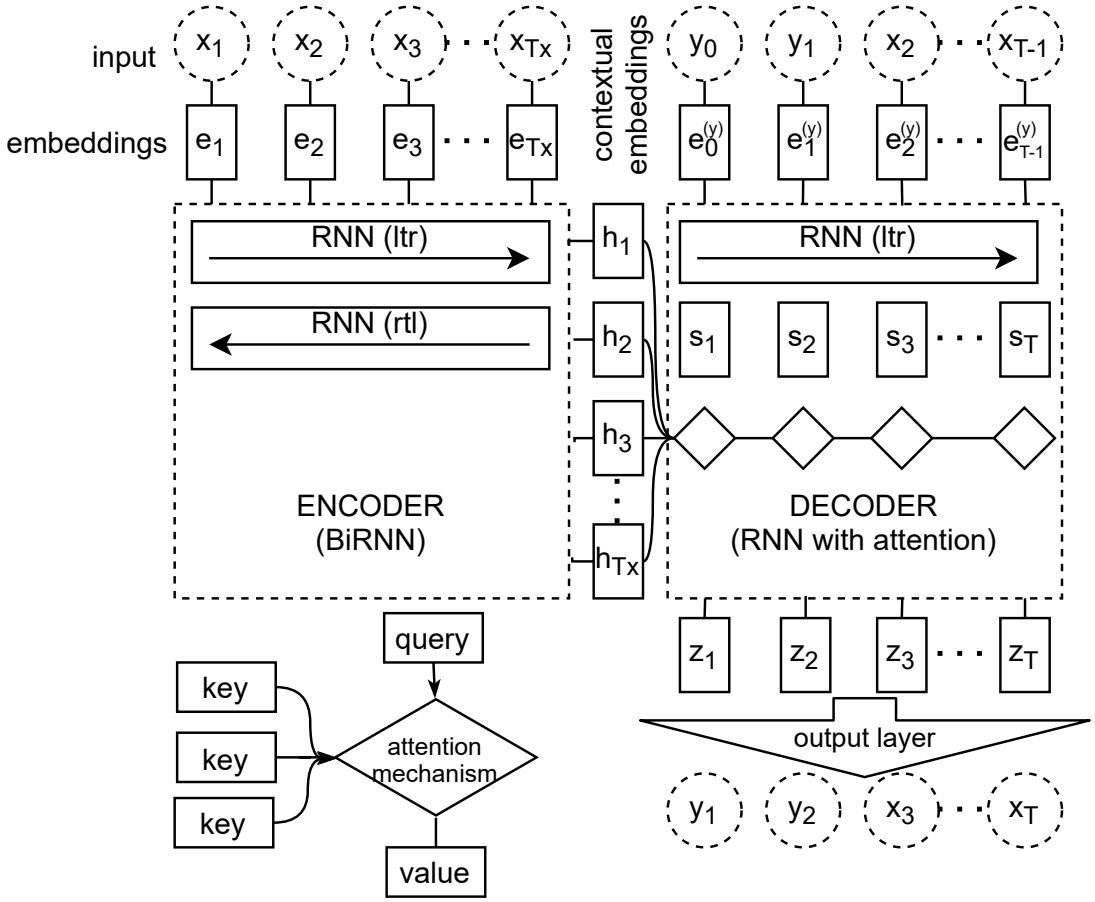

Figure 2.2.: A diagram of a sequence-to-sequence with attention architecture. The encoder produces a set of vectors that are used as keys to the attention mechanism of the decoder.

An alternative or complement to RNNs are CNNs. NMT systems with CNN have achieved good results [49, 50].

In recent years, the most popular NMT systems are based on Transformer [131]. Transformer discards the RNNs to use the attention mechanisms instead. This type of attention mechanism is called *self-attention*, which is nothing other than the same attention mechanism used by sequence-to-sequence with attention but applied to the immediately previous layer. The key idea that made Transformer possible was positional embeddings. The attention mechanism processes all vectors in an order-independent fashion. Adding the information about the position of the vectors allows considering their context. Furthermore Vaswani et al. [131] introduced multi-head attention, which is equivalent to several attention mechanisms running in parallel, and made a smart use of layer normalization [7] and label smoothing [124]. They explored multiple combinations of hyperparameters, training with a learning rate schedule and warm-up.

Several improvements and optimizations to the original model have been suggested [1, 126]. Figure 2.3 shows a diagram of the Transformer architecture.

In this dissertation, all experiments use the Transformer implementation provided by sockeye[61].

2.5. Decoding

The most straightforward method of generating translations from NMT models is to generate tokens one by one starting with the Beginning-Of-Sequence symbol and until the End-Of-Sequence symbol appears. This method is known as *1-best greedy search*. However, this method is vulnerable to the *garden-path problem*.

This problem occurs when the model generates the wrong token. As each token depends on its precedents, a wrong choice can compromise the quality of the translation from that point on, since the algorithm is not able to reverse the choice.

The most popular decoding algorithm is known as *Beam Search*[52, 20, 123], which alleviates this problem by counting a specified number of candidates. This number is called *beam size* or *beam width*. Beam search is a breadth-first tree search where only the *beam width* ω most promising paths are tracked. Each candidate or *beam* has an assigned score that is usually the negative of the logarithm of the product of the probabilities of each of its tokens, which is the same as the

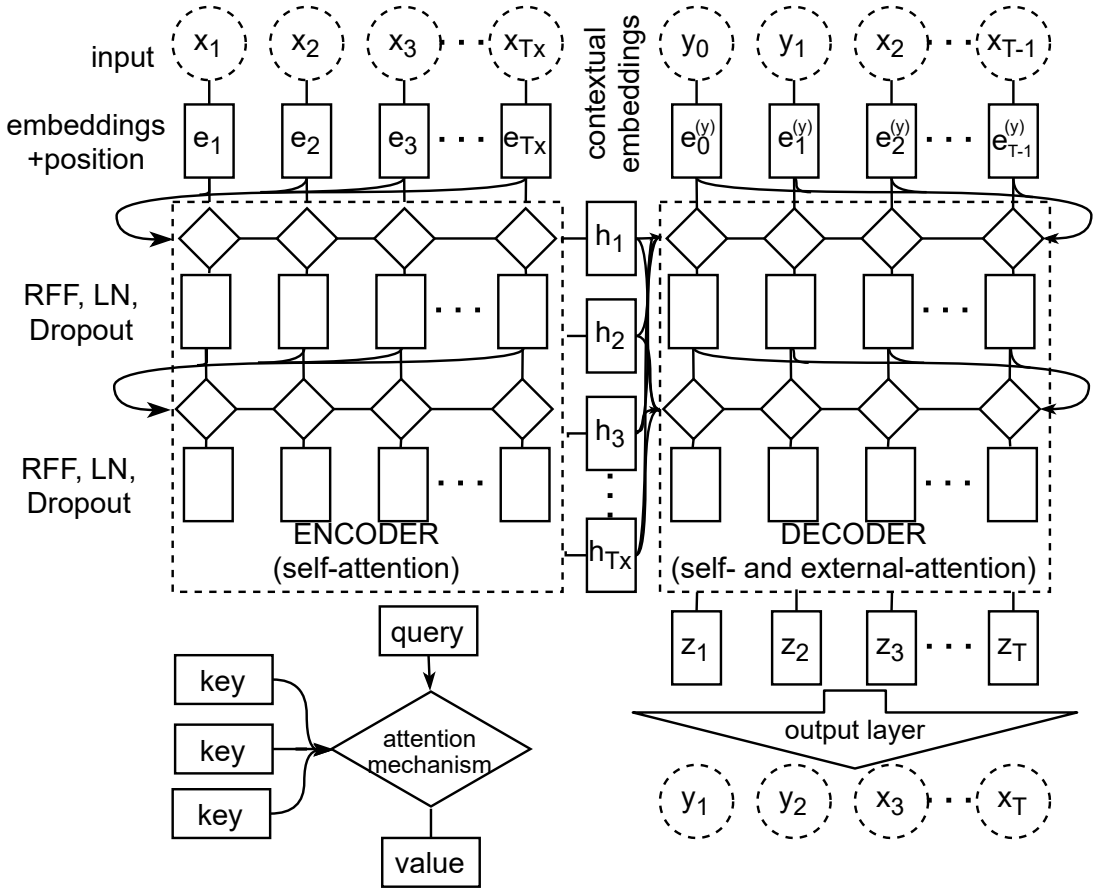

Figure 2.3.: A diagram of a simplified Transformer architecture with two layers in the encoder and decoder. Compare this figure with Figure2.2

negative cumulative log-probability.

$$\begin{aligned} \mathrm{S}(\{y_1, \ldots, y_t\}, x) &:= -\log \prod_{i=1}^{t} p(y_i | \{y_0, \ldots, y_{i-1}\}, x) \\ &= -\sum_{i=1}^{t} \log p(y_i | \{y_0, \ldots, y_{i-1}\}, x). \end{aligned} \tag{2.11}$$

In this way, the best candidates have the lowest score. Note that if $\omega = 1$, *Beam Search* is the same as *Greedy Search*.

At each decoding step, the ω candidates with the lowest scores are selected. NMT models produce a probability distribution for each *beam*. The negative log-probability of each token is added to the corresponding *beam score*. Thus, for

a vocabulary V_y, there are $\omega \times V_y$ candidates. The best ω are selected and the algorithm iterates until all the candidates finish in EOS.

The standard *Beam Search* algorithm has another known problem, which has been called the *Beam Search Curse* [140]. This problem prevents large beam widths from being used. Since the log-probabilities are always negative, the score of the candidates can only increase. It is for this reason that shorter translations tend to score lower. In practice, searches with large ω produce shorter translations and this brevity conflicts with the BLEU *brevity penalty* described in Section 2.7.1. A short translation often incurs the *under-translation* problem whereby some of the information in the source sequence is omitted.

Various methods have been proposed to alleviate or counter this problem, such as *length normalization* which normalizes the score by the length of the sequence.

$$\mathrm{S}_{\mathrm{LN}}(\{y_1, \ldots, y_t\}, x) := \frac{\mathrm{S}(\{y_1, \ldots, y_t\}, x)}{|y|} \tag{2.12}$$

This method is widely used for its simplicity.

On their *GNMT* system, Wu et al. [139] re-formulated length normalization as

$$\mathrm{S}_{\mathrm{LP}}(\{y_1, \ldots, y_t\}, x) := \frac{\mathrm{S}(\{y_1, \ldots, y_t\}, x)}{\mathrm{LP}(y)}, \tag{2.13}$$

$$\mathrm{LP}(y) := \frac{(K + |y|)^\alpha}{(K + 1)^\alpha}, \tag{2.14}$$

with $K = 5$ and hyperparameter α being the length normalization coefficient. They also introduced *coverage normalization* based on the attention matrix of *sequence-to-sequence with attention* models. He et al. [60] proposed a word-reward to improve the score of longer sequences, and Huang et al. [62] made improvements to this method.

Yang et al. [140] they analyzed the problem by comparing various solutions and proposed *BP-Norm*. *BP-Norm* gave the best results among the proposed systems.

In all the experiments in this dissertation, we used the LN of Wu et al. [139] with $K = 5$, $\alpha = 0.6$, and $\omega = 4$. We used this function for simplicity, but it is possible that *BT-Norm* with a large ω would have yielded better results.

Another method that could improve the results in low-resource settings is to use a language model in combination with *BeamSearch* to re-rank the generated candidates [57, 58]. We discuss this method in Section 5.2.1.

2.6. Neural Machine Translation Pipeline

A standard NMT pipeline is divided into two phases: training and decoding. In this section we break down the different steps of the pipeline and discuss some details.

Figure 2.4 shows a diagram of a standard NMT pipeline.

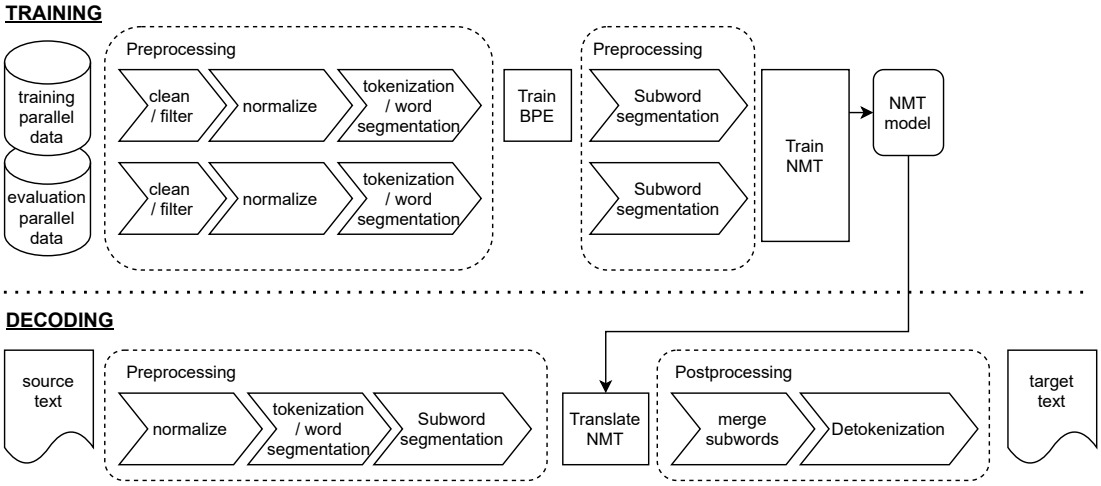

Figure 2.4.: Diagram of a standard NMT pipeline. Training and decoding phases are separated by a dotted line.

Training is divided between data preparation and model training. The training and evaluation data, necessary to train the model, are cleaned or filtered. Usually this step involves removing some of the data. To do this, it is common to use a language detection tool to remove misaligned data. In addition, texts that are too long or with a length ratio that is far from the average are eliminated.

Next, or before, the texts are normalized. The most common things to do in this step are to remove strange characters, or normalize punctuation marks and character variants. The most popular option is the Moses [74] punctuation

normalizer script (`normalize-punctuation.perl`). In some cases, all texts are converted to lower case or a truecasing model is trained.

After that, comes the step of tokenization or word segmentation, depending on the language. Languages like Chinese or Japanese, which do not separate words with spaces, use a technology known as word segmentation to separate the words. Some word segmentation systems for Japanese are Mecab[76], KyTea [93], Juman[77] and Juman++[91, 128]. In other languages, tokenization consists of separating the punctuation and numbers from alphabetic words. This tokenization is usually carried out by means of rules. The most popular option is Moses' tokenizer script (`tokenizer.perl`). However, it does not include rules for many languages. In these cases, it is possible to use the rules of English or a language that is similar to the one intended, although this can introduce noise.

These three steps (cleaning, normalization, and tokenization) are collectively known as preprocessing.

Once the texts have been tokenized, a BPE model is trained. The BPE model, described in Section 3.4, is trained exclusively on the training data. The BPE model, once trained, is used to apply the subword segmentation to the training and evaluation data. Sometimes subword segmentation is also included in the preprocessing. These data can be used to train an NMT model.

With a trained NMT model, texts can be translated using an NMT system. Text is preprocessed similar to training data, but without the filtering step. Once the subword segmentation is applied, the NMT model decodes a translation, which is then post-processed.

Post-processing includes merging the subwords and de-tokenizing the text, putting together punctuation and words according to the practice of the target language.

2.7. Evaluation Metrics

There are two types of evaluation for machine translation systems: human evaluation and automatic evaluation. The human evaluation is the most reliable because a human who knows the two languages of the pair can consider the different possible translations and paraphrases. Automatic evaluation consists of

comparing the translation produced by the system with one or more reference translations produced by professional translators. However, human evaluation is expensive since it requires the intervention of one or more people. That is why there is automatic evaluation.

The annual WMT conference [18, 19, 10, 11] is one of the most popular conferences in machine translation that also organizes a translation systems competition. Enrolled systems are evaluated using human evaluation and automatic evaluation. The human evaluation uses a method called *direct assessment* (DA) [51]. This dual evaluation allows you to calculate the Spearman's correlation coefficient or Pearson's correlation coefficient to find out how correct the automatic evaluation is.

There are several automatic evaluation systems, including BLEU[97], TER[120], BEER[122], YiSi-1[81], ESIM[25, 85] and chrF[99]. Of all the systems, the most popular is undoubtedly BLEU which, although they have some drawbacks, shows a good correlation with human evaluation.

In a recent study, Mathur et al. [86] compared multiple metrics' correlation. In the conclusions, they recommend against using BLEU and TER, and recommend to use chrF, YiSi-1 and ESIM instead. However, YiSi-1 and ESIM both make use of contextual embeddings (such as BERT [39]), which are not usually available for low-resource languages.

BERTScore [144] and BLEURT [114] are two other evaluation methods that use BERT. BERTScore (BS) compares the tokens in the candidate and reference sentences using cosine similarity. BLEURT uses a model trained with synthetic data generated from small perturbations in the Wikipedia data. Both methods show greater correlation with human evaluation than other methods. In Subsection 4.4.2 we use these methods to compare the performance of various systems.

Next, we discuss BLEU and CHRF, which are used in the experiments of this dissertation. Both systems have been used in the last editions of WMT.

2.7.1. BLEU

BLEU [97] is by far the most popular assessment method, and is often featured in most NMT publications.

BLEU computes a score for a produced translation y with w.r.t. a reference

translation (or set of reference translations) y^*. The score is a value betwenn 0 and 1, where 1 represents a perfect translation, but it is usually reported as percentages between 0 and 100. It is a combination of the precision of the n-grams in translations, and a brevity penalty to avoid translations that are too short.

$$\text{BLEU}(y, y^*) := \text{bp}(y, y^*) \frac{\sum_{n=1}^{N} \log p_n(y, y^*)}{N}, \tag{2.15}$$

where N is usually 4, p_n is the precision of the n-grams and bp the brevity penalty

$$\text{bp}(y, y^*) := \min\{1, e^{1-1/\text{lr}(y,y^*)}\}, \tag{2.16}$$

$\text{lr}(y, y^*)$ being the length ratio between the produced and reference translation.

$$\text{lr}(y, y^*) := |y|/|y^*|. \tag{2.17}$$

BLEU normally operates on word n-grams but it can also be applied to character n-grams, in which case it is referred to as *CharacterBLEU*. CharacterBLEU is normally used with languages that do not use white-space between words and for which tokenization can be complicated, such as Chinese or Japanese.

Before calculating the BLEU score, the segments are tokenized using the same tokenizer, which is usually the Moses script used by WMT. However, there are multiple ways to tokenize the same segment. An example is that GNMT[139] system separated the words joined by a hyphen into three tokens. As an example, the word "six-pack" became

- [`"six"`, `"##AT##-##AT##"`, `"pack"`],

where `"##AT##-##AT##"` represents the joining hyphen. This approach yields slightly higher BLEU scores.

The BLEU score is sensitive to the type of tokenization used and it is necessary to use the same tokenizer to be able to compare two systems.

Sacrebleu[102] is a tool that was presented with the idea of facilitating comparable and reproducible results by including tokenization in its pipeline.

All the results of this dissertation use sacrebleu.

2.7.2. CHRF

Although characterBLEU can be used with characters, it works on the precision of n-grams of characters, while CHRF[99] works on the F-score. CHRF has been shown to have a high correlation with human assessment.

The CHRF score is calculated as follows:

$$\text{CHRF}\beta := (1+\beta^2)\frac{\text{CHRP}\cdot\text{CHRR}}{\beta^2\cdot\text{CHRP}+\text{CHRR}}, \tag{2.18}$$

where CHRP is the n-gram precision and CHRR is the n-gram recall. When the parameter $\beta = 1$, both the recall and precision are assigned the same importance.

The n-gram precision and recall are computed from $n = 1$ up to parameter N. Our experiments use $N = 6$ and $\beta = 2$, according to the recommendations of [100].

There is also CHRF++[101] that combines the punctuation of n-grams of characters with the punctuation of n-grams of words.

CHRF is also included in sacrebleu.

2.8. Tokenization Experiments

Table 2.1 shows the BLEU and CHRF2 scores using different preprocessing options, for an low-resource corpus (205,000 pairs of parallel sentences) English-Turkish[127]. In particular, the results of using or not using both, the Moses[74] tokenizer script (with filename `tokenizer.perl`) and punctuation normalizer script (with filename `normalize-punctuation.perl`), are compared.

The experiments use BPE (see Chapter 3) with a vocabulary size of 8,000 subwords. Instead of the mentioned scripts, the equivalent implmentations from sacremoses* were used.

Although the difference in BLEU scores is not significant, we see that the best results are obtained without tokenizing or normalizing the data. The models with best BLEU and CHRF2 scores match in this case. The scores of the model trained on normalized punctuation but no tokenization are 13.37 BLEU and 0.446 CHRF2. The scores of the model trained on normalized punctuation and tokenization are

*Found at `https://github.com/alvations/sacremoses`

Table 2.1.: BLEU and chrF2 scores using different preprocessing options. Scores are formatted as in "BLEU (chrF2)" with the best BLEU score for each direction in bold and the best chrF2 score underlined. The values under `(+/-) punctuation` refer to those using the normalizer script and the values under `(+/-) tokenize` refer to those using the tokenizer script.

	-punctuation		**+punctuation**	
	-tokenize	**+tokenize**	**-tokenize**	**+tokenize**
en → tr	**13.68** (.470)	12.79 (.465)	13.37 (.464)	13.13 (.469)
tr → en	**16.22** (.446)	15.44 (.441)	16.07 (.445)	15.12 (.437)

13.13 BLEU and 0.469 chrF2. We can see that these results of BLEU and chrF2 contradict each other, as the BLEU scores suggest it is better to avoid tokenization, but the chrF2 scores suggest it is better to tokenize. However, rest of the results show a tendency in favor of not using these scripts.

The reason for these results may be that the Moses scripts are not prepared for Turkish data. García-Martínez et al. [48] were aware of this problem and used a modified version of the tokenizer scripts in their experiments.

2.9. Conclusions

This chapter has presented the rudiments of NMT. This knowledge is necessary to understand the rest of the dissertation.

The pipeline of a standard NMT system using BPE has been presented. The preprocessing steps explained in that pipeline have not changed from SMT systems, with the exception of subword segmentation.

In addition, the same type of preprocessing is often used for low-resource and non-low-resource pairs. Our experiments show that for a low-resource pair, which does not have specific tokenization and normalization rules implemented, it is better to use the text without tokenizing or normalizing. This option may depend on the training data and how clean or noisy it is.

Chapter 3.

Subword segmentation

As mentioned in Chapter 1, word-level vocabularies can be very large, particularly for morphologically complex languages. *Subword segmentation* is used to reduce vocabulary sizes as an alternative to selecting the top-N most frequently used words.

This chapter explains the concept of subword segmentation, its motivation, and different methods. Some techniques to improve your performance are presented. The chapter concludes with a set of experiments that illustrate some of the ideas.

3.1. Motivation

As explained in Section 2.6, languages such as Chinese or Japanese that do not separate words with spaces often require an additional preprocessing step known as *word segmentation*. This step is often compared with the tokenization step used for alphabetic languages. This segmentation is usually carried out using dictionaries and morphological analyzers such as Mecab[76], KyTea [93], Juman[77] or Juman++[91, 128].

As shown in Figure 1.3, word vocabularies follow a near-Zipfian distribution and that is why much of the vocabulary is rare. This creates a problem, especially under low-resource conditions and with morphologically complex or agglutinative languages. In agglutinative languages, a single very long word can contain a large number of morphemes.

In addition, proper nouns and named entities are added to the vocabulary, making it impossible to maintain the vocabulary as a closed list. In some domains,

such as the biomedical domain, the number of named entities is always growing, as new drugs and terms of other kinds are created.

With a closed vocabulary, a model has to translate words that did not appear in the training data. In these cases, the best word NMT systems can do is to use a copying or transliteration mechanism, but these add complexity to the system and a point of failure.

3.2. Subword Segmentation

The number of different tokens can be reduced by splitting words into smaller subword units. Subword segmentation increases the length of input and output sequences, and smaller vocabularies result in longer sequences. The optimal vocabulary size depends on the training data, where smaller datasets typically favor smaller vocabularies.

Once data have been segmented into subword units, they are treated as independent tokens in the manner described in Section 2.4. The word embedding matrix can then be considered as a subword embedding matrix containing subword features.

The information regarding which characters comprise a token is ignored. The potential benefits of taking advantage of this information represent the main motivation for the use of *sub-subword features* as described in Chapter 4.

3.3. Linguistically Grounded Approaches

The most popular subword segmentation systems are unsupervised, but there is an option to use linguistically grounded approaches instead. The word segmentation systems for Chinese and Japanese mentioned in Section 3.1, such as Mecab and KyTea, can be classified as subword segmentation systems in a broad sense. Other systems, such as morfessor [133], are designed for alphabetic languages, such as Finnish or English, and separate words into morphemes or morpheme-like units (morphs).

Morpheme segmentation has been used in NMT as an alternative to BPE [9, 13, 41, 96]. Saleva and Lignos [107] compared this type of system to BPE

(explained in Section 3.4) and concluded that these systems do not provide any advantage over BPE.

Furthermore, both the word segmentation systems and the morpheme-based systems do not provide a clear way to segment proper nouns and foreign words.

One more option may be to divide the words using rules. For example, words can be split at certain characters or divided into syllables. This option may be valid if the system languages have a simple syllable system.

3.4. BPE

One very popular subword segmentation method is byte pair encoding (BPE) [47], which was originally proposed as a compression algorithm in 1994, but introduced to NMT by Sennrich et al. [116] in 2015.

BPE is a bottom-up method, since it builds the vocabulary starting with the most granular segmentation and increases the vocabulary until it reaches the desired size.

The BPE subword vocabulary starts with the set of all individual characters in the training data. The final characters in words are considered to be distinct from their counterparts in other locations. A vocabulary is augmented until it has the desired size by merging frequent subword pairs to form new subwords.

There are various approaches to word segmentation in the literature that are similar to BPE. One such approach is the *WordPieceModel* [109]. Instead of merging subwords based on their frequency, this model merges subwords to optimize a likelihood value for a language model. Different merging policies were explored in [138], including frequency, accessor variety and descriptor length gain.

Algorithm 1 is the generalized BPE segmentation proposed by Wu and Zhao [138].

A common practice in NMT systems is to train a single BPE model for the source and target languages jointly, such that the source and target vocabularies are the same. This practice allows a word to be segmented in the same way in both languages. Thus proper nouns and named entities are represented in the same way and are easier to copy. This also helps with transliteration as shown by Sennrich et al. [116].

Algorithm 1: Generalized BPE segmentation algorithm as described by Wu and Zhao [138], re-formatted.

Data:
D, the training corpus;
N, merge times;
g, goodness measure;
Result: the segmented text D' and merge list V
$V \leftarrow [\,]$
$D' \leftarrow$ {Training corpus D with all words split into individual characters.}
for $i \leftarrow 1$ **to** N **do**
    Calculate the goodness scores of all the distinct successive substring pairs according to g
    $V \leftarrow$ append highest scoring pair
    $D' \leftarrow$ merge all the occurrences of the chosen pair
end
return D', V

Figure 3.1 shows an example of BPE segmentation of word *internationalization*. Starting with character-level segmentation, as the size of the vocabulary increases through merge operations, the word is represented by less subwords. The character n at the end of the word is distinguished from n characters within the word. If the word *internationalization* appeared in the corpus, increasing the number of merge operations would include unsegmented word in the vocabulary at some point. With 1,984 merge operations, the word is segmented into *inter-nation-al-iz-ation*, which is closest to morpheme segmentation.

3.5. Unigram Language Model

Another approach to word segmentation is the unigram language model [75], which can produce multiple segmentation candidates. In this model, the subword vocabulary is selected to optimize the probabilities of words by considering the probability of one word to be the product of the unigram probabilities of its subwords.

merges	new	segmentation of word *internationalization*
0		i n t e r n a t i o n a l i z a t i o n$
2	i +n	**in** t e r n a t i o n a l i z a t i o n$
6	t +i	in t e r n a **ti** o n a l i z a **ti** o n$
7	e +r	in t **er** n a ti o n a l i z a ti o n$
8	o +n	in t er n a ti **on** a l i z a ti o n$
16	o +n$	in t er n a ti on a l i z a ti **on$**
24	a +l	in t er n a ti on **al** i z a ti on$
25	a +ti	in t er n **ati** on al i z **ati** on$
72	ati +on	in t er n **ation** al i z ati on$
80	ati +on$	in t er n ation al i z **ation$**
94	t +er	in **ter** n ation al i z ation$
212	n +ation	in ter **nation** al i z ation$
294	in +ter	**inter** nation al i z ation$
1984	i +z	inter nation al **iz** ation$
5067	inter +nation	**internation** al iz ation$
18201	iz +ation$	internation al **ization$**
30267	al +ization$	internation **alization$**

Figure 3.1.: An example of BPE subword segmentation. Symbol n$ represents end of word *n*, distinct from character n. New merges in bold.

Kudo [75] released an opensource implementation called *sentencepiece*[*] with the paper.

A drawback of BPE is that it is deterministic. A word is always segmented into the same subwords. This is okay for inference, but during training, the model is only exposed to one possible segmentation. The unigram language model allows addressing the determinism problem of BPE through the *subword regularization* method. Since the unigram language model segments words probabilistically, it allows to sample different segmentations using the Viterbi algorithm.

The models trained with subword regularization are exposed to more training data through data augmentation. This results to more robust embeddings and models that better understand the composition of words. This improvements are reflected in BLEU scores

[*]Available at `https://github.com/google/sentencepiece`

3.6. BPE dropout

BPE dropout [104] shares a goal with the unigram language model objective and its subword regularization method. Provilkov et al. [104] introduced BPE dropout method as an alternative to Kudo [75]. The main drawback that they find in the subword regularization method is its complexity, since it requires training a unigram language model instead of BPE, which is simpler, and to sample segmentations, it uses EM and Viterbi algorithms.

BPE dropout works on BPE models, that is, the vocabularies are built in the same way as vanilla BPE. While the unigram language model subword regularization method uses a statistical model and dynamic programming to be able to sample different segmentations from the same sequence, BPE dropout uses random noise to discard certain merges, randomly generating a different sequence of subwords each time. This is so because BPE does not store the frequencies of each subword, only the order of the merges. Merges are discarded with a probability p, which is usually 0.1.

Provilkov et al. concluded through several experiments that BPE dropout achieves better results, produces better embedings and models that are more robust to noise.

3.7. Character Vocabulary

As explained in Section 3.4, the minimum size of a BPE vocabulary is given by the number of different characters in the data N_c. A system with a BPE vocabulary with zero merges is basically a character level system, and has a vocabulary size between N_c and $2N_c$, depending on the number of characters that appear at the end of the words.

A language written with Latin characters usually uses less than a hundred characters, including lowercase letters, uppercase letters, digits, punctuation marks and special characters or characters with diacritical marks. This is true for clean and processed data. However, if the data is not clean it can contain all kinds of characters as characters from other writing systems and *emojis*. This problem is most obvious with monolingual data, as it is often raw, compared to parallel data.

Parallel data is typically cleaned up as part of the alignment process[108, 106].

For example, on many occasions monolingual data is extracted from Wikipedia. Figure 3.2 shows the first sentence of the Wikipedia article about Tokyo. We see that although this sentence is written in English, it contains multiple foreign characters, such as IPA characters and Japanese characters. When all the data is considered, these characters are very sparse and difficult to train.

> ′Tokyo′ (/ ′toʊkioʊ / TOH-kee-oh, / -kjoʊ / -kyoh; Japanese: 東京, ′Tōkyō′ [toːkʲoː]), officially the ′Tokyo Metropolis′ Japanese: 東京都, ′Tōkyō-to′) , is the de facto capital and most populous prefecture of Japan.

Figure 3.2.: An example of *English* text extracted from wikipedia. The text contains many non-English characters.

The most common solution to this problem is usually to drop the characters below a frequency threshold. However, this solution creates the new question of what the cutoff frequency should be.

Another solution is to use BPE over bytes of text encoded in UTF-8, after replacing the spaces with an underscore-like character [105, 134]. The character used to separate words was `LOWER ONE EIGHTH BLOCK`. This way, the number of characters can be limited to a maximum of 257. Wang et al. [134] analyzed this solution, which they called *BBPE*, and concluded that it allowed to create more compact vocabularies without deteriorating the BLEU scores. They also observed an improvement in the scores for the multilingual models. Figure 3.3 shows an example of a sentence encoded for BBPE.

Original	D. Trumpです。
BBPE input	44 2E E2 96 81 54 72 75 6D 70 E3 81 A7 E3 81 99 E3 80 82

Figure 3.3.: Example of BBPE input. BBPE learns a BPE vocabulary from sequences of bytes. The spaces between the bytes have been inserted for clarity.

Although the results from Wang et al. [134] show that using the bytes of the UTF-8 Unicode encoding as a basis solves the rare character problem without

damaging performance, Unicode provides another source of normalization: character (codepoint) names. Each codepoint in the Unicode standard is assigned a name, and these names are limited to ASCII characters. The name of a character describes its characteristics. Table 3.1 contains some examples.

i	LATIN SMALL LETTER I
í	LATIN SMALL LETTER I WITH ACUTE
ı	LATIN SMALL LETTER DOTLESS I
い	HIRAGANA LETTER I
イ	KATAKANA LETTER I
胃	CJK UNIFIED IDEOGRAPH-80C3
。	IDEOGRAPHIC FULL STOP

Table 3.1.: Example of Unicode codepoint names.

These names can be used to encode all printable non-ASCII characters. In this way the number of characters is limited to less than 100 characters. This type of encoding exposes information about the characters. Figure 3.4 contains a coding example.

Original	D. Trumpです。
Character reduction	D. Trump [HIRAGANA LETTER DE][HIRAGANA LETTER SU][IDEOGRAPHIC FULL STOP]

Figure 3.4.: Character reduction example. BPE can learn unique tokens to represent common sequences.

3.8. Experiments

Table 3.2 shows the results of using the character reduction described in Section 3.7, according to the example in Figure 3.4. The results are shown in comparison with the results in Table 2.1.

The size of the vocabulary is the same (8,000), as well as the rest of the hyperparameters.

Table 3.2.: Results of various models to compare the use of character reduction. This table is built on the results of Table 2.1. The results follow the format "BLEU (chrF2)". Best BLEU results are shown in bold and the best chrF2 are underlined.

		`-punctuation`		`+punctuation`	
		`-tokenize`	`+tokenize`	`-tokenize`	`+tokenize`
en → tr	All	13.68 (.470)	12.79 (.465)	13.37 (.464)	13.13 (.469)
	Reduction	**13.70** (.466)	12.99 (.466)	13.12 (.458)	12.59 (.465)
tr → en	All	**16.22** (.446)	15.44 (.441)	16.07 (.445)	15.12 (.437)
	Reduction	15.74 (.441)	15.26 (.437)	15.68 (.438)	14.52 (.431)

The new results confirm the trend observed in Section 2.8 that showed better results when neither the punctuation normalizer nor the tokenizer was used.

With regard to character reduction, although the differences in BLEU score are not significant, chrF2 shows a trend towards worse scores when using the character reduction method. In particular, character reduction gives worse results in the case of the English translation. The Turkish SETimes corpus is relatively clean and contains few non-ASCII characters in the English data, and some more in the Turkish data, due to the specific characters of the Turkish alphabet (such as, Ç, Ğ, I, İ, Ö, Ş and Ü).

Table 3.3 repeats the experiment using BPE dropout. All the systems in Table 3.3 do not use any punctuation normalizer or tokenizer.

Table 3.3.: BPE dropout with character reduction. These models do not use a punctuation normalizer and tokenizer. The results follow the format "BLEU (chrF2)". Best BLEU results are shown in bold and the best chrF2 are underlined.

		En → Tr	**Tr → En**
Base	All	13.68 (.470)	16.22 (.446)
	Reduction	13.70 (.466)	15.74 (.441)
BPE dropout	All	14.17 (.479)	**17.64** (.461)
	Reduction	**14.35** (.479)	17.18 (.457)

We observe that BPE dropout improves the results in general. These improvements come in line with those reported by Provilkov et al. [104]. The remarks on character reduction are repeated, there being a small improvement in the case of $En \rightarrow Tr$ and a worsening in the case of $Tr \rightarrow En$. In the case of the $En \rightarrow Tr$ direction, the use of BPE dropout causes the CHRF2 results to be equal.

3.9. Conclusions

This chapter has explained the concept of subword segmentation and various related techniques. There are several methods and variants of subword segmentation but among them BPE is the most commonly used. One reason for this is its simplicity.

In addition, BPE dropout provides a simple method to improve system performance. Our experiments confirm the efficacy of the method and repeat the results of Chapter 2, which indicated that it was better not to use tokenization or punctuation normalization.

The proposed method to reduce characters gives mixed results, damaging the results to some extent in some cases, and not giving significant improvements in others. Even so, the method offers a way to limit the number of characters to a small, predictable number. The effectiveness of the method with data with a greater variety of characters remains to be tested.

Chapter 4.

Sub-subword n-gram features

In this chapter, we explore different approaches utilizing n-gram features to represent translation tokens instead of learning their representations as embedding matrix parameters or using n-gram features to augment embeddings. The proposed method is the one we proposed in Martinez et al. [84]. Because we used BPE subwords to limit vocabulary sizes in our experiments, we will refer to the translation tokens as "subwords" and to their n-gram features as "sub-subword features."

First, we describe the problems that the proposed method aims to solve. Next, we introduce some preliminaries and describe our proposed method, which consists of an architecture used exclusively during the training phase, and an algorithm to select sub-subword features. Next, we analyze different aspects of the proposed method through multiple experiments. We conclude with a summary of the findings of this chapter.

4.1. Sub-subword Information Problem

One problem with the standard embedding method is that each word or subword functions as an independent discrete value, so the spelling of a word has no effect on its embedding. Thus, two graphically close words can have completely independent embeddings. This goes against the linguistic intuition, according to which, two similar words have a high probability of being semantically related.

This opacity is not a problem when a large amount of data is available to compensate for it. BPE alleviates the problem to some extent by dividing the

rarer words into multiple subwords. By segmenting the words into subwords, the model is more exposed to the composition of the words, but this exposure depends on the granularity of the subwords. For this reason, small vocabularies, which create more granular segmentations, work better under low-resource conditions.

BPE dropout further alleviates the problem by exposing different possible segmentations of the same word, but the representation of each subword still remains independent of its spelling.

4.2. Character n-gram Features

Using n-gram features to generate word vectors is not a new concept: Wieting et al. [136] used n-gram count vectors to represent words for different natural language processing (NLP) tasks. Bojanowski et al. [16] trained word vectors on a skipgram language model by representing each word as the sum of the vectors representing its n-grams. This method takes word-boundary symbols into consideration. They considered n-gram sizes ranging from three to six characters. Because representing each n-gram using a unique vector can consume excessive memory, n-grams are grouped into buckets using a hash function, where they share vector representations with other n-grams in the same bucket. Joulin et al. [64] used the same approach to develop a model called *fastText* for text classification.

The feature hashing methods do not guarantee that the feature representations will discriminate every word in the vocabulary. *FastText* uses a large hash table (the recommended size is 20,000) to minimize the risk of having the same representation for multiple words. This kind of large matrices are difficult to handle on NMT models. A feature hashing method requires a hash function to distribute the features. The optimal hash function may depend on the language of the n-grams.

N-gram vectors can complement word vectors, because words are also unique n-grams. Zhang et al. [145] used subword segmentation to complement word-level vectors with subword-level vectors for various NLP tasks that did not include NMT. They compared various subword segmentation methods introduced by Wu and Zhao [138]. Morishita et al. [90] reinterpreted BPE subwords from smaller

vocabulary configurations as features of subwords in a larger vocabulary. The embedding vector for a given subword was represented as the sum of its feature vectors. Based on memory constraints, they only used sub-subword features for input and output embeddings, and not for the final output layer.

Takase et al. [125] explored the use of n-gram features of a fixed size for generating embeddings and output layer weights for language modeling. They also conducted experiments on NMT using trigram features to produce embeddings. In contrast to other approaches, they used a self-attention mechanism to calculate a weighted sum of feature embeddings.

We briefly describe the model proposed by Takase et al. [125], as we will compare our proposed model to it. For a more detailed description, please refer to the original work.

4.2.1. Character n-gram Embeddings

Takase et al. [125] incorporated embeddings inferred from fixed length n-grams (such as bigrams) into standard embeddings by replacing (2.8) and (2.9) with the following two equations:

$$e_i^{(x)} := E_x x_i + c(x_i, \theta_c^{(x)}), \tag{4.1}$$

$$e_j^{(y)} := E_y y_j + c(y_j, \theta_c^{(y)}), \tag{4.2}$$

where $c(\omega, \theta_c)$ is a neural network with a parameter set θ_c that produces dense vector inferred from the n-gram features of word ω.

The n-gram features *augment* word embeddings without replacing them. To produce $c(x_i, \theta_c^{(x)})$ and $c(y_j, \theta_c^{(y)})$ from n-gram features, Takase et al. used what they called *multidimensional self-attention*. If the number of n-gram features of a specific word ω is given by $I(\omega) \in \mathbb{N}$, then

$$c(\omega, \{W_c, S\}) := \sum_{i=1}^{I(\omega)} g_i \odot s_i^{(\omega)}, \tag{4.3}$$

$$S^{(\omega)} := (s_1^{(\omega)}, \cdots, s_{I(\omega)}^{(\omega)}), \tag{4.4}$$

$$g_i := [\text{RowSoftmax}(W_c S^{(\omega)})]_i, \tag{4.5}$$

where $\odot$ denotes element-wise product of vectors, $[\cdot]_i$ is the i-th column of a given matrix and RowSoftmax is a softmax function applied row wise. The matrices $S^{(\omega)} \in \mathbb{R}^{D_e \times I(\omega)}$ represent the embeddings of the n-gram features in the word ω. The dimension of the embeddings is D_e. These n-gram feature embeddings and the square matrix $W_c \in \mathbb{R}^{D_e \times D_e}$ are trainable parameters, that can differ between input and output layers.

4.3. Proposed Method

The method proposed below does not alter the architecture of the NMT model and only has an effect on training. The method is presented in two parts. First the training method is explained and then the unsupervised feature selection algorithm is explained.

4.3.1. Training Method

We denote V_x and V_y as the vocabulary sizes, and d_x and d_y as the numbers of n-gram features for the input and output languages, respectively. During training, our model learns to produce embeddings and output matrices from two sparse binary feature matrices $F_x \in \mathbb{R}^{V_x \times d_x}$ and $F_y \in \mathbb{R}^{V_y \times d_y}$ for the source and target language vocabularies, respectively. Each element in the feature matrices indicates whether an n-gram feature is included in a subword. When the input and output vocabularies are the same, such as when training BPE subwords jointly, the feature matrices F_x and F_y are equal. We will refer to this part of the model as the feature-to-embedding (FTE) network. Unlike the method presented by Takase et al. [125], we use a simple feed-forward network.

Our proposed approach does not train feature matrices. Equations (2.8), (2.9), and (2.6) are replaced with the following equations:

$$e_i^{(x)} := \phi_x(F_x)x_i, \tag{4.6}$$

$$e_j^{(y)} := \phi_y(F_y)y_j, \tag{4.7}$$

$$p_t := \text{Softmax}(\phi_o(F_y)q_t + \phi_b(F_y)). \tag{4.8}$$

The parameter weights for the feed-forward network, namely ϕ_x, ϕ_y, and ϕ_o, are updated during training. If weight tying is applied, then

$$\phi_o(F_y) = \phi_y(F_y)^\intercal. \tag{4.9}$$

Figure 4.1 presents a diagram of the training model using weight tying. Compare Figure 4.1 with the standard encoder-model of Figure 2.1.

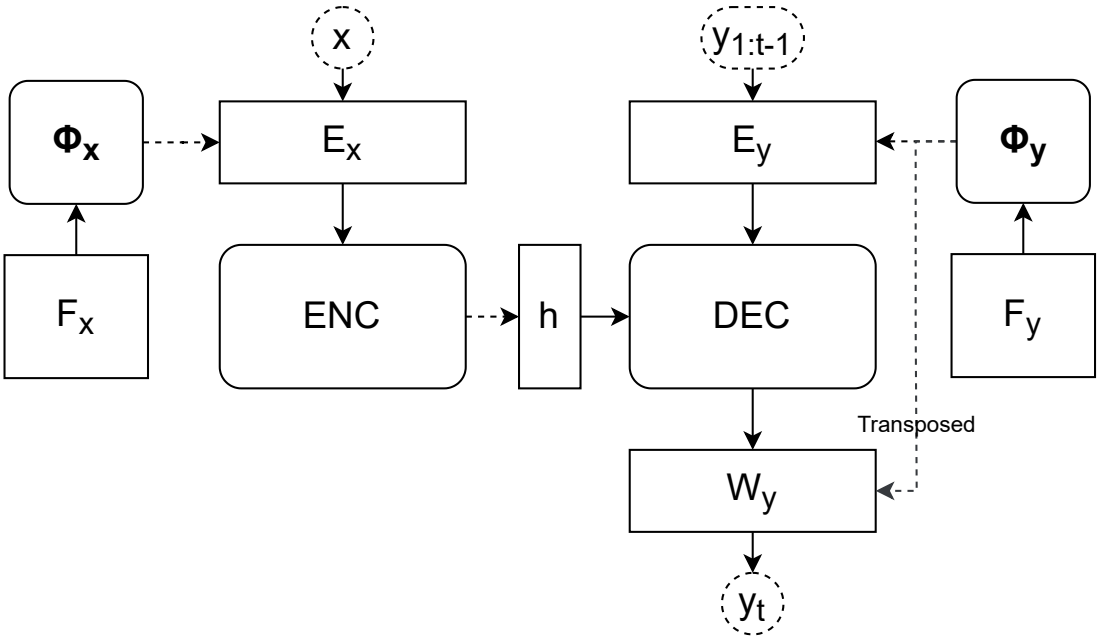

Figure 4.1.: Simple diagram showing an encoder-decoder model using features to produce embeddings. The inputs x and $y_{1:t-1}$, and output y_t are depicted with dashed borders and neural subnetworks have rounded corners. The model depicted in this diagram uses weight tying. The output bias vector b_y is not shown for simplicity. On top of the baseline model, the boxes containing the n-gram feature matrices F_x and F_y, and the feed-forward networks ϕ_x and ϕ_y have been added. E_x and E_y are not trainable parameters. Compare with Figure 2.1.

Once the model is trained, we revert to equations (2.8), (2.9), and (2.6) by

pre-computing the values of E_x, E_y, W_y, and b_y using the following equations:

$$E_x := \phi_x(F_x), \tag{4.10}$$

$$E_y := \phi_y(F_y), \tag{4.11}$$

$$W_y := \phi_o(F_y), \tag{4.12}$$

$$b_y := \phi_b(F_y). \tag{4.13}$$

After these calculations are completed, the FTE weights are no longer necessary and can be discarded. The resulting model is the same size as the base model.

Our goal is to compare different architectures and n-gram features and to determine their utility. As shown later in Section 4.4, we determined that a three-layer feed-forward network yields the best results among the compared architectures. We use a rectifier activation function between layers.

Using all of the fixed-width n-grams can result in very large feature matrices. In practice, when using all available features, we can only train models on bigram features based on memory constraints. To be able to use longer n-grams, we use a custom feature selection algorithm. We extract all n-grams for every possible n and select a final subset using the algorithm described in Section 4.3.2.

4.3.2. Feature Selection

The goal of the proposed algorithm is to select a small number of features that unambiguously represent a given vocabulary. As an example, consider a vocabulary that contains the subwords *ana* and *anana*. When only using bigram features, both subwords are represented as `{ ^a, an, na, a$ }`. In order to disambiguate these two subwords, the trigram feature `nan` may be selected. The algorithm selects the feature that disambiguates the maximum number of subword occurrences, different to feature hashing methods.

Given a large set of potential character n-gram features, the proposed feature selection algorithm selects a small subset. The initial feature-set contains features

for all n-grams contained in the subword vocabulary, where n ranges from one up to the maximum subword length in the vocabulary.

Consider a vocabulary V containing all subwords in a training dataset. The proposed algorithm selects one character n-gram feature at a time to construct a binary decision tree. Selection can be limited to a given number of features M by stopping early. The selected feature partitions the vocabulary into positive and negative subsets. The subwords in the former subset include the selected feature and those in the latter subset do not. For every subset, we select the subword with the highest unigram frequency and regard it as *reachable*; that is, the only subword represented by its selected features. If there is a tie, one of the top-frequency subwords is selected randomly to be regarded as reachable. The rest of the subwords in the vocabulary are considered to be unreachable; that is, other subwords are being represented by their selected features. The optimization objective is to maximize the total unigram probabilities of reachable subwords. The features that best partition the vocabulary in terms of objective are selected. The subword embeddings will be derived strictly from the selected character n-gram features.

Figure 4.2 shows the binary decision tree built after three selection steps on an English vocabulary of about 32,000 subwords. The selected features are `e`, `^t` and `a`. Each node in the tree shows the number of subwords included in the partition. Of all the subwords included, the one with highest unigram frequency is reachable and shown in bold. After choosing the third feature 8 subwords are reachable and 31,835 subwords *unreachable*. Appendix A contains the full set of features selected.

Algorithm 2 presents the process described above in the form of pseudo-code. At each step, there is a set of partitions G and candidate features. Considering all combinations of these partitions and candidate features is very expensive. Instead of considering all partitions in G, we only consider a subset $\Phi(G)$ containing the partitions with highest probabilities $p_g(g)$, defined in Algorithm 2, summing up to a predefined threshold T_g. When selecting a candidate n-gram feature f, we also consider a fixed number (N_Ψ), which is denoted as $\Psi(G)$. $\Psi(G)$ consists of the features with the top-N_Ψ highest $\Omega(f, G)$ values. $\Omega(f, G)$ ranks the relevance of the features with respect to their frequency within the partitions, and it is

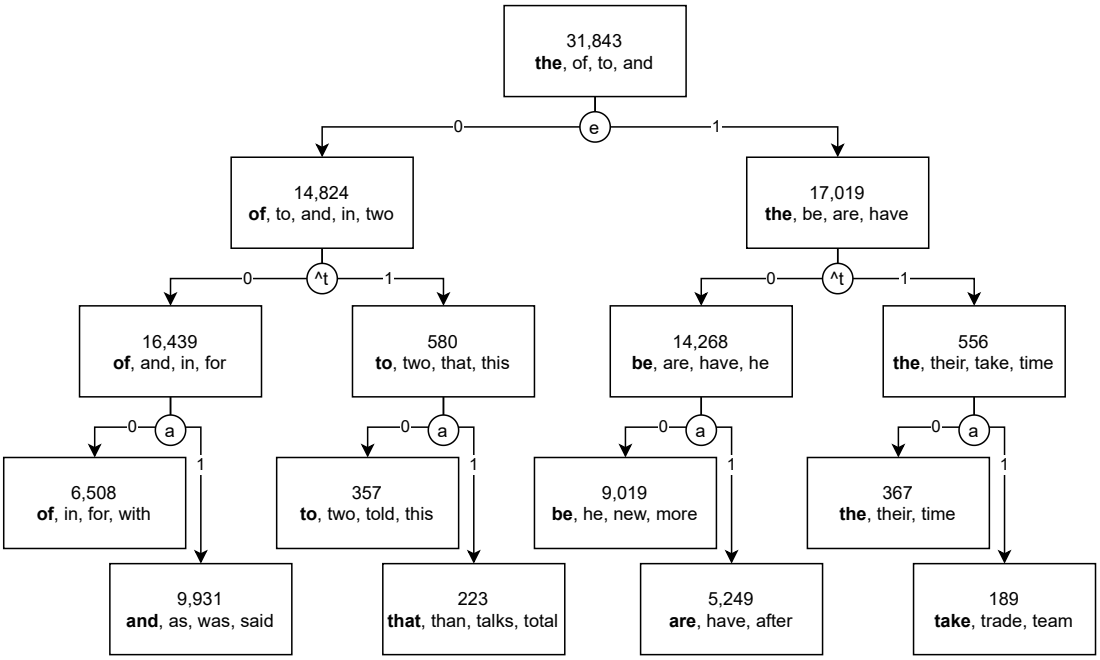

Figure 4.2.: Example of feature selection tree after three selection steps. The selected features are `e`, `^t` and `a`.

defined as follows:

$$\Omega(f, G) := \sum_{g \in \Phi(G)} \omega(f, g) p_g(g), \tag{4.14}$$

$$\omega(f, g) := 1 - \left| 1 - 2 \frac{p_f(f, g)}{p_g(g)} \right|, \tag{4.15}$$

$$p_f(f, g) := \sum_{\{v \in g | f \in Q(v)\}} p_v(v), \tag{4.16}$$

where $Q(v)$ represents the n-gram features that appear in v.

These optimizations are optional and can be disabled by defining $\Phi(G) = G$ and $\Psi(G)$ as the set of all the features that have not been selected.

To illustrate the selections made by the proposed algorithm, next we present some statistics for an English-Turkish dataset when using a vocabulary size of approximately 64,000 words and $T_g = 0.95$. The dataset contained 2,399 bigrams

Algorithm 2: Feature selection algorithm. Ψ and Φ, defined in the text, are used to prune the search space. The notation [ ] represents an empty sequence, $[f_i]$ represents a sequence of a single element f_i, and the operator $\cdot$ denotes sequence concatenation.

Data: V, a set of subword tokens;
$v \in V$;
$Q(v)$, n-gram features that appear in v;
$p_v(v)$, unigram probability of v; and
M, maximum number of n-gram features
Result: F_d, selected feature-list where $d \leq M$
Let:

$p_g(g) := \sum_{v \in g} p_v(v)$
$p_{max}(g) := \max_{v \in g} p_v(v)/p_g(g)$
$p_{mix}(f, g)$**:=**
  $(p_{max}(\{v | v \in g, f \in Q(v)\})$
  $+p_{max}(\{v | v \in g, f \notin Q(v)\}))$
`choose_feature(`G`)`**:=**
  $\arg\max_{f \in \Psi(G)} \sum_{g \in \Phi(G)} p_{mix}(f, g) p_g(g)$

$F_0 \leftarrow [\,]$
$G_0 \leftarrow \{V\}$
for $i \leftarrow 1$ **to** M **do**
  $f_i \leftarrow$ `choose_feature` (G_{i-1})
  $F_i \leftarrow F_{i-1} \cdot [f_i]$
  $G_i^{(1)} \leftarrow \{\{v \mid v \in g, f_i \in Q(v)\} \mid g \in G_{i-1}\}$
  $G_i^{(0)} \leftarrow \{\{v \mid v \in g, f_i \notin Q(v)\} \mid g \in G_{i-1}\}$
  $G_i \leftarrow \{g \in \{G_i^{(0)} \cup G_i^{(1)}\} \mid |g| > 1\}$
  if $|G_i| = 0$ **then** break
end
$d \leftarrow i$
return F_d, d

and 14,933 trigrams. The number of trigram features was too large to train a model using the proposed approach on a single GPU. The number of features selected by the feature selection algorithm was 598, out of which 28.8% were unigrams, 43.5% were bigrams, 14.5% were trigrams, 8% were four-grams, 3.8% were five-grams, and the remaining 1.4% were longer n-grams. Other datasets yielded similar statistics. Table 4.1 shows some features of more than two characters.

Table 4.1.: Some features of more than two characters when using BPE 32K.

language	**features**
English (En)	`ss$, qu@, omo, ts$, opo, ^as@, ^ce, eve, ^ves, dri` ...
Turkish (Tr)	`unu, ını, usu, isi, ini, ununu, ye@, ya@, rbi, ii$` ...
German (De)	`en$, ^...$, ^sich, zuz, ^be, eru, che, nten, ^20$ nten` ...
French (Fr)	`es$, ent, ten, és$, ^19, ^été, ^...$, let, our, ^1.` ...
Finnish (Fi)	`ta$, lal, aa$, ava, eil, aan, saa, iin$, yön$, aaraa` ...

The number of features necessary to unambiguously encode a vocabulary depends on the vocabulary size, the number of unique characters and the writing system or spelling conventions of a language. Noisy datasets contain many extraneous characters such as foreign characters or emojis. The vocabularies of languages that use a large number of characters such as Chinese and Japanese may be difficult to represent using a reduced feature set. In Section 4.6, we discuss this issue and present an approach to circumvent it.

4.4. Experiments and Analysis

To investigate the effectiveness of the proposed approach, we conducted four sets of experiments. The experiments in Subsection 4.4.2 explain the choices made by the proposed approach in comparison to other approaches. Subsection 4.4.3 demonstrates how the proposed approach relates to the chosen vocabulary size and Subsection 4.4.5 demonstrates how it relates to the training dataset size. Finally, Subsection 5.2.4 demonstrates how the proposed approach can be combined with BT.

4.4.1. Experimental Setup

We ran experiments targeting five languages: English (En), Turkish (Tr), German (De), French (Fr) and Finnish (Fi). The source language was Turkish for the models targeting English. For the rest of the models, the source language was English. The Tr-En and En-tr models were trained using the same dataset. Table 4.2 shows the number of sentences of each dataset. These datasets were sampled to make smaller training sets to simulate a low-resource setting.

The training dataset for the English-Turkish translation task comes from Tiedemann [127] and contains 205,000 parallel sentences. The German and French data were distributed as part of WMT-2014 [17, 70] and the Turkish and Finnish data as part of WMT-2017 [18, 70, 127]. For each language pair, the corresponding *newstest* was used to calculate the displayed BLEU scores [97]. The reported BLEU scores are case-sensitive and computed by *sacrebleu* [102]. It should be noted that the BLEU scores reported by *sacrebleu* are not computed in the same way as those reported by Vaswani et al. [131].

The baseline model was a transformer "*base*" [131] with eight attention heads, six layers in the encoder, and six layers in the decoder with embeddings of size 512 and 2,048 units in the hidden layers. The hidden layers for producing embeddings (FTE) contained 3,000 units, except when indicated otherwise. This implementation was based on Sockeye [61]. Each training mini batch contained approximately 4,096 target words. All models were trained on a single *GeForce GTX 1080 Ti* GPU.

In our experiments, the input and target languages shared a subword vocabulary and sub-subword features. All related weights were shared too. As a result, $E_x = E_y = W_y = E$ and $F_x = F_y = F$.

The significance testing was done using bootstrap resampling [69]. We consider improvements with P-value smaller than 0.05 to be significant.

4.4.2. Approach Comparison Experiments

In this subsection, we compare different approaches to determine the best architecture. All models were trained using a vocabulary size of approximately 32,000 subwords. All training datasets contained approximately 200,000 sentences. Ex-

Table 4.2.: Training data statistics. † This dataset is sampled to make training sets of different sizes. ‡ These datasets are sampled to make training sets of 200,000 sentences.

language pair	**sentences**
Turkish-English	205,000
English-Turkish	
English-German	4,520,620 †
English-French	40,842,333 ‡
English-Finnish	2,663,065 ‡

cluding the Turkish-English pairs, the training data were sub-sampled from larger datasets.

Table 4.3 lists the results for different models divided into three sections. Section *A* compares the feed-forward models (l-FF, where $l \in [1, 4]$) to the self-attention (*att*) approach proposed by Takase et al. [125]. The *1-FF* approach is equivalent to summing feature embeddings, similar to some of the models introduced in Chapter 2 [136, 90]. One can see that the best results in Section *A* are provided by the three-layer feed-forward models. Adding extra layers does not result in any significant gains in terms of BLEU scores.

Section *B* in Table 4.3 compares the proposed feature selection algorithm to a naive approach of selecting the features with highest frequency. Both model sets use a similar number of features, but the proposed selection algorithm results in scores comparable to those achieved when using all bigrams, whereas the naive approach yields poor results. When using the frequency-selected bigrams approach, many subwords contain non-unique representations. As an example, if the features `"po"`, `"pa"`, `"oi"` and `"ai"` were not selected, the words *point* and *paint* would have the same representation: {`"^p"`, `"in"`, `"nt"`, `"t$"`}. In such cases, the subwords are indistinguishable and the output layer will always output the one with highest unigram probability.

While the cited approaches [136, 90, 125] use n-gram features to complement standard trainable embeddings, our proposed approach avoids training subword embeddings directly. Section C reveals that for small training datasets, using fea-

Table 4.3.: Different approaches compared in terms of BLEU score results. The models starting with "*2-gram*" used all known bigrams as features, while those starting with "*selection*" use the proposed feature selection algorithm. The *l-FF* models use *l*-layered feed-forward networks to produce embeddings, while the those containing "*att*" use the approach proposed by Takase et al. [125]. For the *+comp* models, the embeddings produced from the features are used to complement the traditional embeddings through addition. It should be noted that the numbers of parameters for all models are the same after training. Best scores excluding subword regularization are shown in **bold**.

		Tr → En	**En → Tr**	**En → De**	**En → Fr**	**En → Fi**
	Baseline	16.7	13.1	18.0	26.8	12.1
A	2-gram 1-FF	16.2	13.0	17.3	25.8	12.6
	2-gram 2-FF	17.3	13.9	18.7	27.8	13.8
	2-gram 3-FF	17.6	**14.3**	**18.9**	**28.2**	14.1
	2-gram 4-FF	17.5	**14.3**	18.8	28.1	14.1
	2-gram att	17.4	13.4	18.0	27.1	13.3
B	selection 3-FF	**17.8**	14.1	**18.9**	28.0	**14.2**
	2-gram-freq 3-FF	6.2	3.5	3.7	6.8	2.4
C	2-gram 3-FF +comp	16.2	13.4	18.2	27.2	12.6
	selection 3-FF +comp	16.5	13.4	18.4	27.1	12.8
D	Regularization	18.5	15.3	19.2	27.4	15.6
	Regularization + selection 3-FF	18.7	15.6	19.3	27.3	15.7

tures to complement standard embeddings is sub-optimal. For example, consider the results for models *2-gram 3-FF* and *2-gram 3-FF +comp*, where the later complements n-gram features with subword features. The results are consistently better for the model without subword embeddings. We believe this occurs because without subword embeddings, models are forced to infer subword features from n-gram features, resulting in better generalization. In contrast, when subword embeddings are available subword features can be represented directly based on subword embeddings without using n-gram features, resulting in poor generalization and the learning of training data biases.

Section *D* in Table 4.3 explores the subword regularization technique proposed by Kudo [75]. Their approach produced good results with significant improve-

ments with respect to the baseline, in accordance with their report. The sampling hyperparameters were $\alpha = 0.2$ and $l = \infty$. Their method obtained significantly better BLEU scores than our proposed approach for Turkish-English, English-Turkish, and English-Finnish. The subword regularization model also obtained a higher BLEU score for English-German but the gain (+0.3) is not significant. For English-French, the subword regularization model obtained significantly worse results than our proposed model. The reason for this may be that both English and French are not morphologically complex. Combining our proposed approach with subword regularization improved the results non-significantly.

BERT-based Scores

The BLEU scores in Table 4.3 show that the proposed method improves baseline results when combined with the subword regularization method. However, the improvements are not significant.

We decided to compare the systems using a more robust evaluation method. Tables 4.4 and 4.5 contain the BERTScore [144] evaluation for the systems from Table 4.3. Table 4.4 also contains the BLEURT [114] scores for the systems targeting English.

Table 4.4.: Comparison of BLEURT and BERTScores for En-Tr systems. The BERTScore within parentheses is the rescaled score, not available for Turkish. BLEURT scores are only available for English. Highest BERTScore and BLEURT scores highlighted.

	Tr → En			**En → Tr**	
	BLEU	BERTScore	BLEURT	BLEU	BERTScore
Baseline	16.7	.911 (.471)	-0.246	13.1	.819
selection 3-FF	17.8	.915 (.498)	-0.194	14.1	.827
Reg.	18.5	.918 (.513)	-0.121	15.3	.832
Reg. + sel. 3-FF	18.7	**.919 (.521)**	**-0.104**	15.6	**.835**

BERTScore and BLEURT use BERT or similar models to evaluate translations considering synonyms and alternative expressions. The results show the superiority of the proposed method in all the cases tested. In the case of En→Fr using the

Table 4.5.: Comparison of BERTScores for En→De, En→Fr and En→Fi systems. The BERTScore within parentheses is the rescaled score. Highest BERTScores highlighted.

	En → De		**En → Fr**		**En → Fi**	
	BLEU	BERTScore	BLEU	BERTScore	BLEU	BERTScore
Baseline	18.0	.829 (.570)	26.8	.859 (.524)	12.1	.809 (.501)
selection 3-FF	18.9	.835 (.575)	28.0	**.864 (.637)**	14.2	.815 (.529)
Reg.	19.2	.836 (.575)	27.4	.860 (.627)	15.6	.817 (.531)
Reg. + sel. 3-FF	19.3	**.837 (.577)**	27.3	.861 (.629)	15.7	**.820 (.536)**

subword regularization method, although the proposed method achieves a lower BLEU score, BERTScore gives it a higher score. However, in the En→Fr case, the proposed model achieves the best result without using subword regularization.

4.4.3. Vocabulary Size Experiments

When using BPE subword segmentation, the vocabulary size can be regulated. The results in Table 4.6 reveal the effects of varying vocabulary sizes.

Table 4.6.: Results for different vocabulary sizes. The English-German models are trained on a large corpus and the English-Turkish models are trained on a small corpus. The scores marked with * have a statistically significant improvement with respect to the best baseline.

(full dataset)		**char**	**2K**	**4K**	**8K**	**16K**	**32K**	**64K**
en → de	baseline	19.0	25.3	25.4	26.0	26.4	26.1	**26.5**
	proposed	—	25.3	25.3	26.1	26.3	26.5	**26.5**
en → tr	baseline	6.8	13.7	13.9	**14.0**	13.6	13.3	12.4
	proposed	—	14.1	14.4*	14.4*	14.1	14.6*	**14.6***
tr → en	baseline	7.0	16.9	16.9	**17.1**	17.0	16.7	15.5
	proposed	—	17.3	17.3	17.5	17.8*	17.8*	**17.9***

The English-German NMT models are trained on approximately 4.5 million sentences and the English-Turkish models are trained on approximately 0.2 mil-

lion sentences. Having a larger vocabulary leads to less word segmenting and shorter token sequences. When the training dataset is sufficiently large, a larger vocabulary results in higher BLEU scores. However, performance drops when there is insufficient data for training the rare subwords. This effect is diminished by the proposed method because embeddings are inferred from more frequent sub-subword features.

The character-level models (*char*) yield poor BLEU scores, particularly for the low-resource settings. As shown by Cherry et al. [28], character-level NMT requires deeper models.

Table 4.7 shows some statistics for English-Turkish models of different sizes. The encoder and decoder parameters of each the model take 168.2 MiB. The size of the embedding matrix and the bias vector depends on the vocabulary size. The *model size* column shows the size of each model, baseline and proposed, after precomputing the embedding matrix and bias vector. The *with FTE* column shows the size of the proposed model without precomputing the embedding matrix and bias vector. This size depends on the number of features displayed in the column *features*. The *test time* column shows the seconds needed to translate the test set of 3,007 sentences. Larger vocabulary sizes produce shorter subword sequences, which results in faster decoding speeds. Decoding was done using beam search with a beam size of 5. The *updates/sec* column shows the times the parameters are updated per second during training. The penalty on training time for the proposed model is bigger for larger vocabulary sizes.

4.4.4. Grammatical Mistakes

Sennrich [115] demonstrated that character-level NMT performs poorly in terms of morphosyntactic agreement as the distance between tokens increases. Their evaluation data, denoted as *lingeval97*, contain sentences with categorized synthetic errors.

We evaluated our models to determine how vocabulary size affects grammatical accuracy and to elucidate the effects of the proposed method. The evaluate model is the large English-German corpus of approximately 4.5 M sentences. Table 4.8 lists accuracy metrics for different models for various error categories. Out of the 13 error categories, the proposed method performs the best on ten error

Table 4.7.: Statistics of English-Turkish models of different sizes. Model sizes are in MiB considering that each parameter takes 4 bytes. † The character-level model has a vocabulary size of 275 characters.

vocabulary	**features**	**model size**	**with FTE**	**test time**	**updates/sec**	
					baseline	**proposed**
Char †	—	169 MiB	—	445.9 s	2.23	—
BPE 2K	151	172 MiB	211 MiB	160.8 s	1.88	1.82 (97%)
BPE 4K	194	176 MiB	214 MiB	137.8 s	1.86	1.74 (94%)
BPE 8K	234	184 MiB	218 MiB	122.0 s	1.82	1.62 (89%)
BPE 16K	301	200 MiB	230 MiB	102.9 s	1.74	1.42 (82%)
BPE 32K	367	231 MiB	257 MiB	101.0 s	1.59	1.12 (70%)
BPE 64K	470	293 MiB	325 MiB	101.4 s	1.37	0.79 (58%)

categories. The absolute best results are shown in bold font and the best results among the baseline models are underlined. The results are divided into four groups.

Some errors seem to be less common with smaller vocabulary sizes, even when the BLEU score is lower. When comparing each vocabulary size independently, we observe that our proposed model is superior to the baseline model with vocabulary sizes larger than 2K subwords. The effect of including sub-subword information is larger for larger vocabulary sizes. As Table 4.6 shows, our proposed model does not significantly improve the BLEU scores for this large corpus. However, our proposed model is more resilient to the grammatical errors evaluated by *lingeval97*. The baseline and proposed models of 64 K subwords both have the same BLEU score of 26.5 but the proposed model behaves better for 10 out 13 grammatical mistake categories.

The first group of results is related to morpheme order (*Compound*) and transliteration (*Transl*). Small vocabularies work best for these error categories based on the nature of the target problem. Overall, including character n-gram information improves transliteration accuracy.

In the German language, the pronoun “Sie” can mean “they” or “she.” NMT models must choose the correct grammatical number for verbs by referring to source sentences. Including character n-gram features improves accuracy because

Table 4.8.: Grammatical accuracy for different error categories. The models correspond to the English-German results in Table 4.6. The best result out of the baseline models is underlined and the absolute best result is shown in **bold**.

	char.	**2K**		**4K**		**8K**		**16K**		**32K**		**64K**	
	bas.	**bas.**	**pro.**	**bas.**	**pro.**	**bas.**	**pro.**	**bas.**	**pro.**	**bas.**	**pro.**	**bas.**	**pro.**
Compound	.700	.874	.888	.888	.895	**.899**	.888	.892	.892	.884	.888	.892	.888
Transl	.921	.979	.984	.984	**.987**	.983	.986	.979	.981	.979	.981	.970	.979
SubjAdeq	.664	.817	.838	.829	.818	.820	.836	.819	.822	.829	**.848**	.822	.840
VerbPart	.712	.903	.896	.893	.886	.908	.910	.893	.926	.911	.929	.929	**.936**
NP	.810	.944	.944	.946	.949	.949	.949	.951	.952	.953	.954	.954	**.956**
SubjVerb	.716	.840	**.855**	.841	.846	.834	**.855**	.841	.848	.839	.846	.843	.839
Aux	.652	.902	.898	.871	.863	.888	**.912**	.889	.892	.884	.911	.879	.906
NichtDel	.536	.911	.848	.902	.930	.930	.951	.942	.949	.945	**.961**	.951	.959
KeinDel	.673	.929	.883	.879	.933	.914	**.937**	.920	.942	.898	.928	.907	.926
AffixDel	.534	.887	.875	.901	.911	.923	.932	.925	**.935**	.933	.933	.922	.927
NichtIns	.822	**.905**	.894	.894	.905	.884	.881	.851	.854	.835	.837	.827	.839
KeinIns	.786	.984	.985	.994	.994	.993	.994	.996	.996	.997	.997	**.998**	**.998**
AffixIns	.847	**.982**	.980	.981	.977	.977	.977	.970	.967	.961	.961	.959	.960
superior	—	7/13	5/13	4/13	8/13	2/13	9/13	1/13	10/13	0/13	10/13	2/13	10/13
BLEU	19.0	25.3	25.3	25.4	25.3	26.0	26.1	26.4	26.3	26.1	26.5	26.5	26.5

the third-person *singular* English verbs in the source sentences include features indicating that they end with the letter "s."

The third group of results is related to agreement. Morphosyntactic agreement between a subject and verb (*SubjVerb*), and between determiners and nouns in noun phrases (*NP*), considers number and gender. Agreement between verbs, particles (*VerbPart*), and auxiliary verbs (*Aux*) is improved by the proposed method. Excluding auxiliary verb agreement, these error categories are less common with larger vocabularies (i.e., shorter sequences).

The final group of errors changes the polarity of a sentence by either inserting or deleting polarity markers. When the negating word "nicht" or other polarity affixes are inserted (*NichtIns*, *AffixIns*), the models operating on longer sequences perform better because they are more likely to drop source sentence information.

There is no optimal vocabulary size. When considering only the BLEU scores, the larger vocabulary sizes are better. However, these improvements come as a trade-off with some error categories such as *NichtIns* and *AffixIns*.

4.4.5. Corpus Size Experiments

Table 4.9.: English-German NMT results for different dataset sizes. The benefit of the proposed method decreases as the size of the training dataset increases. The best results are shown in **bold**. It should be noted that the number of parameters for the proposed model after training is the same as that for the baseline. The scores marked with * have a statistically significant improvement with respect to their corresponding baseline.

	sentences	**baseline**	**proposed**
small-1	205,483	17.1	**19.2 (+2.1)***
small-2	410,966	19.9	**21.4 (+1.5)***
small-3	753,437	22.5	**23.1 (+0.6)***
small-4	1,506,874	24.5	**24.9 (+0.4)***
small-5	2,260,310	25.4	**25.7 (+0.3)**
full	4,520,620	**26.5**	**26.5 (=)**

Table 4.9 compares the results of different methods for different dataset sizes. We sampled random sentences from the English-German training corpus to construct training sets of increasing sizes. We then trained NMT models using these sets. The vocabulary size was approximately 64,000 subwords. We trained three models for each dataset.

One can see that the benefit of applying the proposed method decreases for larger training datasets. For the full dataset of approximately 4.5 M sentences, the proposed method does not improve the results of the baseline model.

The models trained on smaller datasets are more prone to overfitting. We believe the proposed method provides better generalization because subword features are derived from n-gram features. This regularization effect is more prominent for small datasets. For large datasets, the need for regularization is lessened and the proposed method is less salient.

4.5. Character Reduction Experiments

As a continuation of the experiments in Section 3.8, we repeat the experiments including the proposed method.

When using character reduction, the number of features needed to represent the vocabulary is reduced. For a vocabulary of 8,000 subwords, as used in the experiments, 196 features are required without character reduction and 190 with character reduction. The vocabularies are identical in both directions. Table 4.10 extends Table 3.3 which explores character reduction.

Table 4.10.: Reduction of characters with the proposed method. This table is built on the results of Table 3.3. The results follow the format "BLEU (chrF2)". Best BLEU results are shown in bold and the best chrF2 are underlined.

		En→Tr		**Tr→En**	
		Base	**Proposed**	**Base**	**Proposed**
Base	All	13.68 (.470)	15.14 (.487)	16.22 (.446)	17.90 (.465)
	Reduction	13.70 (.466)	15.38 (.489)	15.74 (.441)	17.24 (.458)
BPE dropout	All	14.17 (.479)	15.45 (.491)	17.64 (.461)	**18.07** (<u>.467</u>)
	Reduction	14.35 (.479)	**15.57** (<u>.493</u>)	17.18 (.457)	17.77 (.465)

The proposed model improves the results of its equivalent baseline in all cases.

The observations on Table 3.3 are repeated in this case. The use of character reduction improves chrF2) by 0.2 points compared to the equivalent without reduction in the En→Tr direction and decreases chrF2) by 0.2 points in the Tr→En direction for models with BPE dropout.

4.6. Ideographic Characters

A problem with the proposed method is related to the ideographic characters. While in alphabetic or phonetic writing systems most words consist of several characters, in ideographic writing systems, such as Chinese characters, many words consist of a single character. In particular, if a small BPE vocabulary is

used, a large part of the subwords are single character. The effectiveness of the proposed method may be limited by this characteristic.

The vast majority of Chinese characters are formed by combining various components. Some components can provide information about the meaning of the character. Not all components are semantic, some may be phonetic or otherwise. Exposing these components could improve representation in NMT models. For example, many characters for tree names have the 木 component, such as 楓 (maple), 桃 (peach), 柏 (oak), 松 (pine), 椿 (camellia) and many others. Many characters for insects have the 虫component, such as 蚊(mosquito), 蜂 (bee) and 蛍 (firefly). Also some characters with similar meanings can share a component: 見 (see) 視 (look), 観 (observe). In Table 3.1, we show how Unicode codepoints have an assigned a name. However, in the case of Chinese characters, they all follow the `CJK UNIFIED IDEOGRAPH-XXXX` format, where `XXXX` is an index key.

To deal with this problem, we decided to try decomposing characters into components. We propose a decomposition scheme and test it in the Subsection 4.6.1 experiments. This approach is similar to Zhang and Komachi [142] and Han et al. [59].

There are several CJKV character decomposition databases, in our experiments we use the `hfhchan/ids` dataset * Applying decomposition recurrently, all characters can be represented by 948 bases. These bases can represent 47,553 characters from various languages, including Chinese, Japanese, and Vietnamese, rare or obsolete characters, and regional variations. If we limit the characters to the Japanese data from our training data, we get 5,669 ideographic characters and 703 bases.

Table 4.11 shows some examples of CJKV character decomposition.

This sub-character information can be applied by decomposing before BPE. In this way BPE can learn the most frequent combinations and BPE dropout can be applied to character decomposition. However, such a model has the possibility of generating characters of any combination of components including nonexistent or rare characters.

Another way the decomposition data can be used in combination with the proposed model. That is, to include the n-grams of the decomposition among the

*Found at `https://github.com/hfhchan/ids`

Table 4.11.: Some examples of CJKV character decomposition.

character	decomposition
楓	⿰木⿱凡虫
桃	⿰木兆
柏	⿰木白
松	⿰木⿲八厶
蚊	⿰虫文
蜂	⿰虫⿲夂丰
蜘	⿰虫⿰矢口
蛛	⿰虫朱
見	見
視	⿰示見

features to select.

4.6.1. Experiments

Table 4.12 shows the statistics of the English-Japanese data used in the experiments. Three datasets are used in both directions, of 200,000, 400,000 and 800,000 parallel sentences. The data is sampled from The Kyoto Free Translation Task Corpus [92] for the 200,000 sentence pairs and 400,000 pairs datasets. The 800,000-pair dataset includes data from TED Talks English-Japanese corpus [23] to achieve this amount of sentences.

Table 4.12.: Statistics of the English-Japanese data used in the experiments.

Sentences	English Words	Japanese characters
200,000	4,419,913	7,701,060
400,000	8,910,971	15,361,559
800,000	12,360,397	22,410,306

Table 4.13 shows five models' CHRF2 and Character BLEU scores for English → Japanese translation. The `dec` row contains the results of decomposing the characters before training the BPE vocabulary and without using the proposed

method. The `feats` row contains the results of using the proposed method without decomposing the characters and the `dec>feats` row contains the results of using the proposed method decomposing the characters before applying BPE subword segmentation. The `feats(dec)` row contains the results of including the character decomposition as features of the proposed method but without affecting the subword segmentation.

Table 4.13.: English → Japanese low-resource translation results using sub-character features. Best character BLEU results are shown in bold and the best CHRF2 are underlined.

en→ja		**4K**		**8K**		**16K**		**32K**	
sents	**model**	**chrF**	**cBLEU**	**chrF**	**cBLEU**	**chrF**	**cBLEU**	**chrF**	**cBLEU**
200,000	base	–	–	.112	10.04	.092	7.48	.079	5.99
	feats	–	–	<u>.133</u>	**12.59**	.130	12.15	.118	10.48
	dec	.102	8.74	.104	9.06	.084	6.62	.075	5.39
	dec>feats	.119	10.90	.127	11.82	.125	11.43	.123	11.06
	feats(dec)	–	–	.114	10.26	.122	11.10	.123	10.96
400,000	base	–	–	.152	15.35	.135	12.97	.117	10.60
	feats	–	–	.160	16.18	.156	15.49	.147	14.19
	dec	.144	14.03	.138	13.51	.125	11.78	.117	10.75
	dec>feats	.145	14.07	.155	15.40	.151	14.80	.146	14.20
	feats(dec)	–	–	<u>.162</u>	**16.42**	.157	15.65	.150	14.78
800,000	base	–	–	.192	20.65	.177	18.57	.152	15.30
	feats	–	–	.176	18.56	.194	20.56	.193	20.36
	dec	.177	18.38	.173	18.01	.166	17.15	.161	16.48
	dec>feats	.167	16.99	.188	19.82	.192	20.30	.181	18.82
	feats(dec)	–	–	.197	20.71	<u>.198</u>	**21.10**	.195	20.67

For the smallest dataset, the proposed method manages to improve the baseline results without the need for sub-character features. For larger datasets, the proposed method works best when includig the character decomposition as features. The benefit of this approach is larger for the largest dataset.

Table 4.14 shows the CHRF2 and BLEU scores for Japanese→English translation.

The results show a trend in favor of not using sub-character features. In all three datasets, the models of the proposed method without sub-character features

Table 4.14.: Japanese → English low-resource translation results using sub-character features. Best BLEU results are shown in bold and the best CHRF2 are underlined.

ja→en		**4K**		**8K**		**16K**		**32K**	
sents	**model**	**chrF**	**BLEU**	**chrF**	**BLEU**	**chrF**	**BLEU**	**chrF**	**BLEU**
200,000	base	–	–	.342	5.25	.328	4.55	.311	3.77
	feats	–	–	.320	4.68	<u>.343</u>	**5.42**	.342	5.07
	dec	.312	4.09	.322	4.47	.317	4.19	.284	2.66
	dec>feats	.323	4.98	.329	4.83	.327	4.63	.322	4.65
	feats(dec)	–	–	.316	4.85	.319	4.43	.339	5.16
400,000	base	–	–	.366	6.50	.367	6.36	.349	5.23
	feats	–	–	.348	5.97	<u>.370</u>	**6.59**	.365	5.90
	dec	.347	5.72	.343	5.39	.346	5.38	.331	4.65
	dec>feats	.343	5.77	.345	5.87	.364	6.17	.346	5.67
	feats(dec)	–	–	.361	6.01	.364	6.40	.365	6.27
800,000	base	–	–	.388	8.21	.388	8.01	.387	7.37
	feats	–	–	.366	7.23	.387	8.31	<u>.399</u>	**8.37**
	dec	.382	7.70	.383	7.69	.389	7.79	.378	7.33
	dec>feats	.373	7.57	.382	7.91	.389	7.71	.394	8.11
	feats(dec)	–	–	.373	7.57	.389	8.27	.394	8.27

achieve the best results. However, the difference with the models using sub-character features (`dec>feats` and `feats(dec)`) is not significant. These results show that sub-character features are most effective when they appear in the target language. For the larger vocabulary sizes, the results of `dec>feats` and `feats(dec)` are very close. This confirms the idea that the way in which the sub-character features in the source language is not very relevant.

4.7. Embedding Analysis

Provilkov et al. [104] made a visualization of the embeddings to show that BPE dropout improves the quality of rare embeddings, preventing these from forming clusters. However, their experiments focused on non-low-resource pairs. For their visualization they used the embeddings of two models (BPE and BPE dropout) trained in 4 million English-French sentences.

To better understand the effect that the proposed method has on rare embeddings, we visualized the embeddings using t-SNE. Figure 4.3 shows the embeddings of a model trained using vanilla BPE (`BASE`), a model using BPE dropout (`BPE DROPOUT`) and a model using the proposed method with BPE dropout (`PROPOSED`). The models were trained on a 205,000 sentence pair low-resource Turkish-English corpus using a vocabulary size of 32,000 subwords. To avoid clustering the subwords based on their language we plot only subwords that appear at least once in the English data.

We found that `BPE DROPOUT` does not completely avoid clustering of rare words, although it does perform better than `BASE`. We believe that the difference with respect to the results of Provilkov et al. [104] is due to the fact that in our case the dataset is low-resource.

On the other hand, we observe that the proposed method manages to eliminate the clustering of rare words and also forms better delimited clusters based on graphic and semantic characteristics.

Many of the rare subwords are parts of proper names such as *_Mü*, *_Scepan* and *_Yap*. This explains why they are still clustered together.

To find out the nature of the clusters formed by the proposed method, we use the k-means clustering method to cluster the data into 8 clusters for each layer

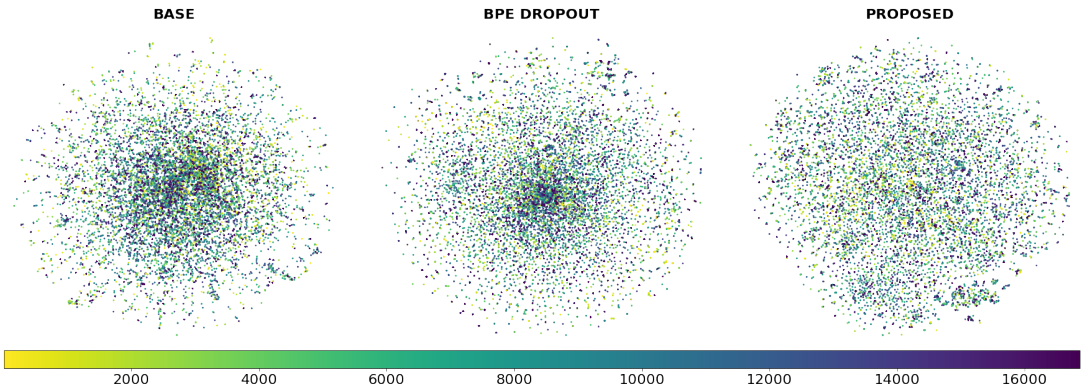

Figure 4.3.: English subword embeddings' t-SNE plot for three different models: baseline, using BPE dropout and usig BPE dropout with the proposed method. The plot shows rarer words in a darker color. The proposed methods reduces clustering of rare words.

in the FTE network.

Table 4.15 contains the most frequent English words for each of the 3×8 clusters.

The subwords shown are those with the highest frequency of each cluster. The recurring politics theme is caused by the domain of the training data.

Layer 1 represents the features selected by the featre selection algorithm. The clusters formed in layer 1 are strictly based on the n-gram features of each subword. For example, cluster $\mathcal{C}_{13}$ contains many subwords with the letter r and cluster $\mathcal{C}_{15}$ contains many subwords with the letter l.

Layer 2 is the result of applying one feed-forward layer to *Layer 1*. The clusters formed in this layer are based on both spelling and semantic features. For example, cluster $\mathcal{C}_{21}$ contains words with capital letter M and cluster $\mathcal{C}_{26}$ contains many subwords with digraph *ng*.

Layer 3 contains the embeddings of the subwords, result of applying one feed-forward layer to *Layer 2*. This is the layer used in Figure 4.3. The clusters formed in this layer are mainly semantic but do expose some spelling features. For example, many subwords in cluster $\mathcal{C}_{32}$ end in s, cluster $\mathcal{C}_{36}$ contains numbers, and cluster $\mathcal{C}_{37}$ contains punctuation marks and isolated letters.

Table 4.15.: Embedding clusters formed by the proposed method in each of the three layers. The symbol _ represents the beginning of a word.

	$\mathcal{C}_{11}$	$\mathcal{C}_{12}$	$\mathcal{C}_{13}$	$\mathcal{C}_{14}$	$\mathcal{C}_{15}$	$\mathcal{C}_{16}$	$\mathcal{C}_{17}$	$\mathcal{C}_{18}$
	_government	_the	_for	's	_also	_is	_and	_will
	_country	_of	_are	s	_all	_as	_in	_would
	_countries	_to	_from	-	_political	_has	_on	_told
layer 1	_during	_a	_their	d	_international	_was	_an	_only
(features)	_under	_that	_were	t	_last	_said	_not	_should
	_country's	_with	_more	:	_talks	_its	_new	_people
	_co-operation	_be	_after	m	_local	_this	_been	_could
	_foreign	_by	_other	ted	_officials	_his	_In	_public
	_agreement	_have	_or	ly	_military	_first	_one	_police
	_number	_–	_over	te	_national	_Times	_than	_while
	$\mathcal{C}_{21}$	$\mathcal{C}_{22}$	$\mathcal{C}_{23}$	$\mathcal{C}_{24}$	$\mathcal{C}_{25}$	$\mathcal{C}_{26}$	$\mathcal{C}_{27}$	$\mathcal{C}_{28}$
	_Minister	_political	_of	_the	_at	_during	_The	_from
	_BiH	_Prime	_to	_that	_it	_against	's	_government
	_Macedonia	_up	_and	_with	_its	ting	s	_more
layer 2	_Monday	_President	_in	_has	_but	_foreign	_In	_Times
	_Macedonian	_people	_a	_have	_two	_including	_By	_economic
	_Montenegro	_per	_for	_he	_after	_meeting	-	_some
	_May	_part	_is	_this	_told	_agreement	d	_former
	_Serbia-Montenegro	_support	_on	_which	_about	_among	t	_most
	_March	_public	_will	_their	_first	_being	:	_must
	_Parliament	_police	_as	_his	_country	_according	m	_time
	$\mathcal{C}_{31}$	$\mathcal{C}_{32}$	$\mathcal{C}_{33}$	$\mathcal{C}_{34}$	$\mathcal{C}_{35}$	$\mathcal{C}_{36}$	$\mathcal{C}_{37}$	$\mathcal{C}_{38}$
	_the	_Times	_first	_government	's	_15	_and	_EU
	_of	_countries	_while	_told	ted	_30	_–	_European
	_to	_euros	_take	_country	ly	_16	s	_Minister
layer 3	_in	_talks	_media	_international	ting	_13	_-	_Kosovo
(embeddings)	_a	_says	_key	_against	te	_18	_In	_Southeast
	_for	_years	_took	_support	se	_3	-	_Serbian
	_is	_officials	_like	_police	re	_24	d	_Serbia
	_that	_crimes	_think	_co-operation	st	_27	t	_Prime
	_on	ts	_team	_foreign	ri	_4	:	_President
	_with	_members	_election	_According	na	_26	m	_Turkish

4.8. Conclusion

We explored different approaches for utilizing character n-gram features for subword-level NMT without changing baseline model architectures or number of parameters.

We analyzed the best approach for utilizing n-gram features and determined that a nonlinear function seems to perform better than a (weighted) sum of plural feature embeddings. For low-resource language pairs, translation accuracy (BLEU score) can be improved by utilizing n-gram features. This approach also improves the grammatical accuracy of the produced translations.

We explicitly analyzed how the vocabulary size affects grammatical accuracy and found that although models using larger vocabularies tend to produce fewer grammatical errors, they are difficult to train when training data is scarce. Using n-gram features to infer embeddings can help with training.

By varying the dataset size, we determined that the proposed method works particularly well on small datasets, but its benefits decrease as the dataset size increases.

We tested the effect of the proposed sub-subword features method in combination with the character reduction method. The results show similar results to those of Chapter 3. The character reduction method allows to reduce the number of features required.

The proposed method also shows an improvement in English-Japanese results, which uses CJKV characters. We tried using CJKV character decomposition with the proposed method. The results show that, when there is enough training data, it is useful to include sub-character information from the target language, but not from the source language.

We use t-SNE to represent the effect of the proposed method on low-frequency subwords. Our method not only succeeds in eliminating low-frequency subword clusters but also better delimits clusters based on graphic and semantic characteristics. We analyzed the content of the clusters formed by the embeddings and found that the deeper layers abstract most of the spelling features while preserving the relevant ones.

Chapter 5.

External Data

We have already mentioned in this dissertation that getting parallel data can be difficult for many language pairs. In particular, pairs that do not include English tend to have little data. That is why there is research in low-resource NMT.

As we have already discussed in previous chapters, some options to improve the performance of NMT systems under low-resource conditions go through the reprocessing of data and refining architectures, hyperparameters and other machine learning techniques.

Another part of the research has focused on taking advantage of external data to make up for the lack of parallel data in target pair. One option is to use monolingual data from the source or target language, since it is easier to collect monolingual text. Another option may be to take advantage of parallel data in relation to a third language, in such a way that there are two pairs, one of them auxiliary, that have overlapping source or target language.

In this chapter, we discuss how to use the data from an auxiliary pair and how to use monolingual data to improve the performance of low-resource pairs.

We do not discuss other ideas, such as using bilingual dictionaries effectively, or data from other tasks.

5.1. Using a third language

A commonly used method of building machine translation systems for pairs with little or no parallel data is pivot translation. In the event that no parallel corpus is available from language L_1 to a language L_2 (for example, Japanese-Spanish),

but there are datasets available from L_1 to a third language L_P and from L_P to L_2, two translation systems can be built and chained to translate from L_1 to L_2 [36, 130, 137].

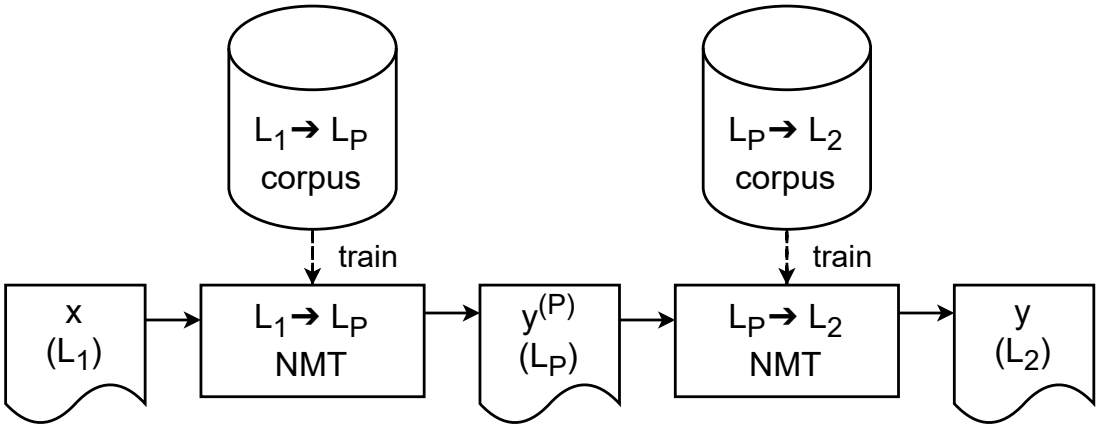

Figure 5.1.: Diagram of a cascade pivot MT. The first model translates into the pivot language L_P from L_1 and the second model translates from L_1 into L_2.

This kind of system is also known as *sequential* or *cascade* MT. In this case, there is no need for parallel data from the target pair, so the pivot translation could be considered as *zero-shot* translation. However, this term is usually reserved for systems that consist of a single model. The term *zero-resource* is preferred instead.

An obvious problem with this type of system is that the errors propagate from the first model to the second. Also, if the pivot language is not adequate, a lot of relevant information may disappear systematically. For example, the pivot language could lack a plural mark, gender distinction, or degrees of formality.

Another problem is the speed of translation, since two translations have to be decoded. To avoid this problem, for SMT models, one can use the technique known as model triangulation to create a model by combining model $L_1 \to L_P$ and $L_P \to L_2$ [130, 148, 89]. For NMT, a solution is to train the $L_1 \to L_2$ translation model using data synthesized by an $L_P \to L_2$ model from an $L_1 \to L_P$ corpus [15, 26].

To improve the performance of pivot translation, models $L_1 \to L_P$ and $L_P \to$

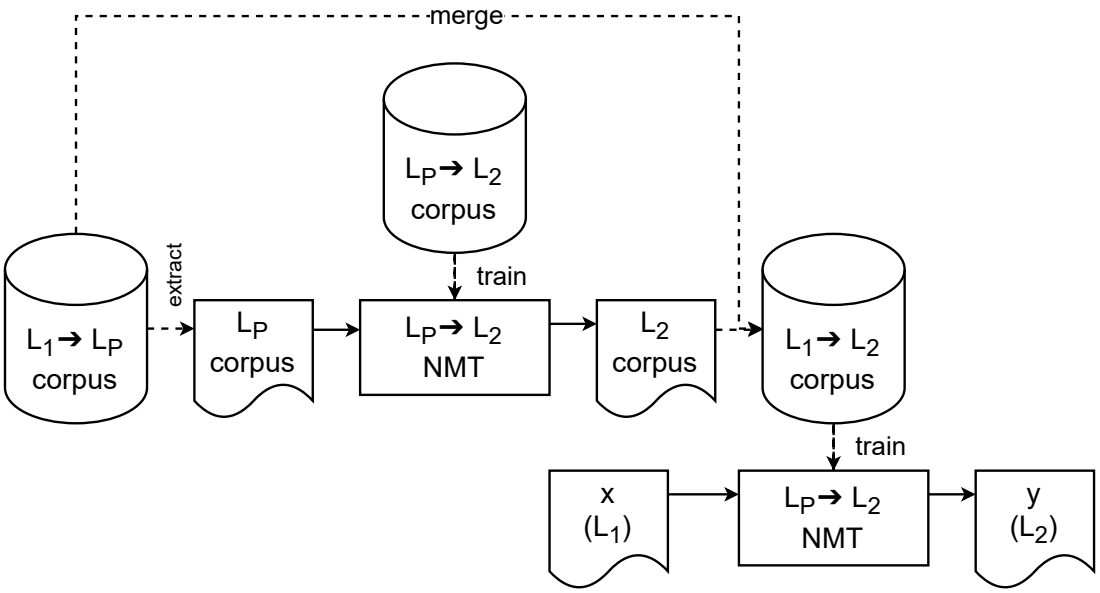

Figure 5.2.: Diagram of a synthesizing pivot MT system creation. $L_1 \rightarrow L_P$ and $L_P \rightarrow L_2$ corpora are necessary. An $L_P \rightarrow L_2$ model is used to translate the L_P sentences of a $L_1 \rightarrow L_P$ corpus to L_2 in order to synthesize a new $L_1 \rightarrow L_2$ corpus to train the final model

L_2 can be trained jointly in pursuit of co-adaptation[27].

Johnson et al. [63] introduced multilingual models to NMT. Multilingual models are capable of translating more than one pair. For this, they used a simple approach that consists of using a special symbol inserted in the source sentence indicating the target language. The architecture of the model can be the same as that of non-multilingual models. In their experiments, they showed that, although the performance of pairs with more resources worsens when sharing a model with other pairs, the performance of pairs with fewer resources improves. Multilingual models allow translation between pairs with zero resources. This is known as the *zero-shot* translation.

In Martınez and Matsumoto [83], we experimented with multilingual models using separate encoders and decoders for each language. We explored the case in which a resourceful L_A auxiliary language is used to improve a low-resource model. For this, data from an $L_1 \rightarrow L_A$ or $L_A \rightarrow L_2$ pair is used. In the case of $L_1 \rightarrow L_A$ the encoder is shared and in the case of $L_A \rightarrow L_2$ the decoder is

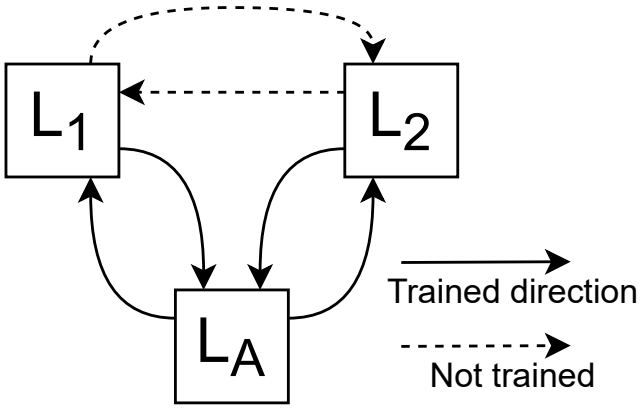

Figure 5.3.: Diagram of multilingual zero-shot MT. $L_1 \leftrightarrow L2$ directions do not have training data, but by sharing a model, it may be possible to produce translations. It is possible to have multiple auxiliary languages.

shared. The results showed the auxiliary data improved the performance of the low-resource pair when the decoder was shared, but not the encoder.

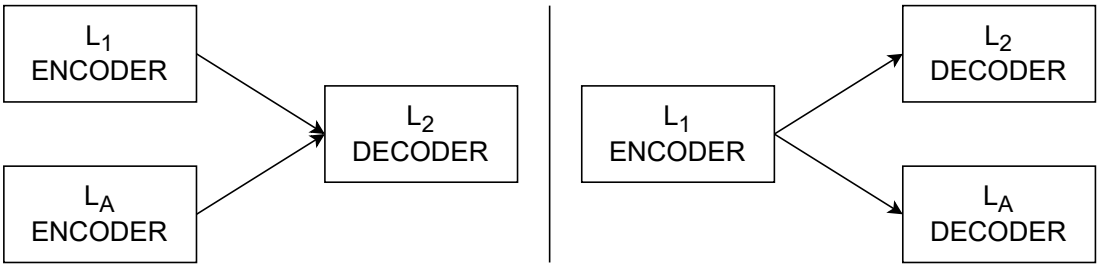

Figure 5.4.: Diagram of two multilingual models. The model on the left has a shared decoder and the auxiliary language is on the source side, and the model on the right a shared encoder with the auxiliary language on the target side. It is possible to have multilingual encoders and decoders that can process multiple languages.

Firat et al. [46] followed a similar method, with a specific encoder and decoder for each language but sharing the attention mechanism. The results show a similar trend, whereby English translation results improve consistently but translations to other languages have mixed results. This may be due to the fact that all pairs included English and therefore in the case of the English translation the shared

part was the decoder.

Zoph et al. [149] used transfer-learning to train low-resource models. They first trained the model on a pair with a large amounts of parallel data and then used that model to initialize the low-resource model, freezing some of the parameters. In their experiments, English was always the target language and the low-resource pairs improved performance, particularly when pre-trained on a related language.

Vázquez et al. [132] explored multilingual models with independent encoders and decoders for each language and a shared attention mechanism. Their experiments gave good results for the multilingual models in both zero-shot and non-zero-shot settings.

Dabre et al. [35] published a comprehensive survey that summarizes the different ideas and techniques of Multilingual Neural Machine Translation.

5.1.1. Auxiliary Language Experiments

We experimented to test the effectiveness of our proposed method in multilingual models. In particular, we tested the effect of leveraging data from the English-German pair to improve the results of the English-Turkish models.

The English-German data is two million parallel pairs drawn from *NewsCommentaryV9*[127] and *EuroparlV7*[70]. The system used to train the models is similar to that described by Johnson et al. [63]. We mark both the source statements and the target statements with a sequence specifying the origin of the data (German or Turkish corpus). For the English $\rightarrow$ Turkish models the English $\rightarrow$ German data is used and for the Turkish $\rightarrow$ English models the German $\rightarrow$ English data is used.

Since there is approximately ten times more data from English-German than from English-Turkish, only 10% of the minibatches are data from the English-Turkish pair. The baseline models were trained on two GPUs, with minibatches of approximately 4,012 tokens on the target side. With one minibatch for each GPU, each update consumed approximately 8,024 tokens of the target language. To maintain this amount of Turkish data for each update, we keep the size of the minibatches the same but we consume twenty minibatches for each update (ten on each GPU).

Table 5.1 shows the results of the experiments. We also tested the effect of

BPE dropout under these settings.

Table 5.1.: Results with an auxiliary language. Using an auxiliary language improves results only when the target language is matched. The results follow the format "BLEU (chrF2)". Best BLEU results are shown in bold and the best chrF2 are underlined.

	En-Tr		**Tr-En**	
	Base	**Proposed**	**Base**	**Proposed**
Base	13.68 (.470)	15.14 (.487)	16.22 (.446)	17.90 (.465)
+BPEd	14.17 (.479)	**15.45** (.491)	17.64 (.461)	18.07 (.467)
+AUX	12.69 (.455)	11.97 (.445)	19.25 (.478)	**19.27** (.479)
+AUX +BPEd	11.96 (.448)	11.29 (.441)	18.91 (.475)	18.45 (.474)

The results show that the auxiliary language only improves the results when it is the target language that matches, that is, the auxiliary language is on the source side. The results improve considerably with the auxiliary language but only in the Turkish → English case. Furthermore the proposed method does not make a significant improvement in this case. BPE dropout also does not improve results, but rather deteriorates them to some extent.

5.2. Monolingual data

Unlike parallel data, monolingual corpora are easier to collect, as these do not require manual annotation. That is why different ways of taking advantage of them have been explored.

5.2.1. Using Target Language Models

As we have already explained in section 2.4, the decoders of the NMT models have an architecture similar to a language model. Under low-resource conditions, the language-model of the NMT model may be under-trained. Monolingual data from the target language can be used to reinforce this LM. There are several ways to do it.

In SMT systems, neural language models, mainly recurrent neural network language models (RNNLM), have been used to re-rank translation candidates [111, 110, 112]. The system produces multiple candidates and the LM selects those that sound most natural. Since Beam Search (described in section 2.5) can produce several candidates, the same technique can be used with NMT systems [57, 58]. This approach requires training an NMT model and an LM model independently.

Gulcehre et al. [57] introduced two methods different to re-ranking that combine two independently trained NMT and LM models. The *shallow fusion* method consists of combining the probabilities produced by the NMT and LM models during the Beam Search decoding. The *deep fusion* method consists of concatenating the hidden state vectors before the output layer. This method requires a third step after the NMT model and the LM model have been trained: the fine-tuning. During fine-tuning all weights except the output layer are frozen. The hidden states of the LM model uses a gating mechanism. Sriram et al. [121] introduced the CF method whereby instead of fine-tuning the combination of the NMT and LM, the NMT is trained with a frozen LM. By training the NMT model exposed to the state of the pre-trained LM, its weights adapt and achieve better results.

In Martınez and Matsumoto [83], we trained a LM in jointly with an NMT sharing the weights of the decoder. For this we use multitask learning (MTL). Domhan and Hieber [40] used a similar approach.

5.2.2. Back Translation

Another way to take advantage of the monolingual data of the target language is back-translation (BT) [118]. Back-translation is a type of *distant supervision* in which a model of the opposite direction to the one that one wants to build is used to synthesize more parallel data. The method requires training an opposite model and the synthesized data is noisy. Still BT has been used extensively with reported good results [118, 98, 43].

BT is particularly helpful for low-resource language pairs, but it requires a big monolingual corpus of the target language. A large monolingual corpus may not exist for some low-resource languages. It also introduces new problems, such as what is the optimal amount or ratio for synthetic data or how noisy the data

can be. Poncelas et al. [98] experimented with different amounts of BT data and found that from a certain amount the increase in the BLEU score was smaller. Although they did not conclude any formula to predict the optimal amount of parallel data, they observed that a model trained only on BT data achieved a performance close to one trained with real data.

Generating millions of translations to synthesize a parallel corpus can take a long time. To reduce the cost greedy search can be used or multiple translations can be generated from single source using Beam Search. Edunov et al. [43] found that adding noise to beam search resulted in best performance.

Burlot and Yvon [22] compared different methods to take advantage of monolingual data. As a result, they found that BT gave the best results and proposed some alternative methods to use when quality BT is not available.

5.2.3. Unsupervised Machine Translation

The most radical way to take advantage of monolingual corpora is unsupervised machine translation [3, 78, 5].

Unsupervised neural machine translation (UNMT) removes the need for parallel data by training only on monolingual data.

Artetxe et al. [3] and Lample et al. [78] independently introduced UNMT as a continuation of earlier research on unsupervised cross-lingual word embeddings [2, 143, 32]. The model described by Artetxe et al. [3] combines denoising autoencoding and backtranslation, with the embeddings initialized with cross-lingual embeddings.

The word embeddings of a language can be built from monolingual data. (appointment) To build CLE in an unsupervised way, you can first build the embeddings for two languages independently. Then a small set of bases or seeds can be used to project the two sets of embeddings to the same space. Numbers are an example of a seed.

Unsupervised machine translation has also been built on SMT [79, 4]. These systems initialize the phrase table using cross-lingual embeddings and are improved using iterative back translation.

Artetxe et al. [5] created an unsupervised SMT model using cross-lingual embeddings and incorporating subword information. This SMT model is then used

to initialize an NMT model that is refined using iterative back-translation. Four models are trained: two SMT models (one in each direction), and two NMT models. As the BT iterations are repeated, the training switches to using only NMT. This method gave very good results in English-French and English-German pairs.

Although unsupervised machine translation has been proposed for low-resource conditions, research has focused on resource-rich languages, primarily English-German and English-French. The reason for this is that building robust cross-lingual embeddings in an unsupervised way requires a very large amount of quality monolingual data, and that the languages are related or use similar expressions. Kim et al. [68] did an analysis of the problem and reflected on the usefulness of unsupervised machine translation. They concluded that UNMT depends on the similarity of the source and target languages, and on the domain similarity of source and target data. They also show that a large amount of monolingual data is not enough to achieve good results and that a small dataset of about 50,000 bilingual pairs is likely to achieve better results.

5.2.4. BT Experiments

Back-translation is one of the most effective methods to improve the performance of low-resource models. We measured how the proposed approach behaves in combination with BT.

Table 5.2 shows the statistics for four language pairs. The rows are ordered from smallest to largest. The Xhosa-Zulu and English-Hausa data were published in the WMT21 news translation task. Both are classified as low-resource in the task, but Xhosa and Zulu are two closely-related languages and English and Hausa two distant languages. The English-Basque data were published for the biomedical task of WMT21. The English-Basque dataset cannot be considered low-resource by the criteria in Section 1.3 but represents two distant languages. The Basque language has a complex morphology that makes its generation difficult.

Ratios can hint about the similarity or dissimilarity of languages. Xhosa and Zulu are related languages and that is why they show a ratio close to one. English and Hausa are distant languages but their morphological characteristics result in sequences of similar length.

Table 5.3 compares the BLEU scores for five English-to-Turkish NMT models.

Table 5.2.: Statistics of the datasets used for BT experiments. The rows of the table are ordered from smallest to largest, the source language being that of the pair. The ratios are the number of words of the largest language compared to the other.

	Sentences	**Words in source**	**Words in target**	**Word ratio**
Xh-Zu	94,323	1,356,127	1,325,168	1.02
En-Tr	207,678	4,428,280	3,654,669	1.21
En-Ha	752,287	11,044,101	11,713,109	1.06
En-Eu	2,627,745	23,225,786	17,472,145	1.33

The first two rows contain the best results from Table 4.6. The three models using BT are represented by the remaining rows. These models have vocabulary sizes of approximately 32,000 words. We translated four million randomly sampled sentences from the Turkish monolingual common crawl corpus. The model *baseline-baseline* uses the baseline model with an 8,000 word vocabulary from Table 4.6 to perform BT on sentences. The models *proposed-baseline* and *proposed-proposed* use the proposed model with a vocabulary size of 64,000 words to perform BT on the monolingual sentences.

The *proposed-baseline* and *proposed-proposed* models give similar results, with *proposed-baseline* being non-significantly superior. The results suggest that there is no benefit to using the proposed method with synthetically enhanced training data.

Table 5.4 shows the results for the rest of the languages in Table 5.2. The BT data were translated using the proposed model. The BT data contain 2 million pairs of sentences.

The CHRF2 scores show better results for the proposed method in all cases. For models using BT data, the BLEU scores are better in the case of Hausa $\rightarrow$ English and English $\rightarrow$ Basque.

Except in the case of English-Basque, which is not an low-resource pair, the synthetic data improves the performance of the models. In the case of Hausa-English, the improvement in performance is especially noteworthy.

In the case of English $\rightarrow$ Basque, the use of synthetic data damages the perfor-

Table 5.3.: Results for different BT approaches for English-to-Turkish translation. Using the proposed approach to perform BT on the monolingual sentences to generate a synthetic corpus (*proposed-baseline*) yields the best performance. It should be noted that the number of parameters for the proposed model after training is the same as that for the baseline. The scores marked with * have a statistically significant improvement with respect to *baseline-baseline*.

approach	**BLEU**
best baseline (BPE 8K)	14.0
best proposed (BPE 64K)	14.6
BPE 32K baseline	13.3
BPE 32K proposed	14.6
baseline-baseline (BPE 32K)	17.3
proposed-baseline (BPE 32K)	**17.8***
proposed-proposed (BPE 32K)	17.7*

mance of the model, but the proposed method significantly improves the baseline score. Bad BT results in this case may be due to domain mismatch of parallel and monolingual data. While parallel data from both training and evaluation are primarily from the biomedical domain, the monolingual data come from various sources.

In the case of Xhosa-Zulu, the BT data improves the results, although moderately. This may be due to the high noise level in the synthetic data, due to the poor performance of the model used to generate them in the first place.

5.3. Conclusion

In this chapter we explain two ways to take advantage of external data to improve the performance of NMT models.

One way is by using parallel data from other languages. We tried to train models using parallel data from an auxiliary language. We found that, when the weights are shared, the results improve only when the auxiliary language is on

Table 5.4.: BT results for various language pairs. (+/-) BT indicates the use or non-use of BT data. The results follow the format "BLEU (CHRF2)". Best BLEU results are shown in bold and the best CHRF2 are underlined.

	-BT		**+BT**	
	Base	**Proposed**	**Base**	**Proposed**
Xh → Zu	6.52 (.416)	9.27 (.470)	9.18 (.471)	**9.71** (<u>.478</u>)
Zu → Xh	6.30 (.421)	8.52 (.468)	8.57 (.467)	**8.82** (<u>.470</u>)
En → Ha	12.02 (.412)	12.48 (.420)	17.54 (.480)	**17.98** (<u>.482</u>)
Ha → En	13.04 (.403)	14.53 (.429)	**16.69** (.460)	15.52 (<u>.461</u>)
En → Eu	16.47 (.456)	**17.34** (<u>.471</u>)	16.44 (.462)	16.35 (.463)

the source side. Combining the proposed sub-subword features method, we verify that it does not provide significant improvements.

Another way is to use monolingual data from the target language. The most common method is BT. When combining the proposed method with BT, we determined that it is helpful to use the proposed method with a BT model, but whether it provides benefits when applied to a forward translating model after incorporating a large synthetic dataset into the training data depends on the target language and the quality of the synthetic data.

Chapter 6.

Conclusion

6.1. Summary

This dissertation explored the problems related to language representation for low-resource NMT. It provided an overview of the related technologies and techniques highlighting their effect on low-resource NMT

We focused on the method we proposed in Martınez and Matsumoto [83] and explored how it relates to other techniques that have been shown effective on low-resource settings.

On Chapter 2, we explained in detail the concepts of NMT necessary to understand the subsequent chapters. Some tools frequently used in non-low-resource NMT may not be available for low-resource languages. On the other hand, these tools may not be necessary for modern NMT models. Our experiments show that for one low-resource pair, which does not have specific tokenization and normalization rules implemented, it is better to use the text without tokenizing or normalizing.

On Chapter 3, we explained in detail the concepts subword segmentation and their relation to low-resource NMT. Some innovations, such as BPE dropout, have a great effect on low-resource NMT. We compared different settings for low-resource NMT through various experiments.

Although BPE can limit the size of the vocabulary to a specific number of subwords, the minimum number of subwords is given by the number of different characters. If a dataset contains a great variety of characters these can occupy a large part of the vocabulary affecting the BPE model. We propose a method to

reduce the number of characters. Experiments show that this method, although it allows to control vocabulary, can have a negative effect in some cases.

On Chapter 4, we described our NMT training method that utilizes sub-subword n-gram features and examined its properties through multiple sets of experiments.

The proposed method also shows an improvement in English-Japanese results, which uses CJKV characters. We tried using CJKV character decomposition with the proposed method. The results show that, when there is enough training data, it is useful to include sub-character information from the target language, but not from the source language.

We use t-SNE to represent the effect of the proposed method on low-frequency subwords. Our method not only succeeds in eliminating low-frequency subword clusters but also better delimits clusters based on graphic and semantic characteristics.

On Chapter 5, we compared different methods to exploit external corpora to improve NMT. We explored using parallel data from other languages. We tried to train models using parallel data from an auxiliary language. We found that, when the weights are shared, the results improve only when the auxiliary language is on the source side. Combining the proposed sub-subword features method, we verify that it does not provide significant improvements.

Another way is to use monolingual data from the target language. The most common method is BT. When combining the proposed method with BT, we determined that it is helpful to use the proposed method with a BT model, but whether it provides benefits when applied to a forward translating model after incorporating a large synthetic dataset into the training data depends on the target language and the quality of the synthetic data.

6.2. Future Directions

In this dissertation, We have mentioned several effective methods for low-resource NMT, such as multilingual NMT, backtranslation, including bilingual dictionaries, and copy-mechanisms. Improvements in any of these techniques can contribute to improving the state-of-the-art. The effectiveness of the techniques de-

pends on several factors such as the quality of the training data, the domain and the source and target languages. The WMT results show that the best models are those that combine various techniques effectively. Thus, techniques developed for unsupervised NMT could be applied in low-resource NMT to achieve better results in the future.

The sub-subword feature method presented in this dissertation achieves good results, but there is still room for improvement. In the future, we wish to explore how the proposed low-resource techniques can be applied to tasks other than NMT, such as named entity recognition (NER). Some tasks such as dialog modeling and automatic summarization use model architectures similar to MT's. Tasks like NER and RE typically have less data than MT systems by their nature. These tasks require manual annotations that are more expensive than sentence aligning. This is why I think the proposed sub-subword features method can be effective.

Additionally, it would also be of interest to try different feature selection algorithms. The algorithm presented looks for a minimum set of features to define the vocabulary, however the selected set is one of many possibilities. It is possible to devise a method to select a minimum number of features, and to use regularization techniques to train more robust models.

The proposed method can be used to select all types of features, not just n-grams, as we have already done in Section 4.6 to include sub-character features.

References

[1] Ali Araabi and Christof Monz. Optimizing transformer for low-resource neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3429–3435, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.304. URL `https://aclanthology.org/2020.coling-main.304`.

[2] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1042. URL `https://aclanthology.org/P17-1042`.

[3] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *CoRR*, abs/1710.11041, 2017. URL `http://arxiv.org/abs/1710.11041`.

[4] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3632–3642, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1399. URL `https://aclanthology.org/D18-1399`.

[5] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. An effective approach to unsupervised machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 194–203, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1019. URL `https://aclanthology.org/P19-1019`.

[6] Philip Arthur, Graham Neubig, and Satoshi Nakamura. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1162. URL https://aclanthology.org/D16-1162.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473*, [cs.CL]:arXiv:1409.0473, September 2014.

[9] Tamali Banerjee and Pushpak Bhattacharyya. Meaningless yet meaningful: Morphology grounded subword-level NMT. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 55–60, New Orleans, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-1207. URL https://aclanthology.org/W18-1207.

[10] Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5301. URL https://www.aclweb.org/anthology/W19-5301.

[11] Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. Findings of the 2020 conference on machine translation (WMT20). In *Pro-*

ceedings of the Fifth Conference on Machine Translation, pages 1–55, Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.wmt-1.1.

[12] Christos Baziotis, Barry Haddow, and Alexandra Birch. Language model prior for low-resource neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7622–7634, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.615. URL https://www.aclweb.org/anthology/2020.emnlp-main.615.

[13] Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. On the linguistic representational power of neural machine translation models, 2019.

[14] Yoshua Bengio. *Learning Deep Architectures for AI*, volume 2. Now Publishers Inc., Hanover, MA, USA, January 2009. doi: 10.1561/2200000006. URL https://doi.org/10.1561/2200000006.

[15] Nicola Bertoldi, Madalina Barbaiani, Marcello Federico, and Roldano Cattoni. Phrase-based statistical machine translation with pivot languages. In *2008 International Workshop on Spoken Language Translation, IWSLT 2008, Honolulu, Hawaii, USA, October 20-21, 2008*, pages 143–149. ISCA, 2008. URL http://www.isca-speech.org/archive/iwslt_08/slt8_143.html.

[16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://www.aclweb.org/anthology/Q17-1010.

[17] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association

for Computational Linguistics. doi: 10.3115/v1/W14-3302. URL https://www.aclweb.org/anthology/W14-3302.

[18] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4717. URL https://www.aclweb.org/anthology/W17-4717.

[19] Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6401. URL https://www.aclweb.org/anthology/W18-6401.

[20] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *ISMIR*, 2013.

[21] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993. URL https://www.aclweb.org/anthology/J93-2003.

[22] Franck Burlot and François Yvon. Using monolingual data in neural machine translation: a systematic study. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 144–155, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6315. URL https://aclanthology.org/W18-6315.

[23] Mauro Cettolo, Christian Girardi, and Marcello Federico. WIT3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th*

Annual conference of the European Association for Machine Translation, pages 261–268, Trento, Italy, May 28–30 2012. European Association for Machine Translation. URL https://aclanthology.org/2012.eamt-1.60.

[24] Abhisek Chakrabarty, Raj Dabre, Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. Improving low-resource NMT through relevance based linguistic features incorporation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4263–4274, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.376. URL https://aclanthology.org/2020.coling-main.376.

[25] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1152. URL https://www.aclweb.org/anthology/P17-1152.

[26] Yun Chen, Yang Liu, Yong Cheng, and Victor O.K. Li. A teacher-student framework for zero-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1925–1935, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1176. URL https://aclanthology.org/P17-1176.

[27] Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. Joint training for pivot-based neural machine translation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3974–3980, 2017. doi: 10.24963/ijcai.2017/555. URL https://doi.org/10.24963/ijcai.2017/555.

[28] Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. Revisiting Character-Based Neural Machine Translation with Capacity and Compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, 2018.

[29] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

[30] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation. *arXiv:1603.06147*, [cs.CL]:arXiv:1603.06147, March 2016.

[31] Automatic Language Processing Advisory Committee. *Language and Machines: Computers in Translation and Linguistics*. The National Academies Press, Washington, DC, 1966. doi: 10.17226/9547. URL https://www.nap.edu/catalog/9547/language-and-machines-computers-in-translation-and-linguistics.

[32] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data, 2018.

[33] Anna Currey and Kenneth Heafield. Incorporating source syntax into transformer-based neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 24–33, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5203. URL https://aclanthology.org/W19-5203.

[34] Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4715. URL https://www.aclweb.org/anthology/W17-4715.

[35] Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. A survey of multilingual neural machine translation. *ACM Comput. Surv.*, 53(5), September

2020. ISSN 0360-0300. doi: 10.1145/3406095. URL https://doi.org/10.1145/3406095.

[36] A. DE GISPERT. Catalan-english statistical machine translation without parallel corpus : Bridging through spanish. *Proceedings of 5th International Conference on Language Resources and Evaluation (LREC), Genoa, Italy, 2006*, pages 65–68, 2006. URL https://ci.nii.ac.jp/naid/20001548326/en/.

[37] Hiroyuki Deguchi, Akihiro Tamura, and Takashi Ninomiya. Dependency-based self-attention for transformer NMT. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 239–246, Varna, Bulgaria, September 2019. INCOMA Ltd. doi: 10.26615/978-954-452-056-4_028. URL https://aclanthology.org/R19-1028.

[38] Michael Denkowski and Graham Neubig. Stronger Baselines for Trustable Results in Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27, Vancouver, August 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-3203.

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[40] Tobias Domhan and Felix Hieber. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505, Copenhagen, Denmark, September 2017. Association for

Computational Linguistics. doi: 10.18653/v1/D17-1158. URL https://aclanthology.org/D17-1158.

[41] Nadir Durrani, Fahim Dalvi, Hassan Sajjad, Yonatan Belinkov, and Preslav Nakov. One size does not fit all: Comparing NMT representations of different granularities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1504–1516, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1154. URL https://aclanthology.org/N19-1154.

[42] Eberhard, David M., Gary F. Simons, and Charles D. Fennig, editors. *Ethnologue: Languages of the World*. SIL International, Dallas, TX, USA, twenty-fourth edition, 2021. URL http://www.ethnologue.com.

[43] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding Back-Translation at Scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D18-1045.

[44] Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2012. URL https://aclanthology.org/P17-2012.

[45] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2090. URL https://www.aclweb.org/anthology/P17-2090.

[46] Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1026. URL `https://aclanthology.org/D16-1026`.

[47] Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2): 23–38, February 1994. ISSN 0898-9788.

[48] Mercedes García-Martínez, Ozan Caglayan, Walid Aransa, Adrien Bardet, Fethi Bougares, and Loïc Barrault. LIUM machine translation systems for WMT17 news translation task. In *Proceedings of the Second Conference on Machine Translation*, pages 288–295, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4726. URL `https://aclanthology.org/W17-4726`.

[49] Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. A convolutional encoder model for neural machine translation, 2017.

[50] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning, 2017.

[51] Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23(1):3–30, 2017.

[52] Alex Graves. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012. URL `http://arxiv.org/abs/1211.3711`.

[53] Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade, 2020.

[54] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning, 2016.

[55] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation, 2018.

[56] Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. Universal neural machine translation for extremely low resource languages. 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 344–354, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1032. URL https://www.aclweb.org/anthology/N18-1032.

[57] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation, 2015.

[58] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. On integrating a language model into neural machine translation. *Comput. Speech Lang.*, 45(C):137–148, September 2017. ISSN 0885-2308. doi: 10.1016/j.csl.2017.01.014. URL https://doi.org/10.1016/j.csl.2017.01.014.

[59] Lifeng Han, Gareth J. F. Jones, Alan F. Smeaton, and Paolo Bolzoni. Chinese character decomposition for neural mt with multi-word expressions, 2021.

[60] Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. Improved neural machine translation with smt features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 151–157. AAAI Press, 2016.

[61] Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. Sockeye: A Toolkit for Neural Machine Translation. *arXiv:1712.05690*, [cs.CL], December 2017. URL https://arxiv.org/abs/1712.05690.

[62] Liang Huang, Kai Zhao, and Mingbo Ma. When to finish? optimal beam search for neural text generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2134–2139, Copenhagen, Denmark, September 2017. Association for

Computational Linguistics. doi: 10.18653/v1/D17-1227. URL https://www.aclweb.org/anthology/D17-1227.

[63] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017. doi: 10.1162/tacl_a_00065. URL https://aclanthology.org/Q17-1024.

[64] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-2068.

[65] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009. ISBN 0131873210.

[66] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1176.

[67] Yunsu Kim, Yingbo Gao, and Hermann Ney. Effective cross-lingual transfer of neural machine translation models without shared vocabularies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1246–1257, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1120. URL https://www.aclweb.org/anthology/P19-1120.

[68] Yunsu Kim, Miguel Graça, and Hermann Ney. When and why is unsupervised neural machine translation useless? In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*,

pages 35–44, Lisboa, Portugal, November 2020. European Association for Machine Translation. URL https://aclanthology.org/2020.eamt-1.5.

[69] Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-3250.

[70] Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT. URL http://mt-archive.info/MTS-2005-Koehn.pdf.

[71] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2009. doi: 10.1017/CBO9780511815829.

[72] Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3204. URL https://www.aclweb.org/anthology/W17-3204.

[73] Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133, 2003. URL https://www.aclweb.org/anthology/N03-1017.

[74] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of*

the Demo and Poster Sessions, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/P07-2045.

[75] Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://www.aclweb.org/anthology/P18-1007.

[76] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-3230.

[77] S. KUROHASHI. Improvements of japanese morphological analyzer juman. *Proceedings of The International Workshop on Sharable Natural Language, 1994*, 1994. URL https://ci.nii.ac.jp/naid/10021991872/en/.

[78] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only, 2018.

[79] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1549. URL https://aclanthology.org/D18-1549.

[80] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *arXiv:1610.03017*, [cs.CL], 2016. URL http://arxiv.org/abs/1610.03017.

[81] Chi-kiu Lo. YiSi - a unified semantic MT quality evaluation and estimation metric for languages with different levels of available resources. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 507–513, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5358. URL https://www.aclweb.org/anthology/W19-5358.

[82] Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1002. URL https://aclanthology.org/P15-1002.

[83] Ander Martınez and Yuji Matsumoto. Improving neural machine translation on resource-limited pairs using auxiliary data of a third language. In *Proceedings of the 12th Conference of the Association for Machine Translation in the Americas*, pages 135–204, 2016.

[84] Ander Martinez, Katsuhito Sudoh, and Yuji Matsumoto. Sub-subword n-gram features for subword-level neural machine translation. *Journal of Natural Language Processing*, 28(1):82–103, 2021.

[85] Nitika Mathur, Timothy Baldwin, and Trevor Cohn. Putting evaluation in context: Contextual embeddings improve machine translation evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2799–2808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1269. URL https://www.aclweb.org/anthology/P19-1269.

[86] Nitika Mathur, Timothy Baldwin, and Trevor Cohn. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, Online, July 2020. Association

for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.448. URL https://www.aclweb.org/anthology/2020.acl-main.448.

[87] Merriam-Webster. Translation. In *Merriam-Webster.com dictionary*. Merriam-Webster.com, 2021. URL https://www.merriam-webster.com/dictionary/translation. Retrieved 11 Apr. 2021.

[88] Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. Strategies for training large scale neural network language models. *2011 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Proceedings*, 12 2011. doi: 10.1109/ASRU.2011.6163930.

[89] Akiva Miura, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Improving pivot translation by remembering the pivot. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 573–577, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2094. URL https://aclanthology.org/P15-2094.

[90] Makoto Morishita, Jun Suzuki, and Masaaki Nagata. Improving Neural Machine Translation by Incorporating Hierarchical Subword Features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 618–629, 2018.

[91] Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2297, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1276. URL https://www.aclweb.org/anthology/D15-1276.

[92] Graham Neubig. The Kyoto free translation task. http://www.phontron.com/kftt, 2011.

[93] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of*

the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 529–533, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-2093.

[94] An Nguyen Le, Ander Martinez, Akifumi Yoshimoto, and Yuji Matsumoto. Improving sequence to sequence neural machine translation by utilizing syntactic dependency information. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 21–29, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://www.aclweb.org/anthology/I17-1003.

[95] Jan Niehues and Eunah Cho. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the Second Conference on Machine Translation*, pages 80–89, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4708. URL https://aclanthology.org/W17-4708.

[96] John E. Ortega, Richard Castro Mamani, and Kyunghyun Cho. Neural machine translation with a polysynthetic low resource language. *Machine Translation*, 34(4):325–346, December 2020. ISSN 0922-6567. doi: 10.1007/s10590-020-09255-9. Funding Information: We would like to thank human annotators for Quechua to Spanish translations and translators: Roque Helmer Luna-Montoya (Academia Mayor de la Lengua Quechua in Cuzco, Peru) for translations; Julia Qquenaya-Apaza (Academia Mayor de la Lengua Quechua, Cuzco, Peru) for verifications as an annotator; Juan Omar Huanca-Balboa (Universidad Mayor de San Simón, Cochabamba, Bolivia) for annotations; Hudith Loaiza (Academia Mayor de la Lengua Quechua, Cuzco, Peru) for annotation; and, María del Rosario Saavedra (Universidad Mayor de San Simón, Cochabamba, Bolivia) for various annotation and translation tasks. We would also like to thank the support from CIFAR, eBay, Naver, NVIDIA and Google. Funding Information: We would like to thank human annotators for Quechua to Spanish translations and translators: Roque Helmer Luna-Montoya (Academia Mayor de

la Lengua Quechua in Cuzco, Peru) for translations; Julia Qquenaya-Apaza (Academia Mayor de la Lengua Quechua, Cuzco, Peru) for verifications as an annotator; Juan Omar Huanca-Balboa (Universidad Mayor de San Sim?n, Cochabamba, Bolivia) for annotations; Hudith Loaiza (Academia Mayor de la Lengua Quechua, Cuzco, Peru) for annotation; and, Mar?a del Rosario Saavedra (Universidad Mayor de San Sim?n, Cochabamba, Bolivia) for various annotation and translation tasks. We would also like to thank the support from CIFAR, eBay, Naver, NVIDIA and Google. Publisher Copyright: © 2021, The Author(s), under exclusive licence to Springer Nature B.V. part of Springer Nature.

[97] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `http://www.aclweb.org/anthology/P02-1040`.

[98] Alberto Poncelas, Dimitar Shterionov, Andy Way, Gideon Maillette de Buy Wenniger, and Peyman Passban. Investigating Backtranslation in Neural Machine Translation. *arXiv:1804.06189*, [cs.CL], April 2018. URL `http://arxiv.org/abs/1804.06189`.

[99] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. URL `https://www.aclweb.org/anthology/W15-3049`.

[100] Maja Popović. chrF deconstructed: beta parameters and n-gram weights. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 499–504, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2341. URL `https://www.aclweb.org/anthology/W16-2341`.

[101] Maja Popović. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4770. URL https://www.aclweb.org/anthology/W17-4770.

[102] Matt Post. A Call for Clarity in Reporting BLEU Scores. *arXiv:1804.08771*, [cs.CL]:arXiv:1804.08771, April 2018.

[103] Ofir Press and Lior Wolf. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E17-2025.

[104] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.170. URL https://aclanthology.org/2020.acl-main.170.

[105] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *openai.com*, 2019.

[106] Gema Ramírez-Sánchez, Jaume Zaragoza-Bernabeu, Marta Bañón, and Sergio Ortiz-Rojas. Bifixer and bicleaner: two open-source tools to clean your parallel data. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 291–298, Lisboa, Portugal, November 2020. European Association for Machine Translation. ISBN 978-989-33-0589-8.

[107] Jonne Saleva and Constantine Lignos. The effectiveness of morphology-aware segmentation in low-resource neural machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages

164–174, Online, April 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.eacl-srw.22.

[108] Víctor M. Sánchez-Cartagena, Marta Bañón, Sergio Ortiz-Rojas, and Gema Ramírez-Sánchez. Prompsit's submission to wmt 2018 parallel corpus filtering shared task. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[109] Mike Schuster and Kaisuke Nakajima. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5149–5152, March 2012. ISBN 978-1-4673-0045-2. doi: 10.1109/ICASSP.2012.6289079.

[110] Holger Schwenk. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING 2012: Posters*, pages 1071–1080, Mumbai, India, December 2012. The COLING 2012 Organizing Committee. URL https://aclanthology.org/C12-2104.

[111] Holger Schwenk, Daniel Dechelotte, and Jean-Luc Gauvain. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 723–730, Sydney, Australia, July 2006. Association for Computational Linguistics. URL https://aclanthology.org/P06-2093.

[112] Holger Schwenk, Anthony Rousseau, and Mohammed Attik. Large, pruned or continuous space language models on a GPU for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W12-2702.

[113] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks, 2017.

[114] Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. Bleurt: Learning robust metrics for text generation, 2020.

[115] Rico Sennrich. How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain, 2017. URL `http://aclweb.org/anthology/E17-2060.pdf`.

[116] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. *arXiv:1508.07909*, [cs.CL], August 2015. URL `http://arxiv.org/abs/1508.07909`.

[117] Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany, August 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W16-2323`.

[118] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. URL `https://www.aclweb.org/anthology/P16-1009`.

[119] P. Sheridan. Research in language translation on the ibm type 701. *IBM Technical Newsletter*, 1955.

[120] Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA, August 8-12 2006. Association for Machine Translation in the Americas. URL `https://www.aclweb.org/anthology/2006.amta-papers.25`.

[121] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. *CoRR*, abs/1708.06426, 2017. URL `http://arxiv.org/abs/1708.06426`.

[122] Miloš Stanojević and Khalil Sima'an. BEER 1.1: ILLC UvA submission to metrics and tuning task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 396–401, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3050. URL https://www.aclweb.org/anthology/W15-3050.

[123] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[124] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[125] Sho Takase, Jun Suzuki, and Masaaki Nagata. Character n-gram Embeddings to Improve RNN Language Models. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[126] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2020.

[127] Jörg Tiedemann. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

[128] Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. Juman++: A morphological analysis toolkit for scriptio continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59, Brussels, Belgium, November 2018.

Association for Computational Linguistics. doi: 10.18653/v1/D18-2010. URL https://www.aclweb.org/anthology/D18-2010.

[129] Daniel Torregrosa, Nivranshu Pasricha, Maraim Masoud, Bharathi Raja Chakravarthi, Juan Alonso, Noe Casas, and Mihael Arcan. Leveraging rule-based machine translation knowledge for under-resourced neural machine translation models. In *Proceedings of Machine Translation Summit XVII Volume 2: Translator, Project and User Tracks*, pages 125–133, Dublin, Ireland, August 2019. European Association for Machine Translation. URL https://aclanthology.org/W19-6725.

[130] Masao Utiyama and Hitoshi Isahara. A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491, Rochester, New York, April 2007. Association for Computational Linguistics. URL https://aclanthology.org/N07-1061.

[131] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[132] Raúl Vázquez, Alessandro Raganato, Jörg Tiedemann, and Mathias Creutz. Multilingual NMT with a language-independent attention bridge. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 33–39, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4305. URL https://aclanthology.org/W19-4305.

[133] Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline.

Technical report, Aalto University, 2013. URL http://urn.fi/URN:ISBN:978-952-60-5501-5.

[134] Changhan Wang, Kyunghyun Cho, and Jiatao Gu. Neural machine translation with byte-level subwords, 2019.

[135] Feng Wang, Wei Chen, Zhen Yang, Xiaowei Zhang, Shuang xu, and Bo Xu. A class-specific copy network for handling the rare word problem in neural machine translation. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2658–2664, 2017. doi: 10.1109/IJCNN.2017.7966181.

[136] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding Words and Sentences via Character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, Austin, Texas, November 2016. Association for Computational Linguistics. URL https://aclweb.org/anthology/D16-1157.

[137] Hua Wu and Haifeng Wang. Pivot language approach for phrase-based statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 856–863, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/P07-1108.

[138] Yingting Wu and Hai Zhao. Finding Better Subword Segmentation for Neural Machine Translation. In Maosong Sun, Ting Liu, Xiaojie Wang, Zhiyuan Liu, and Yang Liu, editors, *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, Lecture Notes in Computer Science, pages 53–64, Cham, 2018. Springer International Publishing. ISBN 9783030017163. doi: 10.1007/978-3-030-01716-3_5.

[139] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto

Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[140] Yilin Yang, Liang Huang, and Mingbo Ma. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1342. URL `https://www.aclweb.org/anthology/D18-1342`.

[141] Jiajun Zhang and Chengqing Zong. Bridging neural machine translation and bilingual dictionaries, 2016.

[142] Longtu Zhang and Mamoru Komachi. Neural machine translation of logographic languages using sub-character level information, 2018.

[143] Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1179. URL `https://aclanthology.org/P17-1179`.

[144] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

[145] Zhuosheng Zhang, Hai Zhao, Kangwei Ling, Jiangtong Li, Zuchao Li, Shexia He, and Guohong Fu. Effective Subword Segmentation for Text Comprehension. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(11):1664–1674, November 2019. ISSN 2329-9304. doi: 10.1109/TASLP.2019.2922537.

[146] Yang Zhao, Jiajun Zhang, Zhongjun He, Chengqing Zong, and Hua Wu. Addressing troublesome words in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 391–400, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1036. URL https://www.aclweb.org/anthology/D18-1036.

[147] Xing Jie Zhong and David Chiang. Look it up: Bilingual and monolingual dictionaries improve neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 538–549, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.65.

[148] Xiaoning Zhu, Zhongjun He, Hua Wu, Conghui Zhu, Haifeng Wang, and Tiejun Zhao. Improving pivot-based statistical machine translation by pivoting the co-occurrence count of phrase pairs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1665–1675, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1174. URL https://aclanthology.org/D14-1174.

[149] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1163. URL https://www.aclweb.org/anthology/D16-1163.

Publication List

Journal Papers

[1] Martinez, Ander and Sudoh, Katsuhito and Matsumoto, Yuji. Sub-Subword N-Gram Features for Subword-Level Neural Machine Translation. In *Journal of Natural Language Processing*, volume 28, pages 82–103, 2021. The Association for Natural Language Processing.

Conference Papers

[2] Martınez, Ander and Matsumoto, Yuji. Improving neural machine translation on resource-limited pairs using auxiliary data of a third language. In *Proceedings of the 12th Conference of the Association for Machine Translation in the Americas*, pages 135–204, 2016. Association for Machine Translation in the Americas.

Other Publications

[3] Le, An Nguyen and Martinez, Ander and Yoshimoto, Akifumi and Matsumoto, Yuji. Improving sequence to sequence neural machine translation by utilizing syntactic dependency information. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 21–29, 2017. Asian Federation of Natural Language Processing.

[4] Wu, Xianchao and Martínez, Ander and Klyen, Momo. Dialog Generation Using Multi-Turn Reasoning Neural Networks. 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 2049–2059, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Acknowledgements

When I decided to do my PhD in Japan, little did I know that it was going to take so long. I came with the double objective of doing a PhD at a good laboratory and becoming fluent in Japanese. Two pretty big challenges.

Since I was a child I have always been interested in languages, and I enjoy coding. I had also studied Japanese for several years, and I thought that what I needed was language immersion and practice. That is why I arrived full of confidence to meet my goals in three years.

However, these two challenges were joined by several others that I did not consider initially. During the PhD process I had to adapt to a new culture, I got married, ventured into various job opportunities, and I experienced a global pandemic.

The field in which I decided to do my PhD, neural machine translation has undergone precipitous changes in recent years, making it difficult to keep up with the state-of-the-art and publish on such a competitive topic. I want to thank all the brilliant people who have helped me stay current and motivated to continue my research.

Thanks to Professor Matsumoto and Professor Sudoh for supporting me in my research. Thanks to Professor Watanabe for his help in writing this thesis and to Professor Nakamura and Professor Shindo for the insightful comments. Thanks to all NAIST professors and colleagues for the discussions and benkyōkais. Thanks to Fujitsu colleagues. Thanks to Kitagawa-san and the friendly NAIST staff for the dedicated help.

Thanks to my wife Yui for her constant support; and apologies, since this PhD has stolen so much of my time from her. And to my parents, brother and relatives in the Basque Country, whom I have not been able to see as much as I would have liked.

Appendix A.

Example of feature selection

This appendix contains the output of the proposed feature selection algorithm executed on an English vocabulary of about 32,000 subwords.

The output is shown in several tables with a maximum of 30 entries in each one. Each row represents a step in the execution of the algorithm. Column *feature* shows the feature selected in each step step. The columns *data coverage* and *vocabulary coverage* represent the subwords that can be represented unambiguously using the selected feature set. The column *data coverage* represents the portion of subwords in the training data that has been covered, and the column *vocabulary coverage* represents the covered subwords in the vocabulary.

The columns *example 1* and *example 2* show up to two examples of new subwords covered, if any. The new words can be defined by the absence of the chosen feature and so it may not contain it.

The column *active partitions* shows the number of ambiguous subword *partitions*, as defined in Section 4.3.2. We observe two phases in the algorithm. First the number of active partitions increases, and then decreases.

The characters `\u240f` and `\u240e` represent the beginning-of-word and end-of-word symbols respectively.

Table A.1.: Example of the output of the proposed feature selection algorithm. Features 1 - 30.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
1	e	2	0.0825	0.0000	–	–
2	\u240ft	4	0.1224	0.0000	–	–
3	a	8	0.1623	0.0000	–	–
4	i	16	0.1989	0.0000	–	–
5	n	32	0.2310	0.0000	–	–
6	r	64	0.2613	0.0000	–	–
7	t	96	0.2932	0.0000	–	–
8	s	187	0.3285	0.0001	tioners	trinski
9	o	352	0.3699	0.0003	termination	tracks
10	h	599	0.4091	0.0017	thirds	technologi@@
11	d	918	0.4464	0.0046	discour@@	withstand
12	c	1395	0.4821	0.0100	hydroelectric	Undersecretary
13	l	1958	0.5204	0.0193	ticul@@	English-language
14	m	2481	0.5559	0.0326	h	four-month
15	u	3018	0.5872	0.0531	Tournam@@	unofficial
16	,	3615	0.6166	0.0820	donors,	Srebrenica,
17	.	3945	0.6440	0.1144	southeast.	anymore.
18	p	4277	0.6710	0.1474	pillar	ballots
19	y	4353	0.6957	0.1756	quickly	securing
20	@	4804	0.7201	0.2222	Shkodra	election@@
21	w	4885	0.7419	0.2380	everywhere	where@@
22	g	4973	0.7630	0.2762	wage	England
23	f	4977	0.7797	0.3042	underworld	influential
24	s\u240e	4923	0.7954	0.3294	issue	shareholder
25	b	4916	0.8104	0.3569	Built	oyed
26	v	4898	0.8219	0.3868	erable	Stankovic,
27	\"	4684	0.8325	0.4150	this,	huge
28	k	4623	0.8416	0.4380	Osijek,	spokesperson
29	\u240fa	4463	0.8501	0.4658	Karic,	academy
30	\u240fe	4341	0.8577	0.4887	emocr@@	eg@@

Table A.2.: Example of the output of the proposed feature selection algorithm. Features 31 - 60.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
31	n\u240e	4292	0.8641	0.4978	Queen	donor
32	T	4263	0.8703	0.5063	Tat@@	Tep@@
33	-	4160	0.8763	0.5194	-euro	power@@
34	\u240fr	3994	0.8821	0.5402	Israel@@	rang@@
35	\u240fi	3895	0.8874	0.5568	itarian	igh
36	S	3807	0.8923	0.5724	Sol@@	Sokol@@
37	A	3764	0.8970	0.5819	sk@@	Assad
38	e\u240e	3610	0.9010	0.5963	\"New	\"we
39	C	3544	0.9046	0.6079	Conc@@	Concerning
40	\u240fo	3448	0.9077	0.6194	Pacolli	Bode
41	(	3443	0.9108	0.6226	left,	(EurActiv,
42	E	3424	0.9139	0.6286	Mrk@@	change
43	’	3314	0.9168	0.6389	isn’t	om’s
44	M	3268	0.9196	0.6488	Party	members
45	P	3216	0.9223	0.6595	Perform@@	perhaps
46	B	3183	0.9249	0.6691	Bist@@	MB@@
47	er	3061	0.9274	0.6800	porter	tren@@
48	N	3049	0.9298	0.6862	News	Neret@@
49	j	2983	0.9322	0.6957	Ognj@@	Miha@@
50	I	2959	0.9345	0.7024	(ISAF)	Ig@@
51	D	2917	0.9367	0.7109	ND	visas
52	)	2888	0.9381	0.7156	September),	September)
53	R	2847	0.9402	0.7236	quasi-@@	Re@@
54	\u240fs	2722	0.9422	0.7350	staff,	kers
55	2	2737	0.9432	0.7360	3rd.	21st.
56	ed	2645	0.9450	0.7427	directing	end,
57	H	2611	0.9467	0.7490	(HN@@	zer
58	1	2643	0.9481	0.7494	–	12th),
59	5	2662	0.9496	0.7510	15th,	25th.
60	0	2692	0.9516	0.7528	125	50th

Table A.3.: Example of the output of the proposed feature selection algorithm. Features 61 - 90.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
61	4	2720	0.9534	0.7548	14th),	24th),
62	6	2749	0.9552	0.7567	6th).	(26
63	9	2777	0.9570	0.7592	2009,	/12/09)
64	3	2816	0.9589	0.7615	1st,	31st)
65	7	2837	0.9608	0.7643	8th).	8th),
66	8	2855	0.9623	0.7668	ht	7/07)
67	ll	2756	0.9638	0.7750	cell@@	ell@@
68	F	2718	0.9653	0.7811	narco@@	For@@
69	es	2605	0.9667	0.7890	Weeks	machines
70	G	2587	0.9681	0.7933	G,	zen
71	K	2545	0.9694	0.7997	Kac@@	uck@@
72	\"\u240e	2477	0.9707	0.8041	y.\"	ly\".
73	L	2444	0.9719	0.8094	gauge	jub@@
74	z	2375	0.9731	0.8169	[a	Krye@@
75	/	2394	0.9741	0.8189	HIV/@@	/SETimes]
76	\u240fl	2334	0.9751	0.8239	log@@	location
77	O	2294	0.9760	0.8288	NGO@@	O),
78	00	2284	0.9769	0.8307	2006	700@@
79	%	2280	0.9778	0.8335	6,	6.
80	r@	2177	0.9786	0.8409	bor@@	art@@
81	ee	2129	0.9793	0.8443	ber	green@@
82	x	2072	0.9801	0.8489	next,	x@@
83	\u240f1	2059	0.9808	0.8513	1/08)	$1
84	V	2033	0.9814	0.8547	Ved@@	VO@@
85	in	1976	0.9820	0.8587	Toni	ations.
86	t\u240e	1940	0.9826	0.8613	heart	Packett:
87	;	1923	0.9831	0.8648	5/11	);
88	s,	1851	0.9837	0.8694	professors,	reasons,
89	oo	1801	0.9842	0.8730	dog	choos@@
90	11	1787	0.9848	0.8750	/11/05)	11m

Table A.4.: Example of the output of the proposed feature selection algorithm. Features 91 - 120.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
91	[	1752	0.9853	0.8777	Fil@@	AFP
92	:	1710	0.9858	0.8813	ovic:	s:
93	U	1680	0.9863	0.8839	Up@@	(D@@
94	J	1639	0.9868	0.8875	going	Jim
95	ss	1579	0.9872	0.8916	ss@@	les@@
96	i@	1491	0.9877	0.8975	Krasniq@@	pi@@
97	s.\u240e	1437	0.9881	0.9009	arrests.	ships.
98	W	1411	0.9885	0.9030	here	Web@@
99	ni	1373	0.9889	0.9057	Bosniak	inated
100	de	1351	0.9893	0.9072	needed,	fined
101	]	1330	0.9896	0.9091	es?	t]
102	\u240f3	1287	0.9900	0.9119	37@@	43
103	o\u240e	1271	0.9903	0.9129	who	pol
104	Z	1249	0.9907	0.9149	AZ@@	fa@@
105	ff	1235	0.9910	0.9158	of.	ff@@
106	se	1197	0.9913	0.9182	addresses	witness.
107	ac	1164	0.9916	0.9204	sacred	financing
108	al	1131	0.9919	0.9226	ailing	fla@@
109	\u240fu	1105	0.9922	0.9246	unconstitutional	uh
110	ro	1083	0.9924	0.9259	corporate	board@@
111	0,	1083	0.9927	0.9265	4,000	150,000
112	0\u240e	1072	0.9929	0.9275	2002	/11/10
113	\u240f5	1038	0.9932	0.9297	5/06)	6/05;
114	22	1026	0.9934	0.9306	22nd)	220
115	n@	983	0.9936	0.9335	conferenc@@	Constan@@
116	nn	953	0.9938	0.9355	question@@	tunn@@
117	a\u240e	919	0.9940	0.9377	mila	lava
118	\u240f2	903	0.9942	0.9389	6.2	20m
119	st	882	0.9944	0.9404	situ@@	stak@@
120	l@	845	0.9946	0.9427	Kle@@	Ble@@

Table A.5.: Example of the output of the proposed feature selection algorithm. Features 121 - 150.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
121	gi	833	0.9947	0.9436	staging	bringing
122	Y	823	0.9949	0.9443	FIC@@	Yug@@
123	\u240fS	812	0.9950	0.9452	SEEC@@	AS
124	\u240f6	791	0.9952	0.9466	/10/06-@@	7/06
125	000	774	0.9953	0.9476	,00@@	12,000
126	pe	759	0.9955	0.9486	scep@@	preparation
127	\u240fd	740	0.9956	0.9498	mad@@	model@@
128	tt	715	0.9957	0.9514	ety	itting
129	re	693	0.9959	0.9528	firefighters	fresco@@
130	q	678	0.9960	0.9539	Iraq@@	Sh@@
131	00,	668	0.9961	0.9545	200,000	40,000
132	te	646	0.9962	0.9559	ites	investigate
133	\u240f7	626	0.9963	0.9572	7/10-@@	4/07
134	i\u240e	609	0.9965	0.9583	Dacic	ari
135	99	604	0.9966	0.9587	1989,	99.@@
136	dd	591	0.9967	0.9596	adding	med@@
137	p@	571	0.9967	0.9609	princip@@	prop@@
138	\u240fE	562	0.9968	0.9615	MESS	FE@@
139	\u240f4	550	0.9969	0.9623	/10/04;	9/04;
140	on	538	0.9970	0.9630	autonomous	counted
141	?	525	0.9971	0.9639	ing	er?
142	ar	506	0.9972	0.9650	guard@@	marg@@
143	\u240fI	495	0.9972	0.9658	DI@@	BI@@
144	rr	477	0.9973	0.9670	arre@@	arrest
145	e@	457	0.9974	0.9683	rev@@	eye@@
146	a,	448	0.9975	0.9688	at,	ean,
147	EE	446	0.9975	0.9689	SEE	SE@@
148	\u240fA	437	0.9976	0.9695	NA@@	ANA,
149	\u240f1999	433	0.9977	0.9698	1999.	1991.
150	.\u240e	421	0.9977	0.9706	5.	1.8

Table A.6.: Example of the output of the proposed feature selection algorithm. Features 151 - 180.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
151	to	416	0.9978	0.9709	protests	tough
152	10	412	0.9978	0.9712	2010.	2012.
153	DP	406	0.9979	0.9716	DPS	(DP@@
154	u@	389	0.9980	0.9728	ouf@@	statu@@
155	mm	380	0.9980	0.9734	am@@	im@@
156	\u240f9	370	0.9981	0.9740	9/08)	8/09-@@
157	X	365	0.9981	0.9744	EX@@	Xhel@@
158	mo	358	0.9982	0.9748	bomb@@	Smo@@
159	a@	346	0.9982	0.9756	Strat@@	agend@@
160	pp	338	0.9983	0.9762	ped	supp@@
161	”	336	0.9983	0.9763	’,	”@@
162	ri	328	0.9984	0.9768	parliamentarians	Adrian
163	SS	324	0.9984	0.9771	DS,	USS
164	\u2019	315	0.9985	0.9777	as	days
165	it	305	0.9985	0.9783	enti@@	title
166	\u240fBB	303	0.9985	0.9785	BBC,	B@@
167	ul	300	0.9986	0.9786	flu	flu.
168	o@	290	0.9986	0.9793	Kosov@@	mom@@
169	am	282	0.9986	0.9798	Ramadanovic	liam@@
170	ea	277	0.9987	0.9801	delay	are.
171	is	270	0.9987	0.9805	idis	i/SETimes]
172	55	264	0.9987	0.9809	550@@	55th
173	\u240fII	262	0.9988	0.9811	I,	II,
174	\u240fc	254	0.9988	0.9816	civic	becom@@
175	1\u240e	251	0.9988	0.9818	121	12
176	gg	245	0.9989	0.9822	big@@	Strugg@@
177	08	241	0.9989	0.9824	/10/08)	80@@
178	id	236	0.9989	0.9828	did,	di,
179	Q@	235	0.9989	0.9829	Q@@	H@@
180	or	229	0.9990	0.9833	torturing	orus

Table A.7.: Example of the output of the proposed feature selection algorithm. Features 181 - 210.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
181	D@	224	0.9990	0.9836	DVD@@	(UD@@
182	\u240f-	219	0.9990	0.9840	by-@@	-by-@@
183	N@	211	0.9991	0.9845	(BN@@	MN@@
184	(S	205	0.9991	0.9848	(RS),	(SRS),
185	ns\u240e	200	0.9991	0.9851	convictions	championships
186	cc	197	0.9991	0.9853	ac@@	oc@@
187	i,	193	0.9992	0.9856	it,	ik,
188	\u240f-@	192	0.9992	0.9857	–@@	-@@
189	\u0107	189	0.9992	0.9860	Kaletovi\u0107	\u0107,
190	\u201c	188	0.9992	0.9861	\u201c@@	\u201cThe
191	AA	184	0.9992	0.9863	(AAK@@	SAA@@
192	\u240fO	181	0.9993	0.9865	MO@@	OT@@
193	na	175	0.9993	0.9869	an.	Canad@@
194	$	175	0.9993	0.9869	$@@	–
195	\u00fc	172	0.9993	0.9871	\u00fcr@@	Tur@@
196	rs	170	0.9993	0.9872	starts	stars
197	..	169	0.9993	0.9873	...	+.
198	44	165	0.9994	0.9876	1244@@	4%
199	he	160	0.9994	0.9879	threat.	arche@@
200	la	155	0.9994	0.9882	gambling	Milanovic
201	33	151	0.9994	0.9885	3@@	33,
202	s@	146	0.9994	0.9888	this@@	mechanis@@
203	S\u240e	143	0.9994	0.9890	SRS	SP
204	ci	137	0.9995	0.9894	icial	recei@@
205	02	133	0.9995	0.9896	2020,	20
206	\u240fP	130	0.9995	0.9898	KP@@	PC@@
207	ck	127	0.9995	0.9900	kic@@	kick@@
208	ev	121	0.9995	0.9904	ive	sever@@
209	n-	117	0.9995	0.9906	month-@@	Macedonian-@@
210	un	112	0.9996	0.9910	unding	trun@@

Table A.8.: Example of the output of the proposed feature selection algorithm. Features 211 - 240.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
211	\u0131@	112	0.9996	0.9910	\u0131@@	–
212	&@	111	0.9996	0.9911	S&@@	R@@
213	le	108	0.9996	0.9913	Welle,	compiled
214	0/	103	0.9996	0.9916	30/0@@	/10-@@
215	\u00e7	103	0.9996	0.9916	\u00e7@@	–
216	(EC	102	0.9996	0.9917	(EC)	(CEC)
217	\u00e9@	102	0.9996	0.9918	\u00e9@@	–
218	si	99	0.9996	0.9920	crisis@@	ison
219	fi	97	0.9997	0.9921	specific@@	affair@@
220	zz@	96	0.9997	0.9922	ez@@	zz@@
221	66	94	0.9997	0.9923	6@@	66@@
222	WW	93	0.9997	0.9924	W@@	WW@@
223	\u2013@	93	0.9997	0.9924	\u2013@@	–
224	vo	91	0.9997	0.9925	ovo,	zov@@
225	0.@	90	0.9997	0.9926	0.@@	0.0@@
226	\u240fBB@	89	0.9997	0.9927	BBB@@	BB@@
227	OC	87	0.9997	0.9928	(ICO)	ICO
228	\u00f6	87	0.9997	0.9928	\u00f6@@	–
229	TR	85	0.9997	0.9930	RTS,	TR@@
230	g@	84	0.9997	0.9930	glob@@	blog@@
231	\u240f19	83	0.9998	0.9931	189@@	198@@
232	en	81	0.9998	0.9932	eness	enting
233	I@	79	0.9998	0.9933	IFI@@	FI@@
234	\u240fexe	78	0.9998	0.9934	exc@@	exec@@
235	C@	75	0.9998	0.9936	CH@@	CR@@
236	\u240fases\u240e	74	0.9998	0.9937	assess	assesses
237	now,\u240e	73	0.9998	0.9937	known,	know,
238	io	71	0.9998	0.9938	biodi@@	Momir
239	t@	70	0.9998	0.9939	text@@	tex@@
240	\u240fign	69	0.9998	0.9940	igning	ining

Table A.9.: Example of the output of the proposed feature selection algorithm. Features 241 - 270.

n	feature	active partitions	data coverage	vocabulary coverage	example 1	example 2
241	12	67	0.9998	0.9941	\u00bd@@	201@@
242	r\u240e	64	0.9998	0.9943	perform	ord
243	ad	62	0.9998	0.9944	Hadj@@	ading
244	CC	60	0.9998	0.9945	ACC@@	AC@@
245	+@	60	0.9998	0.9946	+@@	–
246	\u00eb@	60	0.9998	0.9946	\u00eb@@	–
247	\u011f	59	0.9998	0.9947	g@@	\u011f@@
248	\u240fsup	58	0.9999	0.9947	susp@@	sup@@
249	olo	56	0.9999	0.9948	ols	olos
250	ia.\u240e	55	0.9999	0.9949	Albanian.	Albania.
251	\u240fII\u240e	54	0.9999	0.9950	III	II
252	!	54	0.9999	0.9950	!@@	–
253	e.\u240e	53	0.9999	0.9951	scheduled.	schedule.
254	\u00c7	53	0.9999	0.9951	\u00c7@@	–
255	\u2026@	53	0.9999	0.9951	\u2026@@	–
256	\u240fpop	52	0.9999	0.9952	popul@@	poul@@
257	\u010d	51	0.9999	0.9953	\u010d@@	c@@
258	\u240fVA	50	0.9999	0.9953	TAV	VAT
259	Kiri@	49	0.9999	0.9954	Kiri@@	Kri@@
260	\u240fCNN	48	0.9999	0.9954	CNN@@	CN@@
261	t.	47	0.9999	0.9955	iti.	it.
262	\u0161	47	0.9999	0.9955	\u0161@@	–
263	20	46	0.9999	0.9956	2010	210
264	\u240fLa	45	0.9999	0.9957	Laj@@	Ljaj@@
265	TT	44	0.9999	0.9957	RTT@@	RT@@
266	ara	42	0.9999	0.9959	Parv@@	part@@
267	\u201d	42	0.9999	0.9959	\u201d@@	–
268	PS	41	0.9999	0.9959	(SP)	(SPS)
269	\u240fw	40	0.9999	0.9960	however	whoever
270	los@	39	0.9999	0.9961	Milosos@@	Milos@@

Table A.10.: Example of the output of the proposed feature selection algorithm. Features 271 - 300.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
271	pr	37	0.9999	0.9962	prepares	prepared
272	\u017e	36	0.9999	0.9963	z@@	\u017e@@
273	obb	35	0.9999	0.9963	lobb@@	lob@@
274	anti	34	0.9999	0.9964	anting	ating
275	\u2018	34	0.9999	0.9964	\u2018@@	–
276	im	32	0.9999	0.9965	Dimit@@	Simil@@
277	\u0130	31	0.9999	0.9966	I@@	\u0130@@
278	\u240fass@	30	0.9999	0.9967	ass@@	assass@@
279	\u0160	30	1.0000	0.9967	\u0160@@	–
280	KK@	28	1.0000	0.9968	PK@@	PKK@@
281	\u00e0	28	1.0000	0.9969	\u00e0@@	–
282	EC	27	1.0000	0.9969	CE	CEC
283	yy@	26	1.0000	0.9970	Tayy@@	Tay@@
284	\u00dc	25	1.0000	0.9970	\u00dc@@	U@@
285	\u02c8@	25	1.0000	0.9971	\u02c8@@	–
286	\u015f	24	1.0000	0.9971	s@@	\u015f@@
287	\u0431	24	1.0000	0.9972	\u0431@@	–
288	\u00d6	23	1.0000	0.9972	O@@	\u00d6@@
289	nd-@	22	1.0000	0.9973	and-a-@@	and-@@
290	*	22	1.0000	0.9973	*@@	–
291	ch	21	1.0000	0.9974	ithic	itch
292	\u00ef@	21	1.0000	0.9974	\u00ef@@	–
293	\u0433	21	1.0000	0.9975	\u0433@@	–
294	\u20ac	21	1.0000	0.9975	\u20ac@@	–
295	\u0111	21	1.0000	0.9975	\u0111@@	–
296	\u2014	21	1.0000	0.9976	\u2014@@	–
297	\\	21	1.0000	0.9976	\\@@	–
298	\u00ed	20	1.0000	0.9976	\u00ed@@	i@@
299	\u017d	19	1.0000	0.9977	Z@@	\u017d@@
300	Liber@	18	1.0000	0.9978	Lieber@@	Liber@@

Table A.11.: Example of the output of the proposed feature selection algorithm. Features 301 - 330.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
301	...@	17	1.0000	0.9978	..@@	...@@
302	\u0399	17	1.0000	0.9979	\u0399@@	–
303	\u00f3	16	1.0000	0.9979	\u00f3@@	o@@
304	\u00e2	16	1.0000	0.9980	\u00e2@@	–
305	\u0080	16	1.0000	0.9980	\u0080@@	–
306	\u00c9	15	1.0000	0.9981	E@@	\u00c9@@
307	\u240f@	15	1.0000	0.9981	@@@	–
308	_	15	1.0000	0.9981	_@@	–
309	\u00b0	15	1.0000	0.9981	\u00b0@@	–
310	=@	15	1.0000	0.9982	=@@	–
311	\u039a	15	1.0000	0.9982	\u039a@@	–
312	\u0391@	15	1.0000	0.9982	\u0391@@	–
313	\u039f	15	1.0000	0.9983	\u039f@@	–
314	\u015e	14	1.0000	0.9983	S@@	\u015e@@
315	\u0440	14	1.0000	0.9984	\u0440@@	–
316	ht	13	1.0000	0.9984	roughout	rought
317	11/	12	1.0000	0.9985	/11	/11/11
318	ww@	11	1.0000	0.9986	www@@	w@@
319	\u0093	11	1.0000	0.9986	\u0093	–
320	\u0410@	11	1.0000	0.9986	\u0410@@	–
321	\u0443	11	1.0000	0.9986	\u0443@@	–
322	<	11	1.0000	0.9987	<@@	–
323	‘	11	1.0000	0.9987	‘@@	–
324	\u0422	11	1.0000	0.9987	\u0422@@	–
325	\u00f8	11	1.0000	0.9988	\u00f8@@	–
326	\u0395	11	1.0000	0.9988	\u0395@@	–
327	#	11	1.0000	0.9988	#@@	–
328	\u043b	11	1.0000	0.9989	\u043b@@	–
329	\u00e1	11	1.0000	0.9989	\u00e1@@	–
330	\u010c	10	1.0000	0.9990	C@@	\u010c@@

Table A.12.: Example of the output of the proposed feature selection algorithm. Features 331 - 353.

n	**feature**	**active partitions**	**data coverage**	**vocabulary coverage**	**example 1**	**example 2**
331	\u044f	10	1.0000	0.9990	\u044f@@	–
332	\u00f1	9	1.0000	0.9991	\u00f1@@	n@@
333	\u00e8	8	1.0000	0.9991	\u00e8@@	e@@
334	\u0096	7	1.0000	0.9992	\u009d	\u0096
335	//	6	1.0000	0.9992	/@@	//@@
336	\u00e4	5	1.0000	0.9993	a@@	\u00e4@@
337	\u00c3	4	1.0000	0.9994	\u00c3@@	A@@
338	_m	3	1.0000	0.9994	m	\u00e6
339	dj@	2	1.0000	0.9995	Djind@@	Djindj@@
340	\u00fd@	1	1.0000	0.9996	y@@	\u00fd@@
341	\u0435	1	1.0000	0.9996	\u0435@@	–
342	\u043e	1	1.0000	0.9996	\u043e@@	–
343	\u0110	1	1.0000	0.9997	\u0110@@	–
344	\u2010	1	1.0000	0.9997	\u2011@@	–
345	>	1	1.0000	0.9997	>@@	–
346	\u00a6	1	1.0000	0.9997	\u00a6@@	–
347	\u03bc	1	1.0000	0.9998	\u03bc@@	–
348	\u041c	1	1.0000	0.9998	\u041c@@	–
349	\u00a3	1	1.0000	0.9998	\u00a3@@	–
350	{	1	1.0000	0.9999	{ @@	–
351	}	1	1.0000	0.9999	} @@	–
352	\u044a	1	1.0000	0.9999	\u044a@@	–
353	\u0430	0	1.0000	1.0000	\u0430@@	\u0438@@