Universität Zürich[UZH]

Master's Thesis
for obtaining the academic degree
Master of Arts
in Faculty of Arts and Social Sciences

Machine Translation of Complex Sentences from Latin to German

Author: Sabrina Brändle
Student ID number: 14-705-164

Advisor: M. Sc. Lukas Fischer

Supervisor: Prof. Dr. Martin Volk

Department of Computational Linguistics

Submission date: 01.12.2022

Abstract

Neural Machine Translation (NMT) yields remarkable results for high-resource languages trained on large amounts of data. When there is a lack of linguistic data, the translation quality decreases [Hedderich et al., 2020]. In such data-sparse low-resource scenarios, the translation quality of long sentences is particularly affected [Kondo et al., 2021; Fischer et al., 2022]. In NMT, the translation from Latin to German combines both of these problems since Latin-German is a low-resource language pair and Latin is a low-resource language in the context of NMT which often features long sentences [Garcia and Tejedor, 2020]. Many existing Latin texts are not closely translated or digitally available. However, the Bullinger Digital Project[1] aims to make the Latin texts from Heinrich Bullinger's correspondence and their translations into German digitally available in a database [Fischer et al., 2022].

With Data Augmentation, the amount of training data can be increased for NMT systems that are applied on low-resource language pairs. I propose a Data Augmentation method which consists of splitting long sentences into shorter sequences and translating them into German separately. I compare the translation quality of original long sentences to the quality of the translations of shorter sequences. Reasons for the performance drop in long sentence translations differ depending on the respective NMT architecture [Neishi and Yoshinaga, 2019; Tien and Minh, 2019]. However, literature shows that augmenting data is generally helpful for NMT systems. The method I propose has already been applied similarly to other language pairs in NMT. The results show that this approach, combined with a postprocessing step, increases the translation quality from 21.16 to 21.96 BLEU. In further research, this segmentation method can help achieve and improve state-of-the-art NMT performance in Latin NMT and in low-resource scenarios in general, offering various benefits such as making linguistic knowledge accessible to speakers of different languages [Magueresse et al., 2020].

[1] https://www.bullinger-digital.ch/about

Zusammenfassung

Die neuronale maschinelle Übersetzung (NMT) liefert bemerkenswerte Ergebnisse für Sprachen mit vielen Ressourcen, die auf grossen Datenmengen trainiert wurden. Wenn es an linguistischen Daten mangelt, sinkt die Übersetzungsqualität [Hedderich et al., 2020]. In solchen datenarmen Low-Resource-Szenarien ist die Übersetzungsqualität von langen Sätzen besonders betroffen [Kondo et al., 2021; Fischer et al., 2022]. In der NMT vereint die Übersetzung vom Lateinischen ins Deutsche diese beiden Herausforderungen, da Latein-Deutsch ein ressourcenarmes Sprachpaar darstellt und Latein im NMT-Kontext eine ressourcenarme Sprache ist, die oft lange Sätze enthält [Garcia and Tejedor, 2020]. Viele vorhandene lateinische Texte sind nicht nahe am Text übersetzt oder digital verfügbar. Das Bullinger Digital Projekt[2] zielt darauf ab, die lateinischen Texte der Korrespondenz Heinrich Bullingers und ihre Übersetzungen ins Deutsche in einer Datenbank [Fischer et al., 2022] digital verfügbar zu machen.

Mit Data Augmentation kann die Menge der Trainingsdaten für NMT-Systeme, die auf ressourcenarme Sprachpaare angewendet werden, erhöht werden. Ich schlage eine Methode zur Datenerweiterung vor, die darin besteht, lange Sätze in kürzere Sequenzen aufzuteilen und diese separat ins Deutsche zu übersetzen. Ich vergleiche die Übersetzungsqualität der langen Originalsätze mit der Übersetzungsqualität der kürzeren Sequenzen. Die Ursache für den Leistungsabfall bei der Übersetzung langer Sätze unterscheidet sich je nach der jeweiligen NMT-Architektur [Neishi and Yoshinaga, 2019; Tien and Minh, 2019]. Die Literatur zeigt jedoch, dass eine Anreicherung der Daten für NMT-Systeme generell hilfreich ist. Die von mir vorgeschlagene Methode wurde bereits in ähnlicher Weise auf andere Sprachpaare in der NMT angewendet. Die Ergebnisse zeigen, dass dieser Ansatz, kombiniert mit einem Nachbearbeitungsschritt, die Übersetzungsqualität von 21,16 auf 21,96 BLEU erhöht. Mit weiteren Forschungsarbeiten kann die Segmentierungsmethode dazu beitragen, NMT-Leistung des heutigen Standes in der NMT mit Latein und in ressourcenarmen Szenarien im Allgemeinen zu erreichen und zu verbessern, was verschiedene Vorteile bietet, wie beispielsweise linguistisches Wissen für Sprecher verschiedener Sprachen zugänglich zu machen.

[2]https://www.bullinger-digital.ch/about

Acknowledgement

I would like to take this opportunity to thank everyone who supported me in any way with my master's thesis:

My special thanks go to Prof. Dr. Martin Volk, who has motivated and supported me during this work, and who has taught and inspired me throughout my studies in Digital Linguistics.

For his frequent and dedicated support on a technical and organizational level, I would like to sincerely thank Lukas Fischer, with whom I very much appreciated discussing and finding solutions for the challenges at hand.

I also want to thank Raphael Schwitter for his competent support in linguistic questions and his helpful contribution to the gold standard of the POS-tagged data.

Furthermore, I thank my family and friends for their moral support and for proof-reading my master's thesis.

Contents

List of Figures

List of Tables

List of Acronyms

CLTK	Classical Language Toolkit
CNN	Convolutional Neural Network
DE	German
LA	Latin
LRL	Low-resource Language
MT	Machine Translation
NER	Named Entity Recognition
NLP	Natural Language Processing
NMT	Neural Machine Translation
OCR	Optical Character Recognition
POS	Part-Of-Speech
RNN	Recurrent Neural Network
SMT	Statistical Machine Translation
TEI	Text Encoding Initiative
UD	Universal Dependencies
UTF-8	Unicode Transformation Format (8-bit)
XML	eXtensible Markup Language

1 Introduction

1.1 Motivation

While Neural Machine Translation (NMT) yields remarkable results for high-resource languages trained on large amounts of data, NMT systems for low-resource languages do still not achieve comparable quality. This widespread attention NMT has gained recently is mainly due to the availability of large training corpora [Kondo et al., 2021]. For many low-resource languages, however, Natural Language Processing (NLP) tools are not available or developed because these languages mostly lack labelled and unlabelled data, which in the context of MT refers to translated and raw text, and often also language experts [Hedderich et al., 2020].

Latin, as a low-resource language, is affected by this tendency as well. As the language of science and church, Latin was of great importance for communication for many centuries [Garcia and Tejedor, 2020]. Nevertheless, since Latin is considered an extinct language, many Latin documents and texts are not digitally available or contain a close translation. However, the Bullinger Digital Project[1] aims to make Latin texts and their translations into German digitally available in a database [Fischer et al., 2022]. The text covers the correspondence of Heinrich Bullinger, a Swiss reformer, containing 16^{th} century epistolary Latin. 15% of the letters contained in his correspondence are written by himself, the majority are letters he received from several other writers. In order to contribute to the improvement of NMT for Latin as a language with fewer resources, I am motivated to investigate a Data Augmentation approach for translation from Latin into German. In NMT, Latin–German is a low-resource language pair. While the NMT system developed by Fischer et al. [2022] performs well on short and medium Latin sentences, long sequences still pose serious challenges to the NMT model, as can be seen in the following examples from the Bullinger corpus [Volk et al., 2022b].

The first part of the example is the source sentence, accompanied by the English literal word-by-word translations and the corresponding morphological information.

[1] https://www.bullinger-digital.ch/about

The second part shows the translation of the Bullinger NMT system into German (MT-de.), a corrected German translation (de.), as well as a corrected English translation (en.).

(1.1) *Novarum* *rerum* *nihil* *nunc* *est,* *quod*
New$_{.Gen.Pl}$ things$_{.Gen.Pl}$ nothing now be$_{.3.Sg.Prs.Act.Ind}$ that
scribam.
write$_{.1.Sg.Fut.Act.Ind}$

MT-de. 'Ich habe jetzt nichts Neues zu schreiben.'

de. 'Es gibt jetzt nichts Neues, das ich schreiben werde.'

en. 'There is nothing new now that I will write.'
[accessed 10th October 2022]

(1.2) *Doctissime* *et* *in* *domino* *charissime* *frater,*
Educated$_{.Sg.Sup}$ and in Lord$_{.Dat.Sg}$ dear$_{.Sg.Sup}$ brother$_{.Voc.Sg}$,
scias *velim* *impossibile* *prorsus* *vel* *per*
know$_{.2.Sg.Prs.Act.Sbjv}$ want$_{.1.Sg.Prs.Act.Sbjv}$ impossible complete or for
horas *multas* *rerum* *hactenus* *Berne*
hours$_{.Acc.Pl}$ many$_{.Acc.Pl}$ things$_{.Gen.Pl}$ until.now in.Bern$_{.Dat.Sg}$
gestarum *plenam* *tibi* *seriem* *significare.*
achievements$_{.Gen.Pl}$ complete$_{.Acc.Sg}$ you$_{.Dat.Sg}$ series$_{.Acc.Sg}$ convey$_{.Inf}$.

MT-de. 'Mein gelehrter und liebster Bruder im Herrn, ich wünschte, dass du weisst, dass du dir ganz unmöglich oder seit vielen Stunden die volle Reihe von Ereignissen in Bern gezeigt hast.'

de. 'Gelehrtester und dem Herrn liebster Bruder, ich möchte, das du weisst, dass ich dir unmöglich gänzlich oder über viele Stunden die volle Reihe von Ereignissen, die bis jetzt in Bern geschehen sind, aufzeigen kann.'

en. 'Most learned and dearest brother to the Lord, I would like you to know that it is impossible for me to convey to you completely or over many hours the complete series of events that have happened in Bern so far.'
[accessed 10th October 2022]

While the short sentence in example (1.1) is translated well and the meaning is correctly conveyed, the longer sentence in example (1.2) shows a few problematic translation parts. Most importantly, the subject is not recognized correctly, as is visible in the second part of the translation in ref. *gezeigt hast* (en. *you have shown*). The subject is interpreted as the second person singular, while it is in fact the first person singular visible in la. *velim* [Fischer et al., 2022].

In Neural Machine Translations, long sentences are translated in good quality in high-resource language pairs, models show poor performance in their translation in

low-resource languages such as Latin. This makes the translation of long sentences a major issue in low-resource scenarios [Kondo et al., 2021]. On the one hand, this is due to lack of training data available in low-resource scenarios [Kondo et al., 2021]. While NMT systems for high-resource languages have further improved due to more frequent use of deep neural networks and large language models, building systems for low-resource languages is more challenging, as neural networks require large amounts of data. [Hedderich et al., 2020]. Low-resource languages like Latin typically lack unlabelled and labelled textual data, as well as language experts and native speakers [Magueresse et al., 2020]. On the other hand, the quality of long sentence translations also depends on the NMT architecture [Neishi and Yoshinaga, 2019; Tien and Minh, 2019], as different neural models handle long sequences differently, which will be described in detail in Section 3.1.

But why is it an important task? Latin, even though it was primarily spoken in one city, has universal heritage that has extended its regional and temporal distribution widely. Latin finds its continuations in the Romance languages. Furthermore, it shows a large amount of loan word vocabulary in non-Romance languages, especially in the technical domain. For centuries, Latin was an important language for international communication, particularly in the field of science [McGillivray, 2013]. Considering the impact Latin has had on many other languages, preserving linguistic data in Latin digitally and making translations of Latin accessible provides important language knowledge to more speakers of different languages [Magueresse et al., 2020]. With Data Augmentation, namely the specific approach of splitting long sentences into smaller units, I intend to test a technique to improve the quality of such long sentence translations.

1.2 Research Questions

The research questions to be answered in this thesis are the following:

1. Can the quality of Neural Machine Translation of the 16^{th} century epistolary Latin into German be improved by splitting long sequences into smaller units?
2. Can this be done by automatically splitting the sentences and putting the translated partial sequences back together after the separate translation into German?
3. Can quality be improved by adding the split sequences separately to the training material?

4. When splitting, what are sensible sentence positions to perform the splits?

The work process in this thesis is to extract a sample of long Latin sentences from the corpus, to apply and evaluate several POS-taggers, and to split the POS-tagged sentences in order to tackle the long sentence problem. I start by first extracting a random sample of long Latin sentences of different length categories, and then apply different POS-taggers to the sentences and evaluate them to figure out which tagger yields the best quality. With a POS-tagging gold standard for these sentence tokens, I explore which possible splits of sentences yield better results than translating the complete longer sentence.

The aim of this thesis is to investigate whether the quality of NMT can be improved by applying an NLP Data Augmentation approach. I evaluate and compare the translation results, and finally define whether this Data Augmentation technique is promising for further research and exploration and whether it can be extended to more low-resource languages and scenarios in the future.

1.3 Thesis Structure

In this first chapter 1, I introduce the research questions of this thesis and explain why it is an important and worthwhile endeavour to improve NLP tools for low-resource languages. Chapter 2 introduces the related linguistic background of Latin, the low-resource language at hand, as most Latin sentences have a more complex syntactical structure than English, for example. Chapter 3 covers the respective technical background. I introduce the functionality of the state-of-the-art NMT systems and discuss related technical challenges when these systems are applied to low-resource languages such as Latin.

Having covered the base knowledge of the language and technology in the previous chapters, I continue with chapter 4, which presents related work on NLP with Latin and Data Augmentation approaches. Chapter 5 introduces the data used in the project the thesis is part of. I describe the corpus and the accessibility of the used data. The 6^{th} chapter describes different tools and resources that I used for the subtasks of the thesis such as the programming language, the POS-taggers, and the Machine Translation (MT) system.

Based on this state of knowledge, I elaborate on the resulting methods for all the steps of the splitting task in chapter 7. Chapter 8 deals with the related results of the POS-tagging and the splitting step, as well as with an error analysis for each and an evaluation of the thesis' methods. A conclusion follows in chapter 9, where I

point out the findings of this thesis and future challenges and possibilities to further improve NMT for Latin and low-resource languages in general.

2 Linguistic Background

2.1 Characteristics of the Latin Language and the Bullinger Letters

Latin has served as the main language in science and as the language of the catholic church for many centuries. Nevertheless, there are only a few texts which are digitally available, and, consequently, there are even fewer Latin texts with a close translation. This makes Latin a low-resource language. In the context of Neural Machine Translation (NMT), this becomes important as for most neural models, a large amount of data is needed [Fischer et al., 2022].

As will be discussed in more detail in Section 5, the Latin training data for the Latin–German NMT system is collected from several sources. The respective language of these sources ranges from Classical Latin texts to modern day publications of the Vatican [Volk et al., 2022a]. Since one goal of the Bullinger project is to translate the Bullinger letters into German, it is crucial to state that the Bullinger letters are written in 16^{th} century epistolary Latin [Fischer et al., 2022]. For NLP, however, 16^{th} century Latin texts have the advantage that the writing usually follows the standard of Classical Latin [Volk et al., 2022a].

2.1.1 The Complexity of Defining "Latin"

Among the linguistic aspects that complicate the processing of Latin is the fact that, in the context of NMT, Latin has few resources, that it appears in many diachronic dimensions, and that it is a historical language, which means that there are no native speakers [McGillivray, 2013]. As it tends to be the case for many historical corpora, the training data of the Bullinger corpus is assembled by combining works written over different time periods by different authors [Fischer et al., 2022]. The reason for this frequent case can either be the goal of finding patterns of language change, or the simple fact that available texts are sparse [McGillivray, 2013].

The lack of native speakers for historical languages can pose a problem because manual creation of gold standard resources is more time-consuming and prone to errors than for modern languages. However, since Latin is a well-studied low-resource language because of its widespread use over time, language experts are available [McGillivray, 2013]. Gold standards are used for testing and evaluating automatic models. For the training of these models, a gold standard with task-specific labels for the data is created. In the case of this work, we first annotate a certain number of sentences of the Bullinger corpus with POS-tags in order to evaluate already existing POS-taggers for Latin, as is described in more detail in Section 7.3 [McGillivray, 2013].

But, because Latin is a historical language, the availability of large, high-quality gold standards is particularly expensive and slow. Therefore, creating and annotating large corpora and language resources for historical languages with little data even more important, as will be discussed in a chapter about the Bullinger corpus in 5.1.

The difficulty of defining the Latin language has frequently been discussed in research. In particular, it highlights the importance of the fact that any account of a language stage of Latin must deal with its diverse nature and a series of dichotomies. These dichotomies include differences such as local versus universal language, dead versus extant forms of Latin, and vulgar Latin as opposed to literary Latin [Poccetti et al., 1999]. Therefore, it is important to keep in mind that the Bullinger NMT system tuned to translate 16^{th} century epistolary Latin might not show the same performance for another stage of the Latin language, as hyperparameters are normally optimized for the data in question [Fischer et al., 2022].

While Latin was primarily spoken locally in one city, its regional and temporal distribution has gradually been extended. Therefore, the time span of its usage expanded to long after the end of the Roman empire. Thus, it is not easy to pinpoint the exact time from which on Latin can be considered a dead language. This view led to a focus on Latin grammar during the limited span of the classical era and its literary sources, which was long considered the qualitative peak that defined Latin identity [Giacomelli, 1996].

Besides the fact that Latin has its continuations in some extant languages such as the Romance languages as well as in the technical domain of a few languages not directly derived from Latin, such as English, Latin played an important role as a language of international communication, especially in the field of science. Even today Latin texts are being produced in the contexts of news published by the Vatican as well as the Latin version of the online encyclopedia Wikipedia [McGillivray, 2013].

Even though Latin has many available texts and sources compared to other extinct languages [Mayrhofer, 1980], these texts are very heterogeneous [McGillivray, 2013]. This results in an uneven distribution along various dimensions mentioned above [Poccetti et al., 1999]. However, this heterogeneity does not necessarily reduce the performance of the NMT model for the Bullinger texts, where, as mentioned above, the training data is also collected from different sources ranging from Classical Latin texts to modern day publications [Volk et al., 2022a]. This is because variation in the training data can also make a model more robust, as was shown in Alam and Anastasopoulos [2020], who confirmed the utility of training models with source-side noise leads to more robustness to non-native language inputs.

2.1.2 Characteristics of the Latin Syntax

The difficulty of translating long sentences in NMT is partly related to the syntactical structures of Latin. Today, due to the lack of native speakers, the grammaticality of syntactic structures in Latin can only be established by reference to data preserved in a textual corpus. Latin remained in use until beyond the Middle Ages, which means it continued to evolve long after it ceased to have native speakers. Thus, the well-formedness of Latin sentences is reflected by the grammatical properties of source data previously considered well-formed by native speakers [Horrocks, 2011].

Certain Indo-European languages, including Latin, are "nominative-accusative languages", since subjects and objects can formally be identified through the assignment of different grammatical cases. It is worth noting that the notion of a subject is simply grammatical and not directly linked to a semantic role, as can be seen in the following examples from Horrocks [2011]. Thus, the subject of a sentence can have the semantic role of an agent as in example (2.1), a patient as in example (2.2), a recipient as in example (2.3) or an experiencer as in example (2.4) [Horrocks, 2011].

(2.1) *Caesar* *inimicos* *interfecit*
Caesar enemies killed
en. 'Caesar killed his enemies'

(2.2) *Caesar* *mortuus* *est*
Caesar dead is
en. 'Caesar died'

(2.3) *Caesar* *donum* *accepit*
Caesar gift received
en. 'Caesar received a gift'

(2.4) *Caesar dolorem passus est*
Caesar anguish suffered is
en. 'Caesar suffered anguish'

Analyzing these examples, it becomes clear that a typical feature of Latin syntax is the absence of direct syntactic-semantic correlation. Thus, there are no structural elements in Latin that function systematically in the style of subjects and objects. For the application of NMT, this results in the fact that tokens in the function of subjects, objects and predicates do not have any fixed positions. Besides, they also have variable length and no consistent superficial properties. A characteristic of Latin is that these items can be represented as sets of words that do not necessarily have to be contiguous [Horrocks, 2011].

In English, as a contrasting example, fewer discontinuities are allowed, and the order of constituents is fixed. Here, the rich spectrum of morphological marking available to Latin has been largely lost. Therefore, the identification of the constituents of a sentence highly depends on retaining the structural coherence of them. Similarly, the correct interpretation of grammatical functions requires a fixed syntactical ordering of the constituents. In declarative sentences, for example, transitive verbs precede their objects, and subject noun phrases precede verb phrases [Horrocks, 2011].

In Latin, on the contrary, syntactic relationships are manifested through specific requirements on morphological form that bind elements together into constituents. Therefore, an attributive adjective, which is part of a noun phrase, may be positioned separately from the noun it modifies. This happens for pragmatic or stylistic reasons, while the overt morphological agreement reveals an underlying relationship, as shown in example (2.5) from Horrocks [2011].

(2.5) *et liquidum spisso secreuit ab*
and bright$_{\text{Sg.Neut.Acc}}$ dense$_{\text{Sg.Masc.Abl}}$ he-separated from
aere caelum
atmosphere$_{\text{Sg.Masc.Abl}}$ heaven$_{\text{Sg.Neut.Acc}}$
en. 'and (god) separated the bright heavens from the dense atmosphere'

In this realization, *liquidum caelum* "the bright heavens" and *a(b) spisso aere* "from the dense atmosphere" are constituents, but are not represented contiguously. However, the reordering of constituents for stylistic reasons is more common in poetry. In the domain of prose, on the other hand, although sentential constituents allow extensive freedom in the internal ordering of their components, they mostly maintain their overall structural coherence [Horrocks, 2011].

Another typical discontinuity that can be expected in the Bullinger corpus is shown in example (2.6) from Horrocks [2011]:

(2.6) *Quae* *precatus* *a* *dis* *immortalibus* *sum,* *[...]*
which-things having-prayed from gods immortal I-am, [...]
eadem *precor* *ab* *isdem* *dis* *immortalibus*
same-things I-pray from same gods immortal
en. 'What [...] I requested of the gods, [...] those same things I request of those same gods'

Here, while the direct object relative pronoun *quae* "which-things" is positioned clause-initially as it is usually the case, the periphrastic verb form *precatus sum* "I asked for", is divided by the constituent *a dis immortalibus* "by the-immortal gods". This means, *sum* stays in its default verb position, which is the final position in its immediate clause in Classical Latin. The prepositioning of *precatus* "having-asked-for" is therefore most likely because of the contrast with the following *(eadem) precor* "(the-same-things) I-ask-for" to contrast the time reference (i.e. "what I asked for then, I also ask for now"). Most discontinuities in Classical Latin prose appear in this kind of structure with grammatically driven or pragmatically motivated preposings [Horrocks, 2011].

As the Bullinger letters mainly follow the syntactic pattern of Classical Latin syntax [Volk et al., 2022a], it is important to note that every grammatically correctly formed sentence of Latin consists of one or more clauses. Each of these clauses denote a complete semantic predication. When a sentence consists of multiple clauses, the overall sentential meaning is formed by the meanings of the combined component clauses. The composition of the relevant elements is indicated syntactically through coordination and/or subordination. Whereas main clauses are independent and can stand alone, they can be combined in one sentence through coordination with a conjunction. Each main clause contains a noun phrase subject and a verb phrase predicate, containing the finite verb form related to the subject. In turn, each main clause can contain at least one subordinating clause, consisting of complements or adjuncts [Horrocks, 2011]. When splitting long sentences with presumably higher numbers of syntactical discontinuities, such clauses are the most interesting entities to split, since they contain a complete semantic unit, as will be discussed in chapter 7.

2.2 The Challenge of Processing Low-Resource Languages in NLP

A low-resource language is considered a language which has few or no labelled or unlabelled data. As "low-resource" is an umbrella term, the boundaries concerning

the amount or kind of resources are not clearly defined [Hedderich et al., 2020]. As a consequence, these are mostly languages that are either endangered, less computerized or digitized, or less studied and, consequently, less commonly taught. Hence, low-resource languages are often those which have a low density or low prestige. These qualities often interact or are mutually dependent [Magueresse et al., 2020].

Most often, low-resource scenarios occur when a language lacks task-specific labels in the target language, especially in the context of supervised learning. However, it is not defined at which level of resource-scarcity a language is still considered to be a low-resource language, as the availability of labelled, unlabelled, and auxiliary data is a gradual matter [Magueresse et al., 2020]. The lack of these task-specific labels can be remedied by creating them through manual annotation. But, as low-resource scenarios often co-occur with further scarcity in resources such as financial ones (which can again be related to other factors like prestige or number of speakers), this is not always feasible and can be time- and cost-intensive [Hedderich et al., 2020].

A solution may be to simply use raw text and make use of input embeddings which are trained on unlabelled texts. However, sometimes languages are missing unlabelled texts and even auxiliary data. Auxiliary data typically denotes data with task-specific labels in a different high-resource language, which can be used for example for Transfer Learning, or it can consist of external information sources in form of knowledge bases or gazetteers used for distant supervision. In case of MT, auxiliary data mostly includes other NLP tools for the target language itself, which contribute to the generation of training data [Hedderich et al., 2020].

Consequently, NMT frequently depends on resources such as language-specific data and tools in order to be conducted. Generally, the most problematic case is when not enough unlabelled data exists, which also limits the generation of labelled and auxiliary data needed for the automatic creation of unlabelled data [Hedderich et al., 2020].

In the past few decades, NLP research has led to greatly evolved technologies. Similarly, a lot of research on the creation of tools and techniques to deal with shortages in data and resources has been carried out. Most NLP tasks, including MT, have transitioned from rule-based to statistical techniques, whereas today most techniques make use of neural techniques [Magueresse et al., 2020] due to the evolution of deep neural architectures and optimized hardware. For high-resource languages, the development was optimal because of the availability of large corpora [Chernyavskiy et al., 2021].

As a consequence of these high-quality outcomes for neural techniques with high-resource languages, there has been more research on neural applications for low-resource scenarios [Hedderich et al., 2020]. The techniques on how to deal with a low-resource language such as Latin in NMT will be investigated in the following sections on NMT with low resources in Section 3.2, and on Latin in NMT in Section 3.3, and on related work on NLP with Latin in general in Section 4.1.

3 Technical Background

In the case of the Bullinger corpus, several different optimized NMT models are trained b< Fischer et al. [2022] for the translation from Latin to German, as the they gradually add more training data to the corpus. GoogleTranslate[1] is chosen as a baseline for the comparison of the NMT systems. To measure the performance of the different models, the Fischer et al. [2022] choose the BLEU metric [Papineni et al., 2002]. During the development of the models in the Bullinger Digital Project, GoogleTranslate switched from Statistical Machine Translation (SMT) to Neural Machine Translation (NMT). In early 2021, GoogleTranslate exclusively used SMT for language pairs including Latin. In the meantime, however, GoogleTranslate has created an NMT model for language pairs with Latin as a source language to all available target languages. While the initial BLEU score of the online system with the SMT model scored 7.36, it reaches 17.07 for translations from Latin to German for GoogleTranslate's NMT model [Fischer et al., 2022].

After discussing NMT Models in Section 3.1.1 and various typical approaches that deal with low-resource languages in NMT in Section 3.2, I will outline several experiments Fischer et al. [2022] run during the building process of the Bullinger corpus in 3.3.

3.1 Neural Machine Translation

Machine Translation (MT) denotes the process of automatic translation of text from one natural language into another. In recent years, the paradigm has shifted from Statistical MT to Neural MT. Statistical MT mainly relies on different count-based models and dominated MT research for two decades. NMT solves the translation task with a single neural network and has largely superseded Statistical MT [Stahlberg, 2020].

The principle behind SMT is a combination of two separate processes, namely train-

[1] https://translate.google.com

ing and decoding. The training process involves a statistical translation model which is trained on a parallel corpus, as well as a statistical language model trained on the target language from a monolingual corpus. The translation model deals with the adequacy of the translation from source to target sentence, while the language model deals with the fluency of the output sentence in the target language. During decoding, i.e. the process which actually yields a translation, these two models are then applied, treating the translation as a search problem. It searches through all possible translations and reorderings permitted by the translation model in order to find the one with the highest probability according to the translation and language model [Way and Hearne, 2011].

NMT, on the other hand, uses a single large sequence model, a neural network, which transforms the source sentence into the target sentence. The prediction output is conditioned on the entire source input sequence and the previously produced target sequence. This strategy overcomes the separation of the log-linear model combination from SMT. The integration of neural networks into MT systems after their rediscovery was rather shallow at first. While the neural techniques boosted other fields of NLP, the integration process in MT was much slower. Early attempts used traditional SMT systems, only applying neural networks as components within them, while more recent approaches typically transform the source sentence into the target sentence directly [Stahlberg, 2020].

Embeddings are a key to NMT: They are representations of words or phrases as continuous vectors. This vector encodes the meaning of a word or phrase in such a way that similar meanings are close to each other in the vector space. This representation has the potential to capture morphological, syntactic, and semantic similarity across words or phrases. Embedding matrices are often trained jointly with the rest of the network in NMT. It is also possible to reuse pretrained embeddings trained on unlabelled text beforehand [Stahlberg, 2020]. In the Bullinger Digital Project, Fischer et al. [2022] use sentence embeddings as they are a representations of longer pieces of text than word embeddings and capture similarity between sentences.

NMT models that process sequences are also called sequence-to-sequence models or Encoder-Decoder models. In Encoder-Decoder models, the output sequence is a complex function of the entire input sequence. A sequence of input words or tokens has to be mapped to a sequence of tags that are not directly mapped to individual words [Stahlberg, 2020]. I illustrate the functionality of Encoder-Decoder models in Section 3.1.1 below.

3.1.1 Encoder-Decoder Models

Fischer et al. [2022] use the Transformer architecture in all experiments. More precisely, they implement its base configuration by Vaswani et al. [2017] and the SOCKEYE framework by Hieber et al. [2017]. Their goal is to augment training data and continually optimize hyperparameters, since this greatly improves the translation quality for low-resource NMT. Hyperparameters depend on the size of the training data, which is why the different models are gradually optimized and change with increasing training data size [Fischer et al., 2022].

The Transformer architecture is based on the concept of an Encoder-Decoder model. Such architectures are available via Transformers[2], which is an open-source library consisting of several engineered state-of-the-art Transformer architectures with the goal of opening up the technology to a wider community [Wolf et al., 2020]. With an Encoder-Decoder model, it is possible to encode a input token sequence with variable length into a sequence of vector representations. As the name suggests, this is the encoding step of the translation process. These representations can then be decoded into a sequence of output tokens during the decoding step. Thus, the model does not only capture meanings and interactions at word level, but takes longer sequences into account. The decoding is conditioned on information from both the encodings of the input vector as well as its continually updated internal state [Hieber et al., 2017].

More recently, the concept of attention was introduced [Bahdanau et al., 2014]. With attentional Encoder-Decoder models, fixed-length source sentence representations can be avoided, and the model does not rely on a constant context vector encoding of the complete source sentence anymore. The context vector represents the weighted sums of source sentence annotations. Limited capacity of a fixed context vector is especially problematic with longer input sequences. Instead, during attentional decoding, attention is placed on parts of the source sentence which are useful for the generation of the next token. Instead of one context vector, there are several context vectors, each representing one time step [Stahlberg, 2020].

A generalization of attention is multi-head attention [Vaswani et al., 2017], which consists of several performed attention operations instead of a single one. The number of attention heads is typically 8. The outputs of the independent attention heads are then concatenated. With multiple attention heads, the parts of the sequence can be attended to differently [Stahlberg, 2020].

[2]https://huggingface.co/docs/transformers/index

Besides self-attentional Transformers, there are two other Encoder-Decoder architectures: Attentional recurrent neural networks, and fully convolutional networks. These three architectures have been among the most prominent ones, representing the state-of-the-art in NMT [Hieber et al., 2017].

3.1.1.1 Recurrent Neural Machine Translation

Recurrent Neural Networks (RNNs) were the first to be introduced. The encoder consists of a bidirectional Recurrent Neural Network to encode a source sentence. To predict the output words in the target language, a decoder consisting of a second RNN is used. To avoid difficulties in encoding long sequences into a context vector, attention mechanisms can be applied in RNNs [Stahlberg, 2020].

A peculiarity of RNNs is their recurrent dependency on the previous time step. As a consequence, the computation of RNN hidden states during encoding cannot be parallelized over time. During decoding, the serialization of the computation is even bigger as the first time step has to be completed before the second time step can be started [Hieber et al., 2017].

For attentional RNN architectures, long sentences pose a challenge because the values of hidden states in attention models are too dispersed. As a result, the context vector does not provide useful predictive support to the network during the decision for the next target word [Tien and Minh, 2019].

3.1.1.2 Convolutional Neural Machine Translation

Convolutional Neural Networks (CNNs) are more useful for long continuous sequences, as they do not rely on a serial sequence assumption [Stahlberg, 2020]. CNNs have a faster and simpler architecture based on a succession of convolutional layers. Convolutional layers are capable of extracting different features from the input by convolving the input and passing its result to the next layer. Compared to RNNs, the architecture does not include temporal dependencies, which allows to encode the source sentence simultaneously [Gehring et al., 2016].

An advantage of the CNN architecture is an easier parallelization on GPU hardware which reduces sequential computation. Also, their hierarchical structure connects distant words through a shorter path than sequential architectures. By stacking multiple convolutional layers, the context size can be increased, which is useful for the translation of long sentences. Such deeper models are, however, more difficult to train. During decoding, future information of the next time steps needs to be

masked, and the decoder is connected to the encoder by attention [Stahlberg, 2020].

3.1.1.3 Self-Attentional Neural Machine Translation

For several language pairs, self-attention-based models such as the Transformer [Vaswani et al., 2017] remain the dominant architecture. The layers of a self-attentional Transformer model learn dependencies between words within a sequence itself. Besides NMT, self-attention is often applied to NLP tasks such as sentiment analysis, text summarization, or sentence embedding. An advantage of self-attentional models is, similar to CNNs, short paths between distant words and reduction of sequential computation [Stahlberg, 2020].

The Transformer, the first example of this NMT model class, uses self-attention in the encoder in order to enable context-sensitive word representations depending on the whole source sentence, and within the decoder, in order to account for the current translation history. More specifically, the architecture uses multi-head attention. Furthermore, cross-attention is used between the encoder and the decoder [Stahlberg, 2020].

For the Transformer architecture, long sentences are more challenging to translate, too. While using attention mechanisms has partially remedied the problem, literature suggests further possible reasons for a performance drop [Neishi and Yoshinaga, 2019]. One potential cause is assumed to be how position information is handled by the model. This position information can either be handled in relative or in absolute form. The model attends to relative positions from the periodicity of positional encodings generated using sinusoids of varying frequencies. Position embeddings, on the other hand, are learned position vectors, which represent absolute positions. Neishi and Yoshinaga [2019] suggest that the type of relative position is better for the model performance than the type of absolute position.

Furthermore, the standard Transformer architecture uses a maximum length, truncation, and padding in order to deal with different sequence lengths [Dai et al., 2019]. This leads to a barrier for extended dependency learning, since the attention mechanism is not able to grasp connections beyond this maximum length limit. This lack of contextual information leads to inefficient prediction and thus to a compromised performance [Singh and Mahmood, 2021]. It is also assumed that the performance decrease is partially caused by an insufficient number of long sentences in the training data. Low-resource languages are therefore even more affected by this issue [Kondo et al., 2021].

3.1.2 Neural Machine Translation Decoding

Decoding, or inference, is the task of finding the most likely translation for a given source sentence. With an increasing sequence length, the search space increases exponentially. Popular decoding algorithms to solve this problem are greedy search and beam search. Greedy search works in a time-synchronous manner and, at each time step, selects the single best expansion. A disadvantage of greedy search is the fact that, when only choosing one best expansion, the overall score of a path may end up being comparably low. Beam search, on the contrary, passes a fixed number of possible translation prefixes to the next time step. At each time step, the accumulated scores for all possible continuations of the surviving hypotheses are compared. The selection of fixed number of hypotheses to continue is based on this comparison [Stahlberg, 2020].

3.2 Low-resource Scenarios in Neural Machine Translation

While NMT has achieved qualitatively high performance in high-resource settings, the performance drops in low-resource data conditions. The systems occasionally even underperform phrase-based SMT. However, in literature, it is argued that the performance drop is rather due to a lack of system adaptation to these different conditions of low-resource settings [Sennrich and Zhang, 2019].

Besides an optimized system adaptation, recent research has focused on specific techniques to train translation systems in low-resource scenarios. NMT models generally work best when large amounts of data are available. This data can be manually annotated or generated via direct manual translations which can be used as training data as done in the Bullinger Digital Project. Nevertheless, additional techniques may be required [Hedderich et al., 2020]. Some common techniques are explained in the following chapters.

3.2.1 Transfer Learning

One common technique to deal with low-resource scenarios is called "Transfer Learning". This approach tries to achieve better language representations directly by transferring learned representations and models without the need for labelled target data [Hedderich et al., 2020]. In an MT context, it uses a two-model architecture

with the first model trained on two high-resource languages and the second model covering the source and target language of the translation task. Here, the embeddings of the second model are initialized with the training embeddings with a standard corpus of the first model. Consequently, the second model can be trained more efficiently even with a small amount of data [Magueresse et al., 2020].

This approach of Transfer Learning works best when the involved languages share some linguistic similarities. The technique can be further optimized by using a reordering model which reorders sequences or complete sentences of the source language to make them syntactically more similar to the target language. This way, the transfer of embeddings leads to a better accuracy of the model due to the fact that corresponding words have more similar positions [Murthy et al., 2018].

Transfer Learning can also be applied by introducing a previous model during training. An encoding model can be trained with multiple languages, which results in the model being able to use previously learned language pairs to translate between unseen language pairs. This even works if the target pair has not been trained in a source-target language combination [Magueresse et al., 2020]. Similarly, in the Bullinger Project, Fischer et al. [2022] added a pretraining step to the pipeline. The model has become more robust as it learned to produce the target language, German, more fluently, since a larger training corpus was used. The source language chosen was Italian, as it should be closely related to the source language of the NMT system [Zoph et al., 2016].

The term "Multilingual Learning" refers to another effective form of Transfer Learning applied in MT. In this approach, a shared lexicon and a shared sentence-level embedding are used to train a neural MT model. Thus, the model can be trained on several languages, sometimes with the use of a universal lexical word representation for the design of common embeddings [Magueresse et al., 2020].

3.2.2 Data Augmentation

The traditional approach for low-resource NLP mentioned in the literature is Data Augmentation. More data can be generated or collected in order to obtain better results in low-resource scenarios. Even though this approach can be costly, it is widely used in research [Hedderich et al., 2020].

Data Augmentation refers to generating more labelled data and is usually based on the availability of some labelled data and linguistic knowledge. This leads to a conversion of the low-resource scenario into a high-resource scenario. Literature

reports good results for this method [Hedderich et al., 2020]. However, the downside is that it requires extensive and expensive preparatory work, which can be even more problematic if the target language does not have a significant internet presence, as is the case for Latin [Hedderich et al., 2020].

In Data Augmentation, there are currently three main implementation methods. Firstly, raw text can be annotated by linguistic experts in order to create new or bigger data sets. Common sources for creating raw texts for new data sets are typically mobile applications or social media, as well as governmental documents [Magueresse et al., 2020]. In the Bullinger Project, sources such as social media cannot be used as Latin is considered an extinct language. Nevertheless, there are modern day publications by the Vatican in Latin which are included in the training data. Therefore, sources in the projects' training data cover the language span from modern Latin texts to Classical Latin texts [Volk et al., 2022a].

The second method for the implementation of Data Augmentation is automatic alignment. In this case, raw text is gathered and aligned with text in a higher-resource language. Even though this has been the research focus of many recent studies, it requires linguistic knowledge and is extremely time-consuming. However, it is a useful method as most low-resource scenarios are based on aligned corpora [Magueresse et al., 2020].

While this has also been done on the word-level, sentence-level alignment is more popular and particularly useful for MT. Typically, the alignment is implemented via a similarity score between two sentences. A challenge for low-resource languages is the fact that word-to-word translations preferably require a larger amount of data, which means that these systems require a higher quality translation [Magueresse et al., 2020]. For the collection of the training data of the Bullinger NMT system, Fischer et al. [2022] used automatic tools such as the LASER library [Schwenk et al., 2017] with the Bitext Miner Algorithm [Schwenk et al., 2021] and the Vecalign Algorithm [Thompson and Koehn, 2019] to perform the sentence alignment.

The third trend consists of the modification of specific features to obtain more text instances and sequences. Modifications should be done in a way that does not change the labels. In Data Augmentation approaches such as this one, language models taking context into account can be used as support [Hedderich et al., 2020]. A similar way to perform these modifications is to back-translate the target sentence into the source sentence to obtain paraphrases with the same content [Hoang et al., 2018].

In conclusion, even though Data Augmentation can be time- and cost-intensive,

it offers benefits similar to pretraining in transformer models. Therefore, it is a useful approach that can be combined with further task-specific language technology techniques when unlabelled data is limited [Hedderich et al., 2020].

3.3 Neural Machine Translation of the Latin in the Bullinger Letters

As mentioned in Section 3.1.1, Fischer et al. [2022] use GoogleTranslate as baseline to compare customized NMT systems. The size and domain of the training data plays a great role in the performance of different tested models. In the first experiment, they use training data of 150,000 sentence pairs at that time. This model reaches a BLEU score of 11.14. It already outperforms the SMT baseline by a great margin, but the GoogleTranslate NMT baseline still works better with additional 6 BLEU points on the Bullinger test set [Fischer et al., 2022].

By adding additional corpora to the training data, the BLEU score can be raised to 12.15 BLEU points in a first step with an increase of 21,000 segments, and later to 13.72 with an increase of another 24,000 segments. This shows that the size of the training data has an impact on the performance of the NMT model. Even though increasing the training data size optimizes the NMT output, all NMT models struggle with longer sentences and tend to show lower performance quality in easy tasks such as translating dates [Fischer et al., 2022].

In order to make the model more robust in terms of e.g. preserving numbers better, Fischer et al. [2022] add a pretraining step to the pipeline. Since, according to Zoph et al. [2016], the source language of a pretraining model should be closely related to the source language of the NMT system, Italian is used. From this larger training corpus, the model learns to output target language, in our case German, more fluently. The experiment with the pretraining step involves training an Italian to German NMT system on the Italian-German data, then replacing the training data with the Latin–German corpora, and finally continuing the training on this data. The result achieves an increase in 1.2 BLEU points [Fischer et al., 2022].

Adding another 16,000 segments to the Latin training data and maintaining the pretraining step increases the result by another 1.5 BLEU points. Exchanging the pretraining dataset by another one, which increases the data size from 1,2 million to 6 million segments, results in an improvement of the score of the LA–DE model by another 0.6 BLEU points. With this score, the Bullinger NMT model has reached the performance of GoogleTranslate's NMT baseline [Fischer et al., 2022].

With a normalization step during the preprocessing in the pipeline of the Latin segments of the training data, special characters such as ligatures like *œ* are simplified and automatically split into their base characters, here *ae*. Namely, the Classical Language Toolkit normalizer (Johnson et al., 2021) is used to preprocess the Latin segments. The normalization step raises the BLEU score to 19.5, which shows that it greatly improves the translation quality. While the NMT system still struggles with producing accurate translations for longer and more complicated sentences, this setup outperforms the GoogleTranslate baseline by 2 BLEU points [Fischer et al., 2022].

4 Related Work on NLP for Latin

In this chapter, I discuss relevant background literature and describe how different approaches from the literature are related to this thesis. Section 4.1 discusses methods for Natural Language Processing with Latin, whereas Section 4.2 presents similar techniques of Data Augmentation.

4.1 Work on Natural Language Processing of the Latin Language

This thesis investigates an approach to optimize Neural Machine Translation (NMT) for long sentences in Latin. Even though Latin is not the most researched language in NLP, there are several methods from the literature that to process Latin. This research includes papers dealing with OCR corrections, lemmatization, POS-tagging, corpus or lexicon creation, as well as code-switching and Machine Translation. Mokhtar et al. [2018], for example, propose a new approach to OCR error correction. As OCR systems tend to worsen on historical documents with old manuscripts, the authors propose NMT-based approaches and include a Latin dataset to test their methods [Mokhtar et al., 2018].

One method to carry out POS-tagging for Latin texts is proposed by Guarasci [2017], who uses Wikipedia to develop an annotator. Texts from Wikipedia are also used for the creation in the Bullinger training corpus, as will be described in Section 5.1. In this case, however, Wikipedia is used as a resource for the development of a POS-tagger as well as a Wikipedia-based semantic annotator [Guarasci, 2017]. Another paper offering insights about linguistic annotation is provided by Gries and Berez [2017]. The authors summarize different annotation formats, stating that XML annotation has become a widespread form of annotation [Gries and Berez, 2017]. POS-tagging for Latin is also explored by Stoeckel et al. [2020] as part of the EvaLatin[1] Shared Task for Lemmatization and POS-tagging. The authors develop

[1]https://circse.github.io/LT4HALA/2022/EvaLatin.html

an ensemble classifier called LSTMVoter, which is based on several trained state-of-the-art taggers [Stoeckel et al., 2020]. The difference with POS-tagging in this thesis is that instead of an ensemble classifier, the model with the best accuracy is chosen for further work. One step further than the POS-tagging method by Stoeckel et al. [2020], Erdmann et al. [2016] investigate Named Entity Recognition for Latin, using a conventional POS-tagger as an intermediate step.

Similar to the code-switching investigation on the Bullinger corpus by Volk et al. [2022a], various papers deal with code-switching between Latin and other languages, among them Garrette et al. [2015], who also work with 16^{th} century texts. In their paper, the authors investigate historical OCR as well as word-level code-switching between multiple languages, including Spanish, Nahuatl, and Latin. Another paper by Schulz and Keller [2016] inspects code-switching with the use of language identification models and POS-tagging.

Further work on Latin NLP is dedicated to building corpora, lexicons, as well as linguistic resources and tools for Latin NLP. Among them Litta et al. [2016], who build a word formation lexicon and also provide an online graphical query system to access the lexicon. Papers dealing with lemmatization include Gleim et al. [2019], who investigate POS-tagging and lemmatization in morphologically rich languages such as German and Latin. Similarly, Passarotti et al. [2017] provide further research on the morphological analysis of Latin. The authors introduce a downloadable package of the 3.0 version of Lemlat[2], a morphological analyser for Latin. Its main components include word form analysis, treatment of spelling variation, and a resource for derivational morphology of Latin.

In recent years, different NLP tools have been provided by Passarotti et al. [2019] in their "LiLa Knowledge Base of Linguistic Resources"[3], which is dedicated to building linguistic resources for Latin. In order to build rich knowledge graphs, the authors use Linked Open Data practices as well as unique identifiers to connect words to distributed textual and lexical resources. Franzini et al. [2019] investigate the expansion of the Latin WordNet. This paper‘s focus is to identify the most effective method for its inclusion in the LiLa Knowledge Base of Latin Resources. This inclusion in the Knowledge Base of word formation information is further discussed by Litta et al. [2019]. The authors present how such theoretical and practical issues are addressed in the project. This investigation of word formation by Litta et al. [2019] is continued in Passarotti et al. [2021]. Furthermore, Passarotti et al. [2020] discuss the interlinking of lemmas from the lexical collection of the LiLa Knowledge

[2]https://github.com/CIRCSE/LEMLAT3

[3]https://lila-erc.eu

Base and related challenges raised by harmonizing different lemmatization strategies with different linguistic resources for Latin.

In the field of MT, several papers deal with the Latin language as well. From the Transformer[4] models collection, Bamman and Burns [2020] introduce a contextual language model called Latin BERT[5]. While Fischer et al. [2022] use the base configuration of the Transformer model [Vaswani et al., 2017], Latin BERT is trained on a variety of sources spanning the Classical era to the 21st century in order to perform POS-tagging tasks as well as text prediction tasks, but has not (yet) been trained for NMT tasks [Bamman and Burns, 2020]. Garcia and Tejedor [2020] deal with NMT, too, but work with the language pair Latin-Spanish. The authors use a Transformer-based MT system model as well, and train it on the Bible parallel corpus. Furthermore, they use the Saint Augustine corpus to study the domain adaptation case from Bible texts to the newly built corpus. They state that using in-domain data improves the translation quality of the systems [Garcia and Tejedor, 2020].

There are a few more sources investigating different methods in NLP of Latin on a general level. An early work by McGillivray [2013] provides information about Latin corpora, tools, and various techniques for NLP until the year 2013. More recently, a general work treating digital approaches in classical philology is written by Burns [2019]. The authors discuss strategies to cope with digital resources of ancient Latin and Greek, including data collection, annotation, and open data sources for these languages [Burns, 2019]. Furthermore, Sprugnoli et al. [2022] give an overview of the second edition of EvaLatin, a campaign for the evaluation of NLP tools for Latin. They report results of three shared tasks, namely Lemmatization, POS-tagging, and Features Identification.

Recently, there has been a lot of research in Latin NLP. However, none of the methods from the literature above deals with the optimization of the Latin-German language pair in NMT exclusively besides the Bullinger project. While this section focuses on several NLP tasks with Latin, the next Section 4.2 addresses approaches similar to the sentence segmentation approach for the optimization of Latin-German NMT in this thesis.

[4]https://huggingface.co/docs/transformers/index

[5]https://github.com/dbamman/latin-bert

4.2 Work on Related Data Augmentation Techniques

Related work on sentence segmentation, compression, or simplification in order to augment the training data has been the subject of research in recent years. Translating long source sentences is a problem of NMT still not entirely solved [Li et al., 2021; Kondo et al., 2021; Sountsov and Sarawagi, 2016]. With the architectures of attentional Encoder-Decoder networks [Stahlberg, 2020], the fixed-length source sentence encoding could be replaced by an attention mechanism. This fixed length of the encoding was hypothesized to be a reason for the poor translations by Cho et al. [2014]. Their explanation is that, since this fixed-length vector is ideal for short sentences, it would not be capable of encoding the complicated structure and meaning of a long sentence [Cho et al., 2014].

Pouget-Abadie et al. [2014] work on a solution with automatic sentence segmentation by chopping the source sentence into shorter clauses. First, each segment is independently translated by the NMT model, which in their case has an RNN architecture. After that, these translated clauses are put together in a concatenation and thus form a final translation. For these long sentences, the results of Pouget-Abadie et al. [2014] show an improvement in translation quality. Nevertheless, long-distance reorderings still pose a problem in this approach, as they are only possible within a clause [Stahlberg, 2020]. Even though Fischer et al. [2022] used a Transformer model for the translation, the segmentation technique is similar to the approach used in this thesis in that subsequences of a sentence are translated individually. Instead of POS-tags, Pouget-Abadie et al. [2014] used a so-called "confidence score" to choose segments to translate. This confidence score reflects how confidently the system can translate a subsequence, which is measured by the log-probability of a generated candidate translation using an RNN Encoder–Decoder [Pouget-Abadie et al., 2014].

Another approach on automatic long sentence segmentation is proposed by Kuang and Xiong [2016]. The authors also work out a method to segment long sentences into several clauses by introducing a split and reordering model. For a long source sentence, this method collectively detects the optimal sequence of segmentation points. As in Pouget-Abadie et al. [2014], the NMT system translates each segmented clause independently into a target clause. Without the need for a specific reordering step, the translated target clauses are recomposed to a final translation. Kuang and Xiong [2016] use an RNN Encoder-Decoder model as well, but translate the clauses with an attention-based NMT system, which makes the use of a reordering system obsolete.

Inspired by the idea of long sentence segmentation into shorter clauses, another tech-

nique includes hierarchy-to-sequence attentional NMT models and is implemented by Su et al. [2018]. Their goal is to find optimal model parameters for long parallel sentences as well as exploiting different scopes of contexts better. The segmented clause serves as input to the encoder. Using a hierarchical neural network structure, words, clauses, and sentences are then modelled at different levels. In order to capture contexts for the translation prediction, the decoder applies attention models. The segmentation itself is done according to the source-side punctuation. Su et al. [2018] use punctuation markers such as commas and question marks, which is applied in this thesis as well.

Tien and Minh [2019] investigate long sentences in NMT, too, and propose a method to extract bilingual phrases in order to create a phrase-aligned bilingual corpus. To optimize the NMT model, a preprocessing technique for long sentences is implemented. The segmented phrases are obtained using the phrase table generated by the Moses toolkit [Koehn et al., 2007].

Zhang and Matsumoto [2019], who also work on a sentence segmentation approach for low-resource NMT, use a corpus augmentation method which consists in segmenting long sentences via back-translation. With the generation of pseudo-parallel sentence pairs, they are able to improve the translation performance. The authors first obtain the word alignments of parallel sentences of their language pair. As in [Su et al., 2018] and in this thesis, the splitting is implemented by taking punctuation symbols such as ",", ";", and ":" into account [Zhang and Matsumoto, 2019].

Berrichi and Mazroui [2021] develop two techniques for segmenting long sentences into smaller sub-sentences. One method uses a list of collected lexical markers as segmentation points. These markers are collected through text analysis and consist of words that serve as link between two segments of a sentence such that the segments may be examined separately at the syntactic and semantic levels. Thus, similarly to the POS-tagging information for splitting positions in this thesis, the authors use specific semantic segmenters. A second method integrates parallel phrases extracted by an SMT system into the NMT model. Like Tien and Minh [2019], the authors used the SMT system Moses for the creation of a phrase translation table [Berrichi and Mazroui, 2021].

Şahin and Steedman [2019] introduce a Data Augmentation technique involving dependency tree morphing. By "cropping" sentences through the removal of dependency links as well as "rotating" sentences by moving the other tree fragments around the root, the authors augment training sets of low-resource languages. The cropping is implemented by identifying sentence parts to focus on in the dependency tree, such as subjects and objects. By defining a focus such as the subject or the

object, they form smaller sentences and remove all dependency links other than the focus. In order to apply the rotating, the authors choose the root as sentence center. The flexible tree fragments, which are defined by the morphological typology of the language, are then rotated around the root [Şahin and Steedman, 2019].

A technique for sentence compression is proposed by Li et al. [2020]. The authors use the focus property of Transformer-based encoders, as the core of a sentence is not specifically focused on. This aims at using a relatively short sequence for maximizing the absorption and retention of large amounts of data. With the self-attentional Transformer method, the most salient part of a sentence representation can be established [Li et al., 2020]. This sentence compression method differs from this thesis' approach, as no direct segmentation is applied, but can still be important for the project as Fischer et al. [2022] also use a Transformer architecture.

Besides the segmentation of sentences, the substitution of sentence parts has been investigated in recent research as well. To deal with long sentences in NMT, Shi et al. [2021] work on techniques summarized as "substructure substitution". These techniques generate new examples while keeping the same label, whereas this process is repeated until the training set reaches a desired size. In the setting with POS-tags, the corresponding POS-tags of a text span function as substructure labels [Shi et al., 2021].

As current standard NMT model, the Transformer has difficulty to translate long sentences, Neishi and Yoshinaga [2019] explore reasons for this issue. They investigate differences between Transformers and RNN-based models in how they handle position information which is essential to process sequential data. The authors demonstrate that relative position, in contrast to absolute position, helps translating sentences that are longer than those in the training data [Neishi and Yoshinaga, 2019].

Similar to Neishi and Yoshinaga [2019], Kondo et al. [2021] come up with a new thought for the same question. They train their model differently by only using the given parallel corpora as training data in order to generate long sentences through the concatenation of two sentences. The authors assume the major issue with poor performance in long sentence translation in low-resource languages is caused by an insufficient number of long sentences in the training data. They further improve translation quality when their method is combined with backtranslation [Kondo et al., 2021].

Several sentence segmentation methods have been proposed in the field of speech recognition. [Dalva et al., 2018] propose a sentence segmentation approach including

semi-supervised learning strategies in order to determine the sentence boundaries of a stream of words that are output by automatic speech recognizers. In their paper, Wang et al. [2019] suggest a sentence segmentation approach working with multi-shifted RNNs. The goal of the authors is to segment the unpunctuated transcripts which are generated by automatic speech recognition for a simultaneous interpretation. The multi-shifted RNN model applies sentence segmentation by shifting target signals by multiple durations of time to ensure the next few words belong to a new sentence [Wang et al., 2019]. Similar to Dalva et al. [2018] and Wang et al. [2019], Li et al. [2021] state that sentence boundary segmentation has a large impact on quality of NMT performances, which they can be solved by implementing a Data Augmentation strategy to expose the model to bad segmentations during training.

Other Data Augmentation methods have been proposed in recent research. Among them, Kumar et al. [2020] present an approach for Data Augmentation using pre-trained Transformer models. Fischer et al. [2022] use pretrained models in the context of the pretraining step in the pipeline in order for the model to learn fluent German from a larger training corpus. The chosen language pair for the corpus is German-Italian, as Italian is closely related to Latin [Fischer et al., 2022].

Further Data Augmentation techniques for low-resource translations are investigated by Xia et al. [2019]. On a more general basis, Feng et al. [2021] discuss recent Data Augmentation approaches for NLP. Similarly to Feng et al. [2021], Chen et al. [2021] describe Data Augmentation for limited data conditions.

Despite many useful techniques of sentence segmentation and simplification discussed in this section, the translation of long sentences in NMT is still a challenge [Stahlberg, 2020], which is also the case for the Bullinger corpus [Fischer et al., 2022]. In this context, Section 5.1 highlights the advances in translation quality as the corpus grows, and Section 7.4 explains the sentence segmentation methods of this work.

5 Corpus and Training Data

5.1 The Corpus

The text corpus of letters used in the Bullinger Digital Project, referred to as Bullinger corpus in this thesis, contains the correspondence of Swiss reformer Heinrich Bullinger, who lived from 1504 until 1575. There are different editions of his letters. In ongoing projects, letters since 1523 have been edited and published in chronological order in around 20 volumes.

Many original letters of the HBBW edition (Heinrich Bullinger Briefwechsel) are stored in the State Archives of the Canton of Zurich (Staatsarchiv des Kantons Zürich[1]) and in the Zurich Central Library (Zentralbibliothek Zürich[2]). The HBBW edition is created by the Institute for Swiss Reformation History[3] and its letters are accessible to the public via an electronic edition in PDF[4].

These letters make up around 90% of the Bullinger correspondence. The other originals of the letters are scattered in libraries all over the world, and a particularly large number of them are in the Cantonal Library of Vadiana[5] in St. Gallen, where the VBS edition is stored.

Today, the work on the Bullinger correspondence is divided into two subprojects. The Heinrich Bullinger-Stiftung[6] ensures further publication of the letters (transcription, commentary, and translation) and the continuation of the edition in book form. The edition enterprise has its own website[7].

The second subproject commissioned by the Department of Computational Linguis-

[1]https://www.zh.ch/de/direktion-der-justiz-und-des-innern/staatsarchiv.html
[2]https://www.zb.uzh.ch/de/
[3]https://www.uzh.ch/cmsssl/irg/de.html
[4]http://teoirgsed.uzh.ch/
[5]https://www.sg.ch/kultur/kantonsbibliothek-vadiana.html
[6]https://www.bullinger-stiftung.ch
[7]https://www.irg.uzh.ch/de/bullinger-edition.html

tics at the University of Zurich[8] includes the creation of a database with metadata from each individual letter and links to scans currently being created in the State Archives of the Canton of Zurich (Staatsarchiv des Kantons Zürich[9]) and in the Zurich Central Library[10].

The Heinrich Bullinger-Stiftung[11] has the goal to make Heinrich Bullinger's correspondence accessible to the public and to facilitate new historical and linguistic research. To do this, the correspondence must be deciphered, translated, explained in terms of content and placed in its cultural-historical context. The aim of the Bullinger Digital Project[12] is to make the letters that have not yet been edited digitally accessible and to bring all the correspondence together in a database accessible via the Internet.

The Bullinger corpus consists of more or less 10'000 letters written to Heinrich Bullinger in the 16th century, and additional 2'000 letters written by himself to members of his correspondence network. The letters include topics such as politics and religion and cover a wide range in terms of formality. With the customized MT model for 16th century Latin, all Latin sentences can be translated into modern German [Fischer et al., 2022]. This chapter describes the data and detailed methods used in the creation of the Bullinger corpus and the training data.

For the Bullinger Digital Project, most of the training data is collected by the research team. They used sentence alignment tools such as the LASER (Language-Agnostic SEntence Representations) library [Schwenk et al., 2017], which is applied to the data in combination with the Bitext Miner Algorithm [Schwenk et al., 2021] and the Vecalign Algorithm [Thompson and Koehn, 2019]. The LASER library comes with an encoder to create sentence embeddings. This encoder is trained on Latin and German and almost 100 languages more. With these sentence embeddings, similar sentences can be found across languages [Fischer et al., 2022].

As pointed out in 3.3, Latin is a low-resource language coming with only few texts and parallel data, which makes NMT for the Latin-German language pair challenging. On the OPUS website for example [Tiedemann, 2016], which is a website that hosts many parallel corpora, only 100,000 translated sentences are available for the language pair Latin-German, and also other combinations with Latin do not show greater numbers of segments [Fischer et al., 2022].

[8] https://www.bullinger-digital.ch

[9] https://www.zh.ch/de/direktion-der-justiz-und-des-innern/staatsarchiv.html

[10] https://www.zb.uzh.ch/de/

[11] https://bullinger-stiftung.ch/

[12] https://www.bullinger-digital.ch/about

In the Bullinger corpus, training data is collected from different sources and also generated and translated manually. The language stage of these sources ranges from Classical Latin texts to modern day Vatican publications [Volk et al., 2022a]. Even though the training data covers Latin from a wider range than only 16th century, Fischer et al. [2022] find more data generally yields better results and systems get more robust. From the above mentioned OPUS Corpora, Fischer et al. [2022] use the two largest data sets for the training data of the Bullinger corpus, namely the Wikimatrix Corpus and the bible-uedin Corpus. The Wikimatrix Corpus contains 17,000 automatically aligned sentence pairs and was created by Facebook Research. On the other hand, the bible-uedin Corpus created by Christodouloupoulos and Steedman [2015] consists of the translation from Latin of the bible. 30,000 sentence pairs are included in the training data of the Bullinger corpus.

A part of the training data also consists of manually translated sentences. A small number of sentences of the Bullinger collection is translated by a scholar of the Swiss Reformation Studies Institute at the beginning of the project. This small number of translated sentences is later used as the primary test set. Additional manual translations by the Swiss Reformation Studies Institute have periodically been added to the training data later on [Fischer et al., 2022]. While 154 segments is a small number of manual translations, these high-quality translations of in-domain data are of very high value, as for low-resource languages, NMT systems usually work better when the quality is high and noise can be avoided [Magueresse et al., 2020]. Fischer et al. [2022] also collect a large part of their training data from different websites. The three main websites used were the official website of the Vatican[13], the weekly news summary of Vatican News[14], and the Library of the Church Fathers[15].

The translations of the official Vatican website include different scriptures from the Apostolic Constitutions, Catholic Catechisms, as well as constitutions, decrees and declarations of the Second Vatican Council. The added data consists of 60,589 quasi-parallel sentence pairs [Fischer et al., 2022]. Vatican News creates a news summary which is published weekly in Latin and German since 2004. Out of the entries, additional 6,139 sentences are added to the training data. Even though modern Latin is not entirely identical to the target domain, the high quality of the close translations are extremely valuable for the training data [Fischer et al., 2022].

From the Library of the Church Fathers, 21,573 parallel segments were added to the training data. This library consists of a collection of ancient Christian literature

[13]https://www.vatican.va

[14]https://www.vaticannews.va/de.html

[15]https://bkv.unifr.ch/de

with German translations, containing authors such as Hieronymus, Ambrosius, or Augustinus. Fischer et al. [2022] crawl all Latin source texts which have a German translation. With an average of 40 German tokens, the sentence length is remarkably high. Another 35,620 parallel segments can be collected from the Biblia Vulgata, a Latin translation of the Bible from the 4th century translated into German in the 1830s [Fischer et al., 2022].

Besides this additionally collected data through web crawling, English translations of Classical Latin can be used from the Perseus Digital Library [Clérice et al., 2022]. After downloading all text from their git repository[16], English-Latin sentence pairs are mined and English sentences are then translated into German via DeepL[17] in order to create a Latin-German parallel corpus. Additional 14,870 sentence pairs were added from the Perseus Digital Library [Fischer et al., 2022].

A part of the data is added from transcriptions and translations from other projects which used letters of the Bullinger correspondence, namely from the Zurich Letters [Robinson, 1846], the Blarer Correspondence [Schiess and Badische Historische Kommission, 1908], and the regests. Fischer et al. [2022] scan these letters and apply an OCR software to digitize the text.

1,825 English-Latin sentence pairs are collected from Zurich Letters [Robinson, 1846], which consist of the correspondence between different Swiss reformers, among them Bullinger, with English Bishops. The letters, which are available in Latin and English, are aligned and the English sentences are again translated into German with DeepL. This edition is valuable for the training data as it is identical to the target domain [Fischer et al., 2022].

The Blarer Correspondence [Schiess and Badische Historische Kommission, 1908] contains the correspondence between Bullinger and the Blarer brothers Ambrosius and Thomas. As the German translations of these letters are merely summaries, GoogleTranslate[18] is used to translate the German sentences of the letters into Latin, since the quality of GoogleTranslate has been improved recently. This approach was based on the idea of backtranslation [Sennrich et al., 2016]. These additional 2,868 German segments with Latin translations were marked with a special symbol to highlight that these sentences are backtranslations, as they tend to be erroneous.

Regests are German summaries of the content preceding the already edited Bullinger letters. 24,188 segments can be added to the training data by using the regests and

[16]https://github.com/PerseusDL

[17]https://www.deepl.com

[18]https://translate.google.com

again their backtranslations with GoogleTranslate. Even though the use of second person singular from the letters is replaced by third person use in the summaries, using the regests guarantees that the model encounters the names of most of Bullinger's correspondents, and covers most of the other named entities and specific vocabulary [Fischer et al., 2022].

5.2 The XML Structure

The data from the Bullinger corpus comes with a specific XML structure which is the same for each letter. The Extensible Markup Language[19] (XML) is a markup language and file format. It us used for storing, transmitting, and reconstructing arbitrary data. XML is, of many current formats of annotation, the format which emerged to be the most widespread form of annotation [Gries and Berez, 2017].

As described on the Website of the Bullinger Digital Project[20], the letters are organized by their respective editions. Each edition directory contains a number of letters with the corresponding edition number and each letter is represented in one XML file. Within the XML file, the structure is the same for each letter: It contains the metadata, the regest of the letter, as well as the transcription of the letter content itself [Volk et al., 2022b].

The metadata normally contain information such as the letter's source and its references, its date, numbers of corresponding scanned pages, the sender and addressee of the letter, and languages contained in the letter. The "letter" part of the XML file contains one sentence per line, each containing an attribute specifying in which language the sentence is written. Besides the 75% of the letters written in Latin, there are another 25% written in Early High German, while the languages may as well appear together in one letter [Volk et al., 2022b]. This language attribute comes into play during the extraction of the long sentences, as will be described in Section 7.2.

[19] https://www.w3.org/XML/

[20] https://www.bullinger-digital.ch/

6 Tools and Resources

Below, I introduce and describe the tools and resources used in this master's thesis. I do the programming with the Python Programming Language[1]. For the POS-tagging, I use two different models: The CLTK[2] POS-tagger and the POS-tagger by UDPipe[3]. The Neural Machine Translation of the project uses the Transformer[4] architecture and the SOCKEYE[5] framework.

6.1 The Python Programming Language

Python is a programming language with efficient high-level data structures. These high-level data types allow the expression of complex operations in a single statement. Statements are grouped by indentation instead of beginning and ending brackets, as it is often the case in other programming languages such as Java for example. Besides, the declarations of variables or arguments are not necessary in Python [Python Software Foundation, 2001-2022].

Python is an interpreted programming language, in which the interpreter reads and executes the source code instead of a direct translation by the target machine, as it is the case for compiled programming languages. Python also has a simple approach to object-oriented programming. With this approach to object-oriented programming, while some classes in Python are pre-defined, it is also possible to define new classes and create instances of them. With the definition of a class, the operations that can be performed on it are specified [Python Software Foundation, 2001-2022].

The Python programming language is known for its readable syntax and dynamic typing, which makes it useful for scripting and rapid application development. The Python interpreter is easily extensible with new functions and data types that are

[1] https://www.python.org/about/

[2] https://docs.cltk.org/en/latest/

[3] https://lindat.mff.cuni.cz/services/udpipe/run.php

[4] https://huggingface.co/docs/transformers/index

[5] https://github.com/awslabs/sockeye

implemented in C, C++, or in other languages that can be called from C. When writing scripts in Python, i.e. writing longer programs in a text editor to prepare the input for the interpreter and running it with that file as input, it is possible to split content into several files. These definitions from one module can be imported into other modules to reuse functions from other scripts. A module is, per definition, a file that contains Python definitions and statements. The name of the file is the module name with an appended suffix `.py` [Python Software Foundation, 2001-2022].

Besides modules, it is also possible to import packages and libraries. A package is a way of structuring Python's module namespace and groups together a collection of modules. Packages use "dotted module names". The import of a method from the multi-module package NumPy may thus be `from numpy.random import default_rng`, for example. As a simplification, packages can be thought of as directories in a file system, while modules are files within directories [Python Software Foundation, 2001-2022].

A library is also defined as a collection of related modules and packages which are grouped together, but with the specific goal to use them in a program or another library. An example for a library is the Matplotlib library, which is a standard library for the generation of data visualizations in Python [Python Software Foundation, 2001-2022].

In this thesis, I use Python for all the different steps of the splitting process. In a combination of different scripts, I use the programming language for the extraction of long Latin sentences from the XML data, for the POS-tagging step, and also for the splitting itself. I describe these steps in more detail in Section 7.

6.2 POS-Tagging Models

In this thesis, I use two different POS-tagging models to test how well they work on the Bullinger letters with 16th century epistolary Latin. The CLTK POS-tagger[6] and the UDPipe POS-tagger[7] will be described in the following sections.

[6]https://docs.cltk.org/en/latest/

[7]https://lindat.mff.cuni.cz/services/udpipe/run.php

6.2.1 CLTK

The Classical Language Toolkit (CLTK) is a Python library and open-source framework. It offers Natural Language Processing for pre-modern languages [Johnson et al., 2014–2021]. CLTK was founded in 2014, supporting NLP for historical languages. It addresses the need for complete text analysis pipelines for less-resourced historical languages such as Greek and Latin. The advantage of CLTK is that NLP tasks such as tokenization, lemmatization, part-of-speech tagging and related morphological analysis can be performed without resorting to external tools, web applications, or web services [Burns, 2019].

Pipelines which are pre-configured are available for 19 languages, among them Classical Chinese, Gothic, Hindi, Middle High German, and Latin. The CLTK library comes with a few native data types, namely `Word`, `Sentence`, `Doc`, `Process`, and `Pipeline` [Johnson et al., 2014–2021].

First of all, the `Process` data type takes and returns a `Document`, also called `Doc`. A process does some specific information processing within the `Doc` and annotates each `Word` object at `Doc.words`. The `Word` data type contains all processed information for each word token. It has several attributes, such as `Word.lemma`, `Word.pos`, or `Word.embedding`. The data is added to each `Word` by a process. A `Sentence` data type, on the other hand, contains the sentence embeddings, which consist of a weighted average of these word embeddings of the sentence [Johnson et al., 2014–2021].

The `Doc` data type contains the original input string to `NLP().analyze()`, called `Doc.raw`, which is the command used when processing data to obtain analyzed output data. The `Doc` data type also contains `Doc.words`. This is a list of `Word` objects which is the input and output of each `Process` and as well the final output of the `NLP()` function. And finally, the `Pipeline` data type contains a list of `Process` objects called `Pipeline.processes`. For some languages, predefined pipelines are available. For all languages, however, custom pipelines can be created [Johnson et al., 2014–2021]. Details about the usage of the CLTK library in this thesis will be discussed in Section 7.3.

6.2.2 UDPipe

UDPipe 2 is a trainable pipeline which can perform sentence segmentation, tokenization, POS-tagging, lemmatization and dependency parsing as well. It can be trained on annotated data in CoNLL-U format, but already trained models are available for

almost all Universal Dependencies[8] treebanks [Straka, 2018].

The UDPipe model makes use of an artificial neural network with a single joint model for the processes of POS-tagging, lemmatization and dependency parsing. It is trained on the CoNLL-U training data as well as pretrained word embeddings. It can be freely used for non-commercial purposes and provides annotation models for more than 50 languages, including Latin. Many of these languages are non-Indo-European, such as Arabic, Indonesian, or Irish [Straka, 2018].

While UDPipe is available as a downloadable program compatible with Linux, Windows and OS X, it can as well be used as a web application. When using UDPipe as a downloaded program, the desired Universal Dependencies language models need to be downloaded as well. Besides that, it is also usable as library in programming languages such as C++, Python, Perl, R, Java, C# [Straka, 2018].

When using the web service, the language in one of the three training models needs to be selected and the input text to annotate has to be provided as raw text or as a file. The model can annotate Part-of-Speech labels as well as more complex sets of grammatical features, such as case, person, gender, and tense to each individual word [Straka, 2018]. More information on the models used in this thesis will be provided in sections 7.3 and 8.1.

6.3 Neural Machine Translation Framework

In the experiments on the Bullinger Corpus, Fischer et al. [2022] implement the Transformer architecture in its base configuration by Vaswani et al. [2017]. As framework, they use the SOCKEYE[9] framework by Hieber et al. [2017]. The SOCKEYE framework is an open-source sequence-to-sequence toolkit for NMT. It serves as an experimental platform for researchers as well as a framework for training and applying models. As mentioned in Section 3.1.1, the three most prominent encoder-decoder architectures are attentional recurrent neural networks, convolutional networks, and self-attentional transformers. The toolkit is written in Python and is built on Apache MXNET[10] [Chen et al., 2015], and thus offers scalable training and inference for all of these three architectures [Hieber et al., 2017]. A further advantage of SOCKEYE is the support of a wide range of optimizers, as well as normalization and regularization techniques and recent inference improvements. A

[8] https://universaldependencies.org/

[9] https://github.com/awslabs/sockeye

[10] https://mxnet.incubator.apache.org/

user can customize different model settings, incorporate new designs, or run existing standard training recipes [Hieber et al., 2017].

While the presence of many independent toolkits brings diversity to the field, it does not contribute to the possibility to compare architectural and algorithmic improvements. SOCKEYE addresses the need for an NMT toolkit which contains all the best ideas from current literature and findings in the fast changing field of NMT. As a lot of engineering is required to achieve "production-ready" performance and the optimum between translation quality and computational efficiency, it allows for hyper-parameter tuning, architecture modifications, and empirically effective heuristics [Hieber et al., 2017].

7 Methods

This chapter describes my methods for all the steps of the sentence splitting task. I start with the extraction of long sentences from the corpus. With the extracted sentences, I test different POS-taggers and with the gold standard of the POS-tagging step, I experiment with different splitting approaches to create shorter sequences to translate. The code used is visible in the dedicated Git repository[1]. In order to facilitate the running of the scripts, shell scripts are provided occasionally. A short description of the pipeline is provided in the corresponding `README` file.

7.1 The Corpus Structure

As pointed out in Section 5.1, the complete corpus of the Bullinger letters includes more or less 12'000 letters from the 16th century [Fischer et al., 2022]. The final digital version of all Latin texts from the Bullinger Digital Project and their Machine Translations into German are available digitally[2] [Volk et al., 2022b]. During the digitization process, the letters were brought into an XML format.

To achieve this format, as described in Section 5.2, the letters of the Bullinger corpus were scanned by Fischer et al. [2022] and an OCR software was applied for the digitization of the text. The digitized letters are formatted in an XML structure organized in different directories according to their edition. Each XML file containing one letter consists of the metadata, the regest, and the transcription of the letter [Volk et al., 2022b].

The following example in Figure 0 shows the structure of such an XML file as well as the letter itself, which is structured with one sentence per line.

[1] https://github.com/sabrinabraendle/ma-bullinger-split.git

[2] https://www.bullinger-digital.ch/

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<file>
 <metadata id="10016" id_hbbw="1" id_irg_sort="1" id_sort="100010">
  ...
 </metadata>
 <regest fk_bibliography="1" nr="1" page="45">
  ...
 </regest>
 <letter lang="la" edition="HBBW" vol="1"
 fk_bibliography="1" nr="1" page="45">
  ...
 </letter>
  <footnotes fk_bibliography="1" nr="1">
  ...
 </footnotes>
</file>
-----------------------------------------------------------------------

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<file>
 ...
 <letter lang="la" edition="HBBW" vol="1"
  fk_bibliography="1" nr="1" page="45">
  <div ref_regest="1">
   <p>
    <s lang="la" state="auto">...</s>
   </p>
   <p>
    <s lang="la" state="auto">Mitto tibi, mi ... </s>
    <s lang="la" state="auto">Dedico eas nomini tuo, ... </s>
    ...
   </p>
   <p>
    <s lang="la" state="auto">Ex academia nostra ... </s>
   </p>
   <p>
    <s lang="la" state="auto">Henrichus Bull[ingerus], ... </s>
   </p>
  </div>
 </letter>
 ...
</file>
```

Figure 0: XML Letter Structure in the Bullinger Corpus

7.2 Sentence Extraction

In the sentence extraction step, I use the script `extraction.py` to search for the Latin sentences in the "letter" part of each XML file, which make up 75% of all letters. Each letter is structured in the XML file with one sentence per line, whereas the language is marked by an attribute such as `lang="la"`. In a first step, I loop over all `.xml` files and collect all sentences in Latin and store them in lists if they have the sentence lengths I want to investigate. Punctuation symbols are counted as tokens as well in this task. The three different length categories are defined as

1. 15-19 tokens,
2. 20-24 tokens,
3. and 25 or more tokens.

Once all Latin sentences of the desired lengths are collected in the three lists, 200 sentences are randomly sampled from each category. For each length category, a text file is created containing the sentence and one sentence of the corresponding length per line, with the corresponding edition one line above the sentence.

7.2.1 Length Categories

In the test set of the Bullinger corpus, shorter sentence lengths are more frequent. The train set contains a larger number of longer sentences than the test set. However, most sentences of the train set are shorter than 30 tokens. A visualization of the frequency of different sentence lengths in the test and train set is shown in Figure 1.

The definition of "long" sentences generally depends on the language and the corpus. The literature suggests that models specifically show a performance drop in translating sentences that are longer than those in the training data [Kondo et al., 2021; Neishi and Yoshinaga, 2019]. Berrichi and Mazroui [2021] find in their study that the translation quality drops significantly when the sentence length is greater than 30 words, which makes them define "long" sentences as sentences longer than 30 words. Kuang and Xiong [2016] perform their experiments on sentences longer than 30 words, too. In this thesis, I intend to observe changes in the translation quality when sentences of all three length categories are split into shorter sequences.

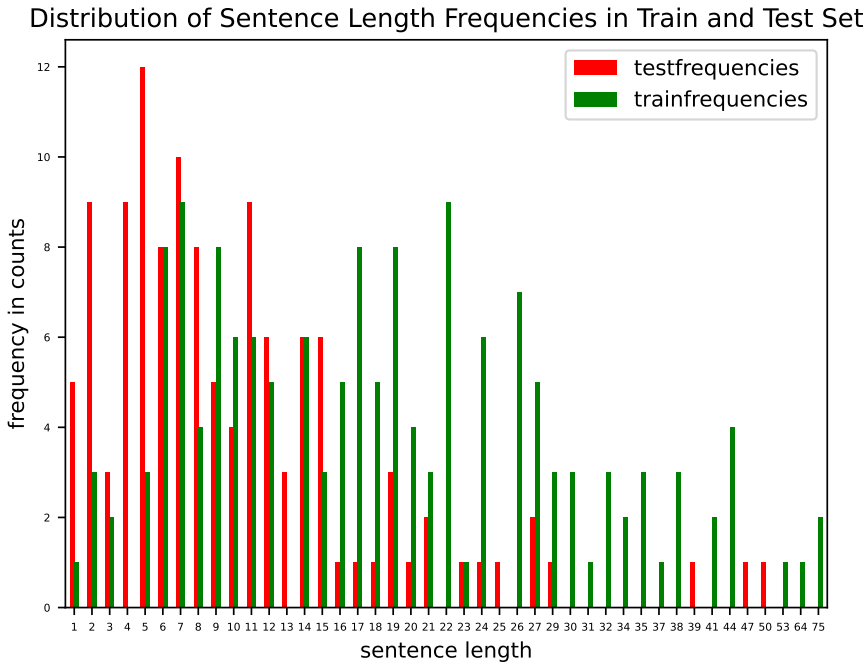

Figure 1: Sentence Length Distribution of the Train and Test Set

7.2.2 Extraction Challenges

Challenges when extracting Latin sentences include the parsing of the `.xml` files and the corresponding work with the Python package `lxml`[3]. It is sensible to parse the `.xml` files with the `os.walk()` command, which traverses the root directory and thus accesses the directories' files inside the root.

Furthermore, the command `"".join(element.xpath("text()"))` makes sure the complete text from a sentence is extracted, as parts or tokens in the text are marked with special tags, such as `<entity id="204" type="pers">Wernhere</entity>` for named entities. The complete data and code is in the dedicated Git repository[4].

[3]https://lxml.de/

[4]https://github.com/sabrinabraendle/ma-bullinger-split.git

7.3 POS-Tagging

In this section, I discuss the process of POS-tagging long Latin sentences as well as the elaboration of a POS-tagging gold standard in order to compare different POS-taggers. While POS-tagging annotation for low-resource languages can be done manually and then be extended to greater amounts of data with automatic methods, which is usually more efficient [Zennaki et al., 2015], this thesis uses the already existing POS-taggers from the CLTK[5] and from UDPipe[6].

7.3.1 Sentence Formats for the POS-Tagging Models

After extracting long sentences of the three length categories, the CLTK POS-tagger can be directly applied to the extracted sentence files with a one-sentence-per-line format by the installed `cltk` package. For this step, I use the script `cltk_postagger.py`. The output of the Latin NLP model (`NLP(language="lat")`) can be linguistically analyzed by the standard method `.analyze(text=input)`. By using the `.tokens` and the `.pos` methods, the tokens and their corresponding POS-tag are written to an output file.

UDPipe offers a web service for tagging, where the input can be provided as raw text or as complete input files [Straka, 2018]. In order to obtain the desired output format, the extracted files need to be formatted differently using the script `udpipe_formatter.py`. The input file can be processed with one sentence per line, and the tokens separated by a tab space. On the web service, the models used are the three UD 2.10 model versions "latin-ittb-2.10-220711", "latin-proiel-2.10-220711", and "latin-perseus-2.10-220711". Further web service settings used in this thesis include the horizontal input format and the tokenizer which takes pre-segmented input. The horizontal format specifies that each sentence is on a separate line and its tokens are separated by spaces. The pre-segmentation tokenizer takes input which is assumed to be already segmented. Each sentence is expected to be tokenized with respect to sentence breaks and on a separate line [Straka, 2018].

7.3.2 POS-Tagging Models

In this section, I provide information on the different POS-tagging models and their functionalities with respect to the Bullinger corpus data and the POS-tagging step

[5]https://docs.cltk.org/en/latest/

[6]https://lindat.mff.cuni.cz/services/udpipe/run.php

of the splitting pipeline.

7.3.2.1 CLTK

The Python library Classical Language Toolkit (CLTK) is an NLP framework dedicated to the processing of pre-modern languages. It is characterized by a modular processing pipeline and provides models and entire pipelines for approximately 20 languages. The software architecture contains diverse algorithmic structures [Johnson et al., 2021]. For morphological parsing in the case of Latin, the CLTK relies on Stanza[7]. The morphological taggers output values such as the word class and grammatical categories, which are then normalized to the CLTK data types described in Section 6.2.1 that model the annotations of the Universal Dependencies project [Johnson et al., 2021].

Stanza is based on neural network components, with modules built on top of the PyTorch[8] library [Qi et al., 2020]. Stanza features a neural network pipeline and has pretrained neural models for 70 languages. For POS-tagging, Stanza uses a bidirectional long short-term memory network (Bi-LSTM) as the basic architecture. The POS module labels the words with their universal POS (UPOS[9]) tags as well as treebank-specific POS (XPOS) tags [Qi et al., 2020].

Unfortunately, it is not yet possible to report any formal evaluations of CLTK's models' accuracies, as most parts of the pipelines wrap models trained by upstream projects, such as Stanza. While accuracy reports of these projects are available respective to their training sets, they lack evaluations against outside benchmarks which do not yet exist for pre-modern languages [Johnson et al., 2021].

Thus, while the CLTK is predominantly created for the use on classical languages, its neural architecture and relatively high amount of training data makes it an interesting toolkit for this thesis, even though the training data might not entirely cover the domain of 16$^{\text{th}}$ century epistolary Latin [Fischer et al., 2022].

7.3.2.2 UDPIPE

UDPipe 2 provides already trained models for almost all Universal Dependencies[10] treebanks. UDPipe uses a single joint model for the processes of POS-tagging,

[7] https://stanfordnlp.github.io/stanza/

[8] https://pytorch.org/

[9] https://universaldependencies.org/u/pos/

[10] https://universaldependencies.org/

lemmatization and dependency parsing. For POS-tagging, UDPipe applies a multilayer bidirectional LSTM to process embedded words, just as the CLTK (see Section 7.3.2.1) [Straka, 2018]. The 2.10 models, which are used in this thesis, additionally implement multilingual BERT and RobeCzech to provide contextualized word embeddings. These models are based on Universal Dependencies 2.10 treebanks [Straka et al., 2021].

The three UDPipe models used in this thesis include the ITTB model, the PERSEUS model, and the PROIEL model. Performances are available for these three models. For POS-tagging, the models achieve the following results based on F1 scores and measured against the testing portion of the data evaluated against raw text [Straka et al., 2021].

UDPipe Model	performance	training data size
latin-ittb-ud-2.10-220711	98.91	450'515 tokens
latin-perseus-ud-2.10-220711	91.83	200'163 tokens
latin-proiels-ud-2.10-220711	96.69	29'138 tokens

Table 1: Specifications of the used UDPipe models [Straka et al., 2021]

The ITTB model, with the largest training data size of the three models, is trained on the ITTB treebank[11]. It contains Latin data from the Index Thomisticus Treebank. The data comes from the Index Thomisticus corpus by Roberto Busa. This corpus contains the complete work by Thomas Aquinas, who lived from 1225 until 1274, and 61 other authors related to Thomas. The texts are written in Medieval Latin [Cecchini et al., 2018].

The PROIEL model is trained on Latin data from the PROIEL treebank[12]. It covers most of the Vulgate New Testament translations, as well as selections from Caesar's Gallic War, from Cicero's Letters to Atticus, from the first book of Cicero's De officiis, and from Palladius' Opus Agriculturae [Cecchini et al., 2018].

Finally, the PERSEUS model is trained on the Universal Dependencies PERSEUS Latin Treebank[13]. This treebank consists of an automatic conversion of a selection of passages from the Latin Dependency Treebank 2.1. The original data were morphologically and syntactically annotated in a semi-automatic way. The Morpheus morphological analyzer is used for the morphological annotation and lemmatization. The syntactic annotation is done manually. The Latin treebank contains parts of

[11]https://universaldependencies.org/treebanks/la_ittb/index.html

[12]https://universaldependencies.org/treebanks/la_proiel/index.html

[13]https://universaldependencies.org/treebanks/la_perseus/index.html

works from Augustus, Cicero, Vergil, Ovid, Petronius, Tacitus, and a few other authors [Cecchini et al., 2018].

Even though the training data of the PROIEL model and the PERSEUS model date back to the 4th century or earlier, the two models are taken into account for the POS-tagging. And, while Thomas Aquinas lived in the 13th rather than the 16th century, the ITTB model covers Medieval Latin, which is promising for the goal of this thesis.

7.3.3 The Tagsets

The UPOS tagset of the Universal Dependencies Project includes all of the POS-tags in table 2.

UPOS	word class
ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary
CCONJ	coordinating conjunction
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	other

Table 2: UPOS Tagset[14]

While the CLTK uses the features defined by version 2 of the Universal Dependencies project [Johnson et al., 2021], the three UDPipe treebanks lack some of the tags. None of the three uses the SYM tag. Additionally, the ITTB treebank does not use INTJ, the PROIEL does not use PART and PUNCT, and the PERSEUS treebank has no AUX, DET, or PART tag [Cecchini et al., 2018]. For the gold standard in

this thesis, the UPOS tagset is used, with some specific adaptations and strategies for the comparison described in Section 7.3.4.

7.3.4 Tagging Decisions

For the gold standard of the POS-tagged files, the tags DET and PART were mapped to PRON and ADV, respectively. As not all of the models make use of the DET tag, I decide to count this tag as pronoun, as in attributive position all pronouns become DET. By the models which use PART, only the word *non* is tagged with this POS-tag, which is why I replaced it with ADV in the gold standard. The evaluation of the different models with respect to the gold standard is done with the script `accuracies.py`.

As not all models use the AUX tag, I replace it with VERB. Uncategorizable words such as non-Latin words are tagged with the X tag. Tokens tagged with X in the gold standard are excluded from the evaluation, as they are not relevant for the research questions of this thesis.

7.4 Sentence Splitting

I implement the Data Augmentation method of splitting long sentences into smaller sequences on POS-tagged files obtained from the gold standardization using the script `splits.py`. For the splitting, I examined different splitting rules with different positions for the split. These different splitting rules are pointed out in Section 7.4.1.

7.4.1 Splitting Rules

As described in Section 2.1.2, syntactical discontinuities are frequent in Latin. Tokens functioning as subjects, objects and predicates do not have any fixed positions. Nevertheless, it is possible to group tokens in a Latin sentence into clauses with a complete semantic prediction, as every grammatically correctly formed sentence of Latin consists of one or more clauses. Since a sentence that consists of several clauses, which is often the case for long sentences, is syntactically composed through coordination or subordination, it seems sensible to consider punctuation markers or conjunctions as splitting positions [Horrocks, 2011].

7.4.1.1 Splitting at Punctuation Markers

Similar to Su et al. [2018] and Zhang and Matsumoto [2019], I implement splitting at punctuation markers. This rule is based on the assumption that most clauses are separated by commas or similar tokens. Using this splitting rule, I inspected three translation scenarios:

1. Translating all clauses separately.
2. Putting the first and last sentence part back together and translate the clause composition.
3. Only splitting the sentence at a punctuation marker in the middle of the sentence, and translating these two sentence halves.

I implement the segmentation by first splitting at every punctuation marker. Once I obtain the splits, I translate them with the Bullinger NMT system. An example of the first splitting method can be seen in 7.1. The second splitting method consists of adding the first and the last part of each sentence, following the observation that a subject often occurs in the beginning of a sentence, and a predicate rather in the end. Example 7.2 shows a sentence split in half and its corresponding translation. The third experiment includes identifying punctuation markers and only splitting the sentence in the middle, irrespective of the number of punctuation-separated clauses it contains. An example is shown in 7.3.

(7.1) *Nos erimus fortes in domino, licet diffugiant multi iam*
We will.be brave in the.Lord, even.though flee many already
ex urbe nostra et suas dimittant pecunias.
out city our and their transfer money

split-MT-de. 'Wir werden stark sein im Herrn. , auch wenn viele aus unserer Stadt fliehen und ihr Geld vergeben .'

MT-de. 'Wir werden tapfer im Herrn sein, obwohl viele aus unserer Stadt fliehen und ihr Geld entlassen.'

ref-de. 'Wir werden tapfer sein im Herrn, auch wenn schon viele aus unserer Stadt fliehen und ihr Geld übergeben.'

en. 'We will be brave in the Lord, even though many are already fleeing our city and transferring their money.'

[accessed 10th October 2022]

(7.2) *Utinam familiam meam possem dimittere ad vos, si altius*
If.only family my I.could send to you, in.case deeper
ingruerint mala, quo fortius agere possem!
increase the.evils, in.order.to better act be.able.to!

split-MT-de. 'Ich wünschte, ich könnte meine Familie zu euch schicken. , damit ich stärker handeln kann!'

MT-de. 'Ich wünschte, ich könnte meine Familie zu euch gehen lassen, wenn die Übel vertieft werden, damit ich stärker handeln könnte!'

ref-de. 'Wenn ich doch meine Familie zu euch senden könnte, falls die Übel weiter zunehmen, um allein besser agieren zu können!'

en. 'If only I could send my family to you in case the evils continue to increase, in order to be able to act better on my own!'

[accessed 10th October 2022]

(7.3)

Quam	*vera*	*sint,*	*nescio;*	*sunt*	*tamen,*	*qui*	*pro*
How	true	they.are,	I.don't.know;	there.are	however,	who	as
certissimis	*venditent.*						
certain	will.sell.						

split-MT-de. 'Wie wahr sie sind, weiss ich nicht. ; es gibt jedoch einige, die sie für die sichersten verkaufen.'

MT-de. 'Wie wahr sie sind, weiss ich nicht; aber es gibt einige, die sie für die Gewissheit verkaufen.'

ref-de. 'Wie wahr diese Nachrichten sind, weiss ich nicht; es gibt aber Leute, die sie als ganz sicher ausgeben.'

en. 'How true this news is, I don't know; but there are people who pass it off as quite certain.'

[accessed 10th October 2022]

7.4.1.2 Splitting at Conjunctions

Conjunctions are used to combine clauses in a sentence through coordination or subordination [Horrocks, 2011]. Similar to Berrichi and Mazroui [2021], who identify words that serve as links between two sentence segments and use them as semantic segmentation markers as splitting position, I use tokens with a conjunctional POS-tag CCONJ or SCONJ as splitting position.

With the SCONJ tags, I split the sentences before the token tagged with SCONJ and store the segments as new sequences. Additionally, I store a sequence by excluding the clause containing the SCONJ completely. The second splitting position is defined as the next punctuation marker. This segmentation is based on the assumption that the sentence still has meaning without the subordinating clause. Example 7.4 shows the splitting process at each SCONJ tag of a sentence, example 7.5 one of the extraction of the clause starting with SCONJ.

In the case of CCONJ tags, I implement the splitting of the sentences before each

CCONJ token as well. I then store all obtained segments as new sequences and translate these. An example for the splitting at CCONJ tokens is shown in example 7.6.

(7.4) *Si ille advenerit, fortassis negotia imperii*
When he will.have.arrived, perhaps affairs of.the.realm
tractabuntur.
will.be.dealt.with.

split-MT-de. 'Wenn er angekommen ist, Vielleicht werden die Angelegenheiten des Reiches behandelt.'

MT-de. 'Wenn er angekommen ist, wird er vielleicht mit den Angelegenheiten des Reiches befasst werden.'

ref-de. 'Wenn er angekommen sein wird, werden vielleicht die Reichsangelegenheiten behandelt werden.'

en. 'When he will have arrived, perhaps the affairs of the realm will be dealt with.'

[accessed 10th October 2022]

(7.5) *Accepit a Ioanne Blasio iuramentum eum aliud non*
Take from Johannes Blasius oath he anything.else not
scire quam episcopum atque comites illos viros esse probos.
know that bishop and followers his men be righteous.

split-MT-de. 'Es liess Johannes Blasius schwören, daß er nichts anderes wisse. Diesen Bischof und seine Gefolgsleute seien rechtschaffene Männer.'

MT-de. 'Es liess Johannes Blasius schwören, dass er den Bischof und seine Gefolgsleute für rechtschaffene Leute halte.'

ref-de. 'Es liess Johannes Blasius schwören, dass er nichts anderes wisse, als dass der Bischof und seine Gefolgsleute rechtschaffene Leute seien.'

en. 'It made Johannes Blasius swear that he knew nothing but that the bishop and his followers were righteous people.'

[accessed 10th October 2022]

(7.6) *De libris apud nos absolutis his nundinis nondum potui*
About books with us appeared this fair not.yet been.able.to
expiscari; sed hac hebdomada id efficiam.
find.out, but this week that I.will.do.

split-MT-de. 'Von den bei uns abgeschlossenen Büchern, die auf diesen Messen abgefasst wurden, konnte ich noch nicht herausfinden; Aber in dieser Woche werde ich es tun.'

MT-de. 'Was die bei uns abgeschlossenen Bücher angeht, so konnte ich noch nicht herausfinden, aber ich werde es in dieser Woche tun.'

ref-de. 'Über die zu dieser Messe bei uns erschienenen Bücher konnte ich noch nichts in Erfahrung bringen; aber in dieser Woche werde ich das tun.'

en. 'I have not yet been able to find out about the books published at this fair, but I will do so this week.'
[accessed 10th October 2022]

7.5 Neural Machine Translation of the Splits

In order to evaluate the translation quality with these splitting rules, I follow two different approaches:

1. Translating the clauses separately into German and recomposing them to sentences.
2. Adding the separately translated clauses to the training data.

These two procedures are explained more precisely in Section 7.5.1 and Section 7.5.2.

7.5.1 Recomposing translated Clauses into Sentences

After the automatic splitting of the POS-tagged sentences with the script `splits.py`, I translate the files with the separate split clauses using the webservice[15] of Bullinger Digital into German. The translated German clauses are then automatically recomposed to sentences using the script `splits2sents.py`.

In order to evaluate the quality of these recomposed sentences, I perform a manual qualitative analysis on the output of the third length category of 25 and more tokens, assessing the adequacy as well as the fluency of a translated and recomposed sentence with scores of 0 or 1. Adequacy estimates whether the output conveys the same meaning as the input sentence. Therefore adequacy is not given if parts of the message are lost, added, or distorted. Fluency, on the other hand, assesses whether the output is fluent in the target language, including grammatical correctness as well as idiomatic word choices.

7.5.2 Adding Clause Translations by GoogleTranslate to the Training Data

During the translation of shorter subsequences of the sentences, it became clear that the Bullinger NMT system has difficulties translating clauses that are not complete

[15] https://translate.bullinger-digital.ch/

sentences. It is plausible that, as the system is mainly trained on data containing complete sentences, it produces complete sentences on the target side as well. Since GoogleTranslate improved the translation quality for language pairs including Latin, GoogleTranslate was already used by Fischer et al. [2022] for backtranslations from German sentences of the Blarer letter into Latin. Therefore, I examine the translation of clauses with GoogleTranslate as well. The hypothesis is that the translation quality can be increased as the model is trained on more in-domain data. For this experiment, split sentences of four splitting rules are used: splitting at each punctuation marker, splitting at each CCONJ tag, splitting at each SCONJ tag and extracting this clause, and splitting at SCONJ tag. In order to implement GoogeTranslate, the API accessible for the Bullinger Digital Project is used. After translating, the clauses are added to the training data in order to evaluate whether this improves the model performance.

The first step in this approach is to extract all long sentences of the three lengths categories from the XML data corpus. I then split them based on the splitting rules and evaluate the translation quality after the application of the GoogleTranslate API. For the extraction of all long sentences, I use the script `extraction_all.py`. The script `remove_testsents.py` makes sure no test sentences are contained in the input file. The POS-tagging is applied on all extracted sentences by the script `traintest_google_postagger.py`. For the training, it is crucial to shuffle the input sentences, which is done by the script `traintest_google_shuffle.py`. After the shuffling, I apply the splitting with the script `traintest_google_splits.py`. The output split clauses are then translated using the GoogleTranslate API and are then added to the training data in order to train a model and evaluate its performance.

8 Results

This chapter describes the results of the POS-tagging and the splitting experiments. The results are manually inspected and include an error analysis as an evaluation of the applied methods.

8.1 POS-Tagging

In this section, I discuss the results from the application of the different POS-tagging models to the Latin sentences of the three length categories.

I make the following comparisons:

1. Overall accuracies of the POS-taggers
2. Accuracies per length category
3. Detailed accuracies per POS-tag

8.1.1 Evaluation of the POS-Tagging Results

The accuracies obtained from the gold standard shown in Figure 2 suggest that the CLTK POS-tagger achieves best results with a mean accuracy over all models and sentence length categories of 91.95%. The ITTB model reaches an overall accuracy of 86.19%, closely followed by the PERSEUS model with a mean accuracy of 85.24%. The lowest accuracy is achieved by the PROIEL model with 69.93%.

The sentence length seems to have a smaller impact on the result than the model choice, as there are only small differences in their overall accuracies. The accuracy over all models and tags is highest for the length category of 20-24 tokens with 84.14%. The category of 15-19 tokens follows with 83.02% and, unsurprisingly, the category of 25 and more tokens shows the lowest accuracy with 82.83%. However, the differences are small.

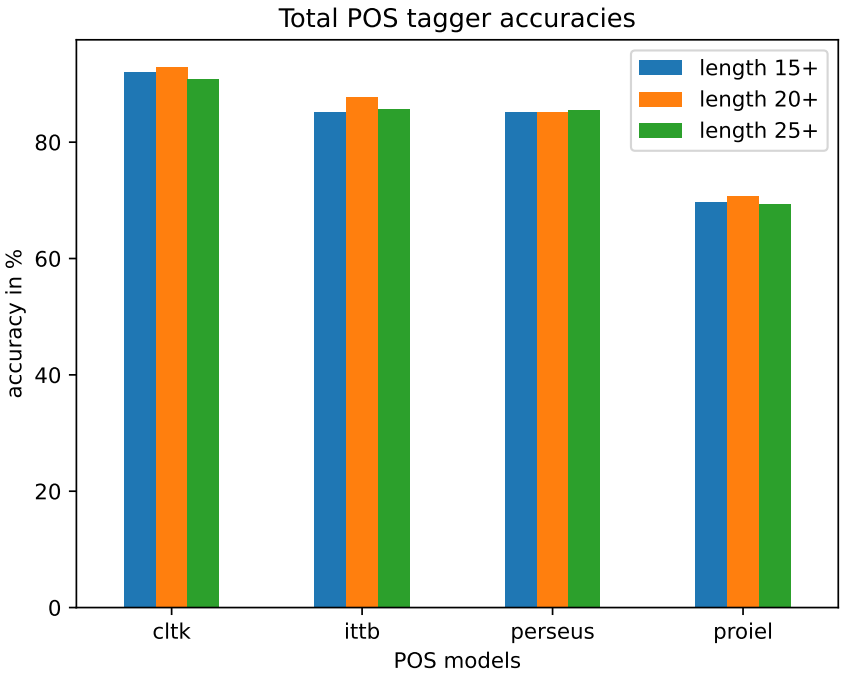

Figure 2: Overall accuracies of the four POS-taggers

The models perform very differently depending on specific POS-tags. While certain tags are almost consistently tagged correctly by all models, others are more problematic. As Figure 3, Figure 4, and Figure 5 show, the model performances with respect to different POS-tags look similar across the three length categories. Thus, the accuracy in relation to a POS-tag seems to depend on the model and less on the sentence length.

These performance differences between models in the accuracy of specific POS-tags are visible in Figure 6, for example. While the PROIEL model performs moderately well on PROPN tags across all sentence lengths with an average performance with 69.79%, both the ITTB and the CLTK model show low average performances with 30.14% and 19.74%, respectively. The PROPN POS-tag is not part of the tagset of the PERSEUS model. While the accuracies of the same model are not identical across sentence lengths, they stay on a similar level.

Observing the ADJ POS-tag, on the other hand, all models achieve good accuracies, whereas the ITTB model performs slightly worse than the other models with an average accuracy of 75.2%, which is shown in Figure 7.

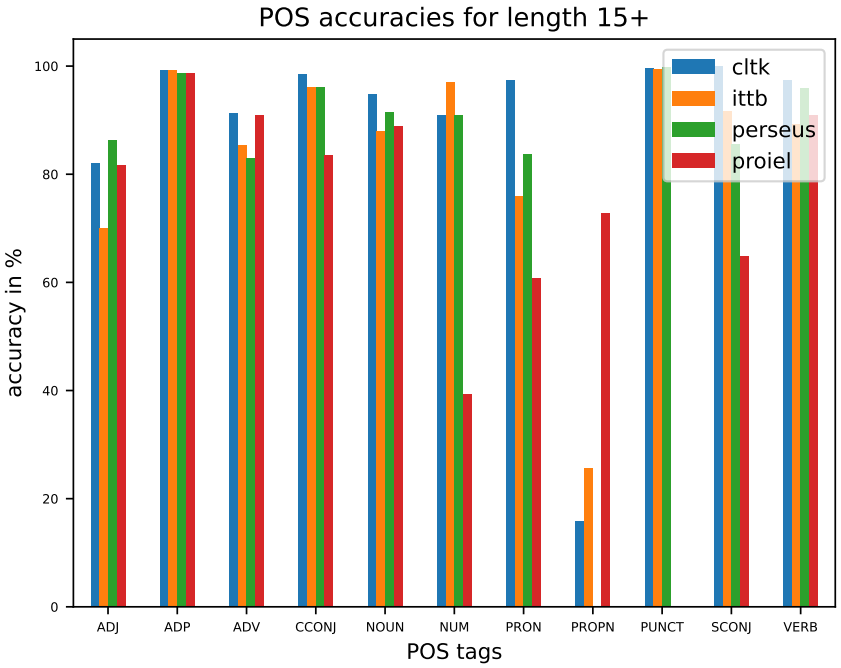

Figure 3: POS-tagger accuracies with sentence length 15-19 tokens

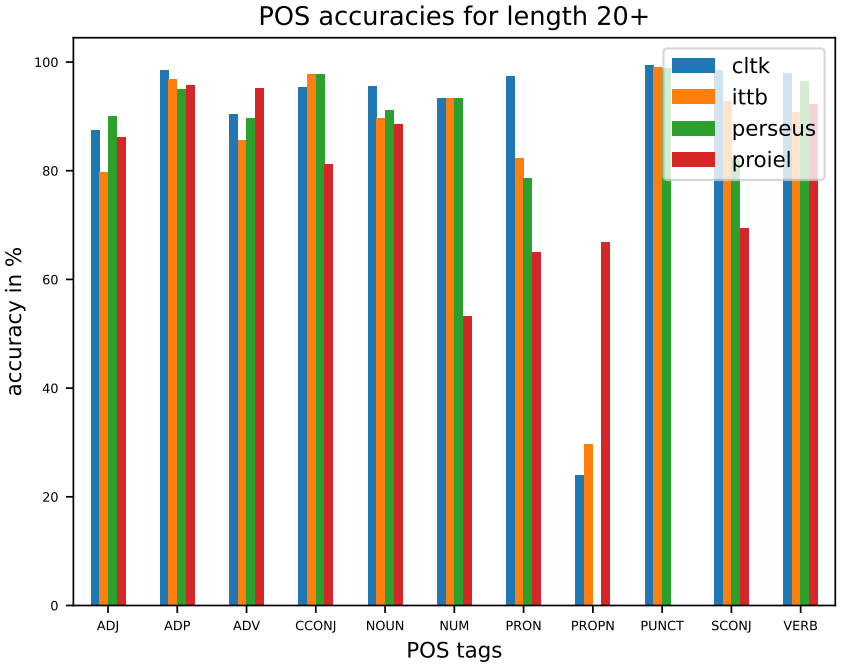

Figure 4: POS-tagger accuracies with sentence length 20-24 tokens

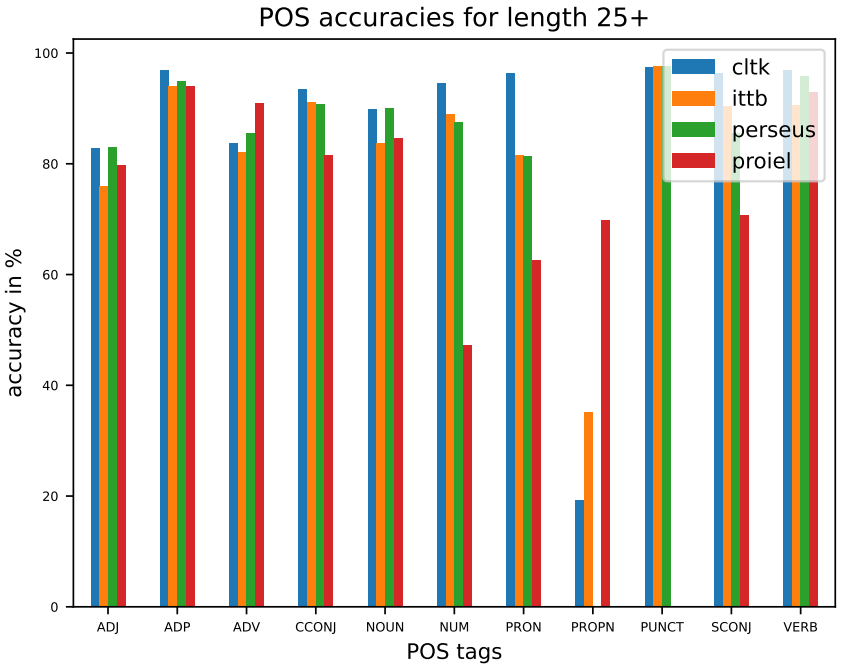

Figure 5: POS-tagger accuracies with sentence length 25 tokens or longer

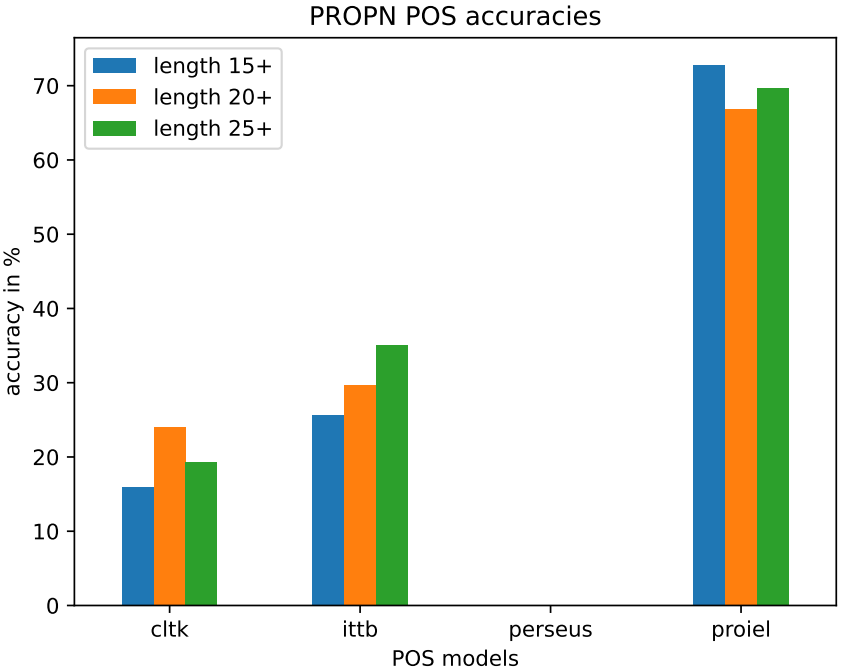

Figure 6: PROPN POS-tag accuracies across all models

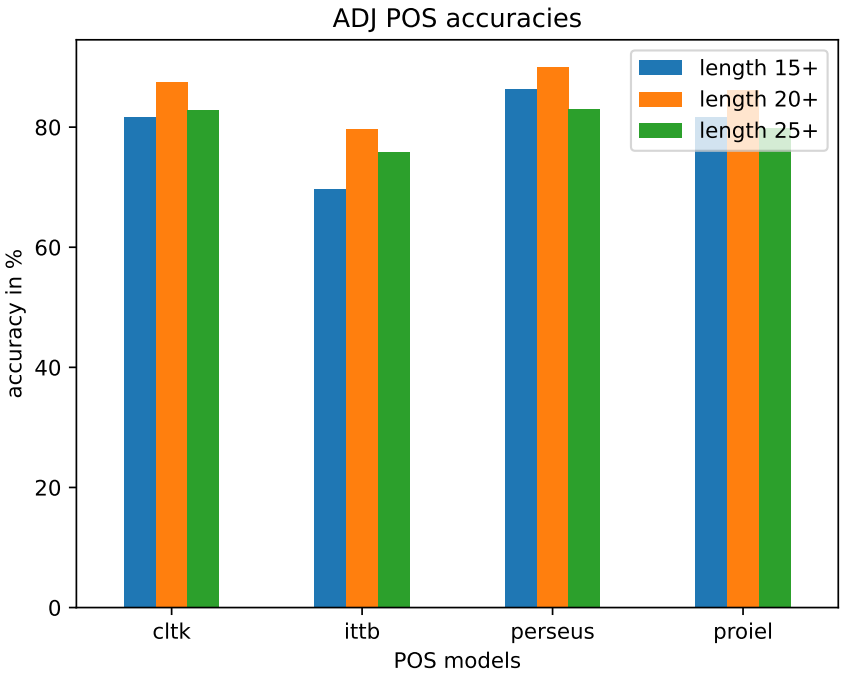

Figure 7: ADJ POS-tag accuracies across all models

A similar tendency can be seen in Figure 8 on the CCONJ tag. With an average over the sentence categories of 82.11%, the accuracy of the PROIEL model is clearly lower than that of the other models. The CLTK model shows the best performance with 95.82% accuracy on average. A similar result is achieved for the SCONJ POS-tag as shown in Figure 9. Again, the PROIEL model achieves the lowest average accuracy with 68.32%, and the CLTK model the highest one with 98.29%.

As the coordinating and the subordinating conjunctions are important POS-tags for the splitting step of this work, it is crucial to rely on a POS-tagger with good accuracies of these tags, the best of which being the CLTK tagger.

8.1.2 Error Analysis

For the error analysis, the script `pos_error_analysis.py` detects incorrectly tagged tokens and exports them into an analysis table. There are a few patterns which occur repeatedly for specific POS-tags, occasionally depending on the models. As the results are very similar for the different length categories, I present the error plots of the longest category and examples from the Bullinger corpus [Volk et al., 2022b].

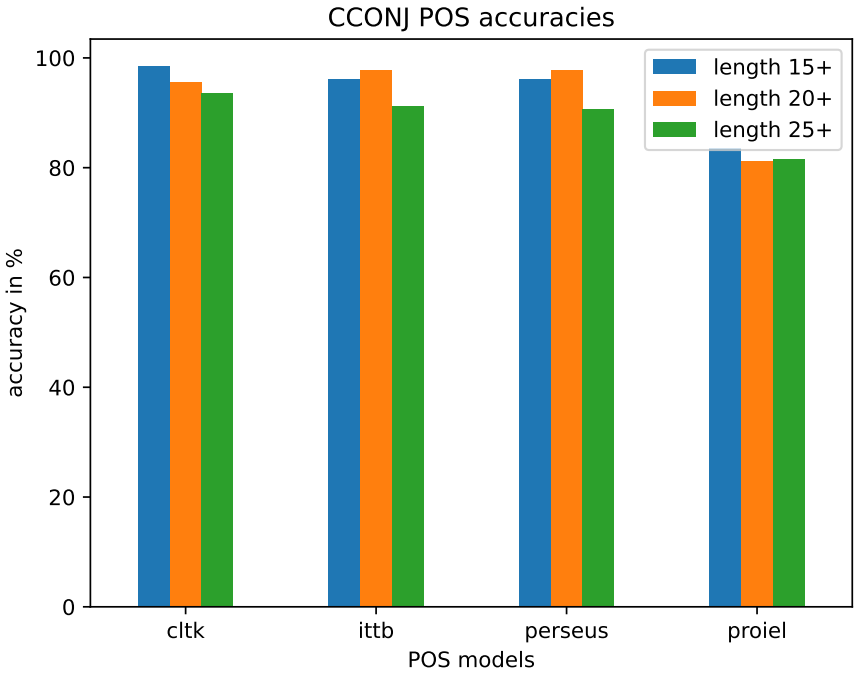

Figure 8: CCONJ POS-tag accuracies across all models

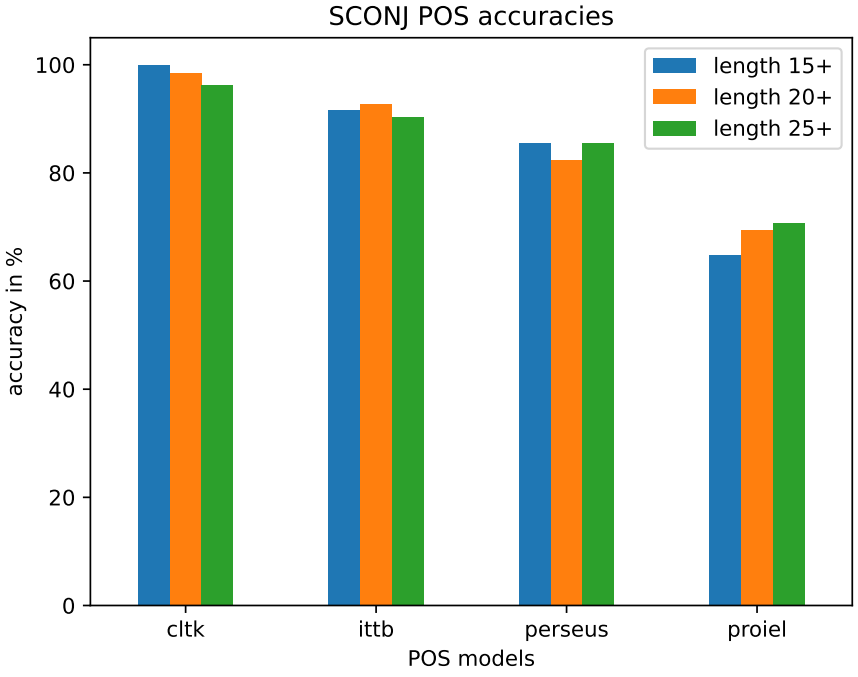

Figure 9: SCONJ POS-tag accuracies across all models

1. Incorrect annotation of PROPN

   In 42.3% of all cases across all models and sentence lengths, PROPN is mistaken for a NOUN, as shown in Figure 10 for the sentence length of 25 and more tokens. An example for such an error is shown in 8.1. Especially for rare tokens, it is plausible that all POS-taggers used can mistake proper nouns for nouns, since the difference between NOUN and PROPN is of a purely semantic rather than morphosyntactic nature. The lexical character of PROPN tags makes them anomalous in the UPOS scheme [Sprugnoli et al., 2022]. The CLTK model scores lowest on this POS-tag, while the PROIEL model scores best.

   (8.1) *S* . *Remitto nunc exemplum tuę illius modestissimę*
   NOUN PUNCT VERB ADV NOUN PRON PRON ADJ
   ad Philippum epistolę et gratiam habeo plus quam
   ADP PROPN NOUN CCONJ NOUN VERB ADV ADV
   maximam ,
   ADJ PUNCT
   de. 'Ich gebe dir jetzt ein Exemplar deines bescheidenen Briefes an Philipp zurück und danke dir mehr als möglich'
   en. 'I am now returning a copy of your humble letter to Philipp and thank you more than I can'
   [accessed 17th October 2022]

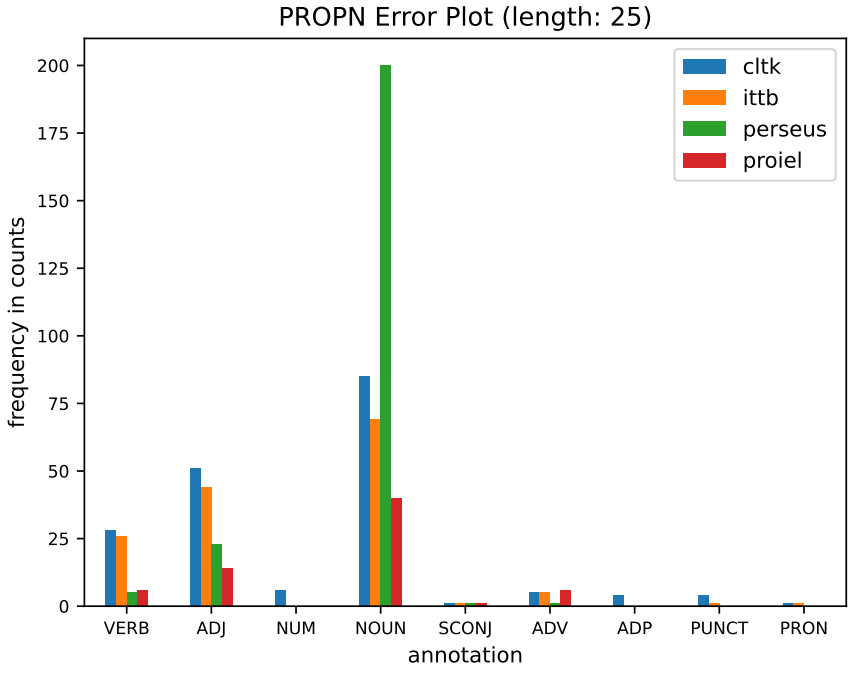

Figure 10: Error Plot for PROPN tag

2. Incorrect interpretation of ADJ and NOUN

   In most erroneous annotations for nouns, they are interpreted as adjectives and vice versa. This becomes obvious in Figure 11 and Figure 12. In Latin, adjectives and nouns or proper nouns almost completely overlap on their inflectional paradigms. Thus, a distinction based on formal criteria can incur in difficulties [Sprugnoli et al., 2022]. Frequently, both parts of speech are also misinterpreted as verbs. These errors occur in all models, among them most often in the ITTB model. Example 8.2 shows the noun *hypocrita* misinterpreted as adjective, and example 8.3 the adjective *mendax* misinterpreted as a noun, in the same sentence.

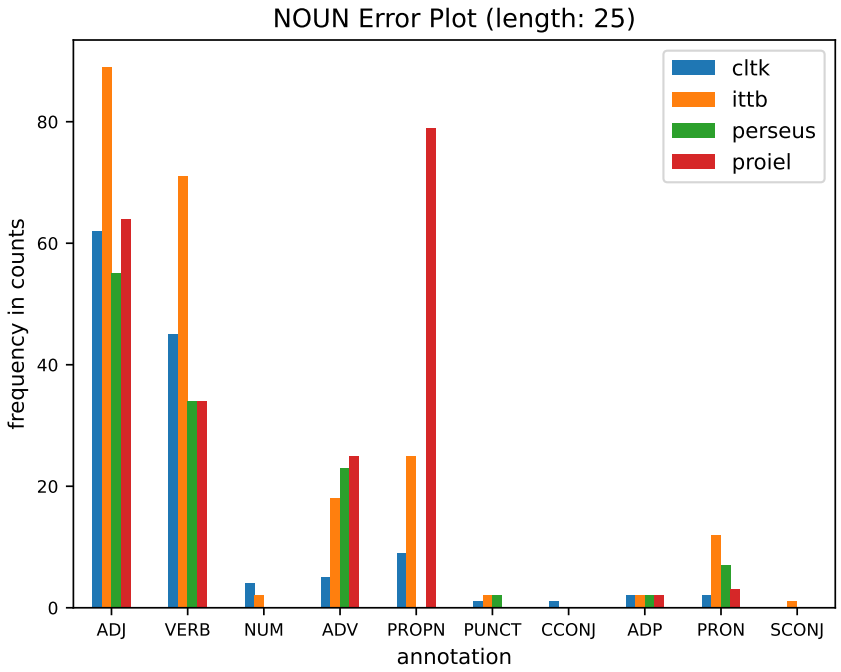

Figure 11: Error Plot for NOUN tag

(8.2)

,	*quum*	*hypocrita*	*tamen*	*vanus*
PUNCT	SCONJ	NOUN	ADV	ADJ

de. ', doch der Heuchler ist eitel'
en. ', but the hypocrite is vain'
[accessed 17th October 2022]

(8.3)

,	*quum*	*hypocrita*	*tamen*	*vanus*	*et*	*mendax*
PUNCT	SCONJ	NOUN	ADV	ADJ	CCONJ	ADJ

de. ', doch der Heuchler ist eitel und lügnerisch'
en. ', but the hypocrite is vain and lying'
[accessed 17th October 2022]

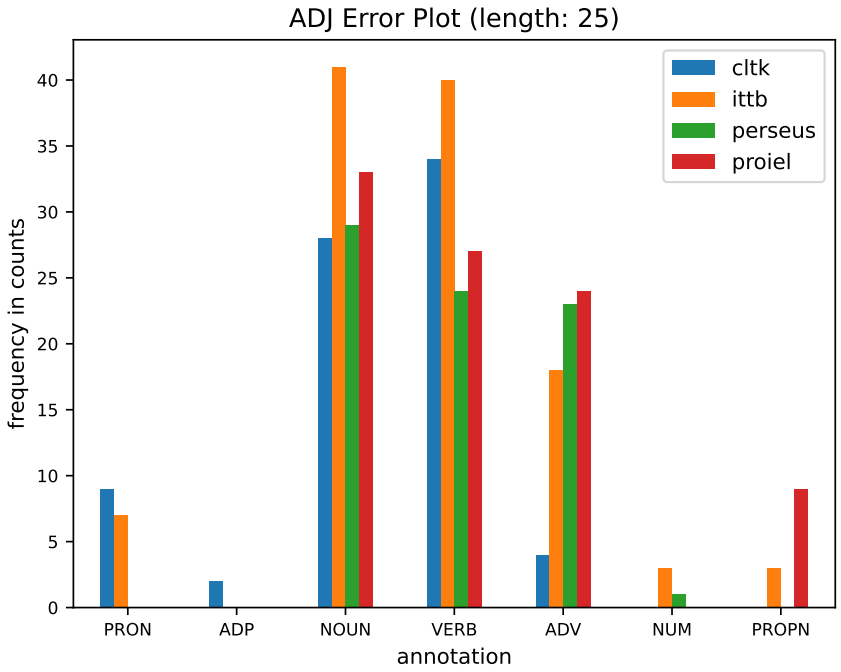

Figure 12: Error Plot for ADJ tag

3. Mistakes in PRON

As shown in Figure 13, PRON identification errors are mostly due to an erroneous annotation as ADJ tags. This can be caused by the same reasons as errors in tagging nouns, as inflectional paradigms of nouns and adjectives can overlap in pronouns as well [Sprugnoli et al., 2022]. Such an error is visible in example 8.4, where *omnia* is mistaken for an adjective.

(8.4) , *cum* *ostendimus* *omnia* *legi* *in* *scriptis* *Zvinglii*
PUNCT SCONJ VERB PRON VERB ADP VERB PROPN
et *Oecolampadii* ,
CCONJ PROPN PUNCT
de. ', indem wir in den Schriften von Zwingli und Oecolampadius alles gelesen haben,'
en. ', having read everything in the writings of Zwingli and Oecolampadius,'
[accessed 17th October 2022]

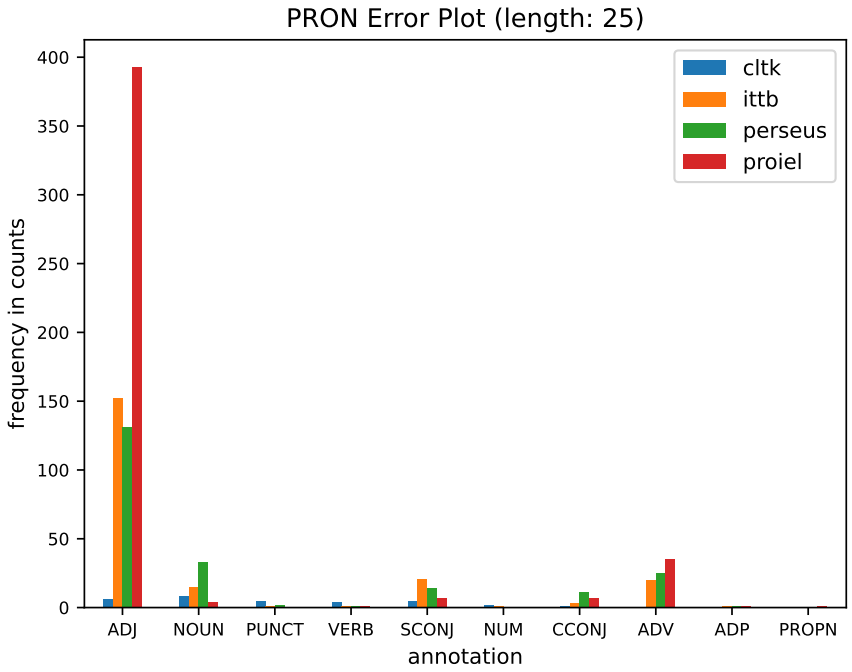

Figure 13: Error Plot for PRON tag

4. Mistakes in conjunctions CCONJ and SCONJ

   Both CCONJ and SCONJ tokens are most frequently mistaken for adverbs. This can be seen in Figure 14 and Figure 15. In a few cases, these two POS-tags are misinterpreted as pronouns, and the CCONJ tags even as nouns, verbs, or adjectives. Example 8.5 shows a misinterpretation as adverb for a CCONJ tag for the token *nec*, example 8.6 for a SCONJ tag with the token *tametsi*.

   (8.5) , *quod illius me participem esse volueris , nec*
   PUNCT SCONNJ PRON PRON ADJ AUX VERB PUNCT CCONJ
   dubito ,
   VERB PUNCT
   de. ', die du an ihm teilhaben willst, und ich bezweifle nicht,'
   en. 'that you want to share in it, and I have no doubt,'
   [accessed 17th October 2022]

   (8.6) , *tametsi et quedam de ista cum Pellican quoque*
   PUNCT SCONJ ADV PRON ADP PRON ADP PROPN ADV
   communicarim .
   VERB PUNCT
   de. ', auch wenn ich einiges über dieses Thema mit Pellican teilen würde.'

en. ', although I would share some things with Pellican on the subject.' [accessed 17th October 2022]

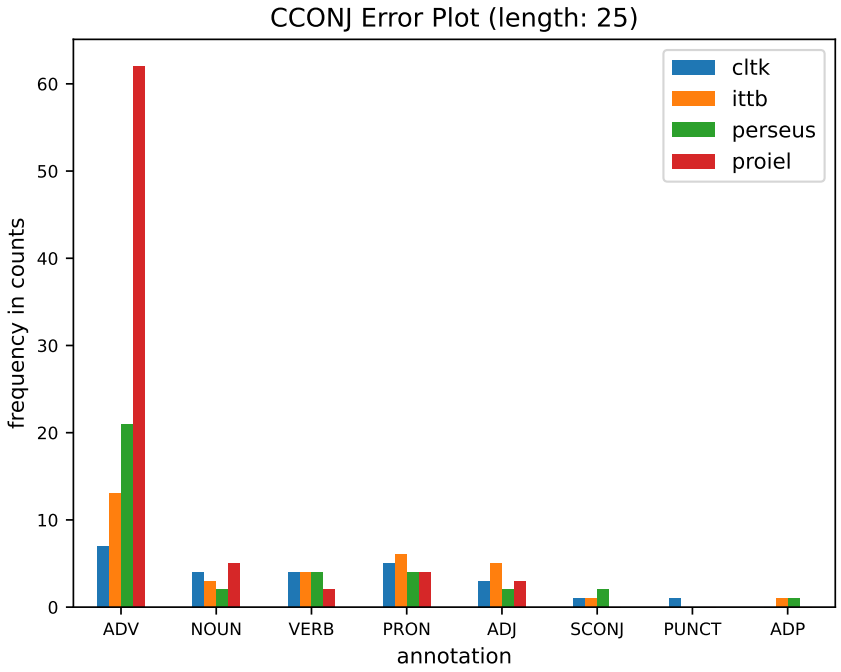

Figure 14: Error Plot for CCONJ tag

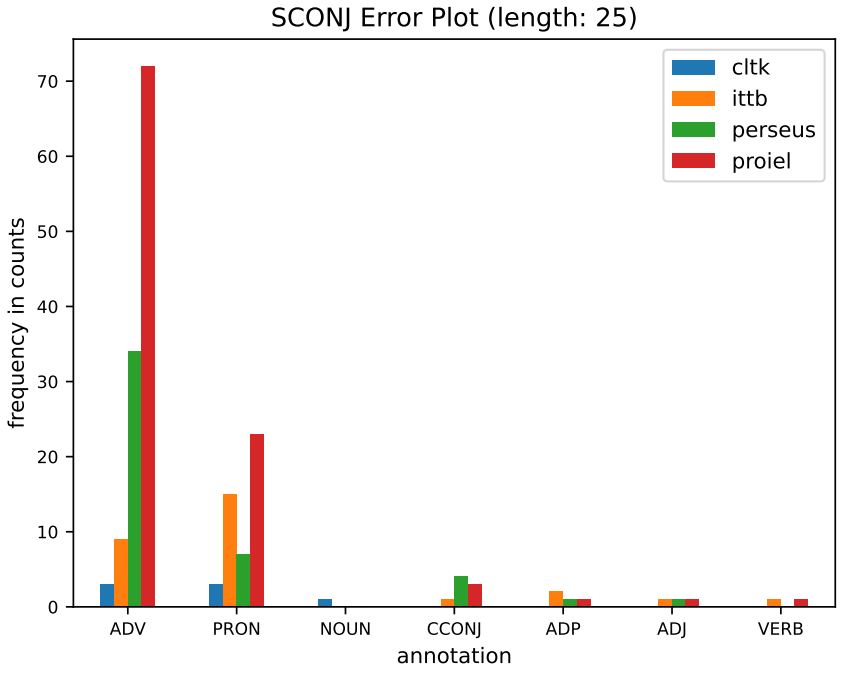

Figure 15: Error Plot for SCONJ tag

Besides the differing accuracies for individual POS-tags, it is noteworthy that all four models perform best on the sentence length category of 20-24 tokens, followed by the category of 15-19. The category of the longest sentences with 25 and more tokens shows the lowest model performance. Supposing that a sentence length of 20-24 tokens is frequent in Latin, it is possible that the POS-taggers work best on sentence lengths they were trained on. Furthermore, very long sentences (25 and more tokens) could be a reason for the model's performance drops in POS-tagging for sentences longer than 25 tokens, as it does in a similar way in the work of Berrichi and Mazroui [2021] or Kuang and Xiong [2016] for Machine Translations.

8.1.3 Weaknesses of the Models

As the four POS-tagging models were trained on different data, the models have their individual strengths and weaknesses. An analysis of the models is provided in the following sections.

8.1.3.1 CLTK

As most parts of the pipelines of CLTK wrap models trained by upstream projects such as Stanza, the CLTK POS-tagger is a well trained model when considering that Latin is a low-resource language. Even though it is predominantly created for the use on classical languages, it achieves the highest overall accuracies of the four models [Johnson et al., 2021].

A weakness of the CLTK tagger is the incorrect annotation of PROPN tags. Nevertheless, the CLTK tagger performs well on the conjunction tags as well as on most other tags, which ensure its reliability. The high accuracy of the model is also plausible since the interest in Classical Latin has generally been greater than in Medieval Latin. Studies on Latin grammar have focused on the limited span of the classical era and literary sources which emerged during this time, as it was considered the qualitative peak of Latin [Giacomelli, 1996].

8.1.3.2 UDPIPE

Surprisingly, the ITTB model did not outperform the CLTK model, even though it is trained on the largest training data size of the three UDPipe models and the training data includes Medieval Latin from the Index Thomisticus corpus, stemming from the 13^{th} century [Cecchini et al., 2018]. The model, similar to the CLTK tagger, shows a

low performance on PROPN tags over all sentence length categories. Nevertheless, for all other POS-tags, the average accuracy is acceptable, only dropping slightly for the PRON tag compared to the CLTK tagger.

Covering training data from 4th century or earlier, it is plausible that the PROIEL model and the PERSEUS model show a lower performance on the Latin data of the Bullinger letters [Cecchini et al., 2018]. Specifically, the PERSEUS model scores moderately well on PRON and SCONJ tags, with average accuracies of 81.21% and 84.44%. The PROIEL model shows particularly low scores on PRON and SCONJ as well, only achieving mean accuracies of 62.75% and 68.32%, respectively. Nevertheless, the scores on PROPN and ADV tags are remarkably high compared to the other models, achieving average 69.79% and 92.28%.

Still, these tags are not the most important ones for the research questions at hand. These peculiarities can be caused by differing training data as well as the relatively small training sets of these two models. The fact that Latin is a low-resource language results in sparse training data, especially from ancient times. Even though the classical era in Latin was much more in the focus of research, the size of the training data is rather small, and both models are outperformed by the CLTK model [Giacomelli, 1996].

8.1.4 Discussion of the POS-Tagging Process

As observed in the results of the POS-tagging task, the choice of the POS-tagging model seems to have a larger impact on the performance than the length of a sentence. Even though all models performed best on sentence lengths of 20-24 tokes, the performance differences are small. Furthermore, the four models have different strengths and weaknesses with respect to different POS-tags, possibly due to the data they were trained on.

All models had difficulties annotating PROPN tags, which denote proper nouns. Mostly, such tokens are mistaken for nouns. In addition, adjectives and nouns are often mistaken for one another as the inflectional paradigms overlap to a great extent. This also applies to pronouns, on certain occasions [Sprugnoli et al., 2022]. Furthermore, coordinating and subordinating conjunctions are occasionally erroneously tagged as adverbs.

For the 16^{th} century epistolary Latin of the Bullinger letters, the POS-tagger from CLTK achieves the best overall accuracies, especially for the POS-tags which are crucial for the splitting step in this thesis. The POS-tags which need to obtain a

good accuracy in the tagging step include the PUNCT tags, as well as the CCONJ and SCONJ tags [Fischer et al., 2022].

8.2 Sentence Splitting

In the following sections, I discuss the results of the splitting method with the splitting rules implemented in the script `splits.py` and the results of the translations described in 7.4. With these splits, I carry out the following two translation experiments:

1. Recomposing clause translations by Bullinger NMT into sentences
2. Adding clause translations by GoogleTranslate to the training data

8.2.1 Evaluation of the Splitting Rules

I first evaluate the general translation quality in Bullinger NMT of each splitting rule after the application of the splitting technique in this section. For the first experiment, I perform a simple manual qualitative evaluation on the files with sentences of 25 or more tokens. The evaluation is done according to the two criteria of adequacy and fluency. Using the script `evaluate_splits2sents.py`, the evaluation yields differing results for the different splitting rules. Figure 16 presents an overview of the results of the manual evaluation. In Section 8.2.2 and 8.2.3, I proceed with a detailed error analysis of the respective translation experiments.

8.2.1.1 Splitting at Punctuation Markers

As described in Section 7.4.1.1, I examine the following experiments using splitting rules with punctuation markers:

1. Translating all clauses separately.
2. Putting the first and last sentence part back together and translate the clause composition.
3. Only splitting the sentence at a punctuation marker in the middle of the sentence, and translating these two sentence halves.

The results for the first splitting rule show that it yields sensible clauses, as suggested in Su et al. [2018] and Zhang and Matsumoto [2019] as well, even though the clauses

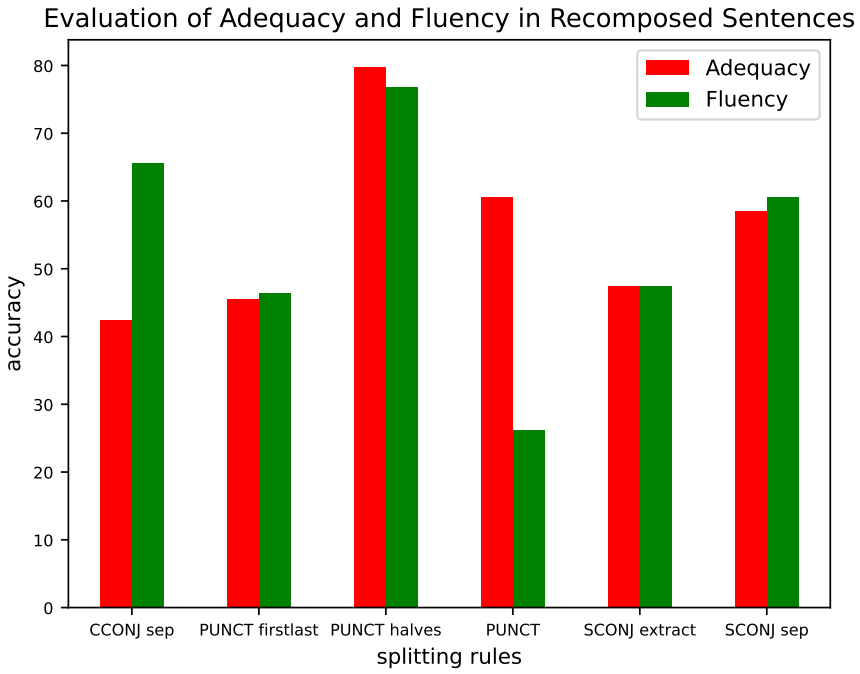

Figure 16: Evaluation of Bullinger NMT with different Splitting Rules

tend to be short occasionally. The translation of all separate clauses split at punctuation markers produces a more exact translation than the translation of the same long sentence without splits, which is reflected in a score of 60.61% in the manual evaluation. A problematic aspect is that Bullinger NMT has difficulties translating short and incomplete sentences, which results in a lack of fluency in these cases as is shown in example 8.7 and reflected in a score of only 26.26%. Recomposing a sentence from the first and the last part of a sentence produces good translations in some occasions, but not consequently over a larger amount of sentences. The obtained adequacy score in the manual evaluation is 45.46%. An example is shown in 8.8. Since the resulting sentence is shorter in length, it loses most of its sentence information, which results in this lack of adequacy. Splitting a sentence in halves, on the other hand, works well. This is especially the case when the sentence structure consists of two parts of equal length. The adequacy score reaches 79.80%, the fluency score 76.77%. In other cases, however, translating two halves may lack context as well, which is shown in example 8.9. However, in most cases, the greatest part of the sentence information is preserved. Overall, this splitting rule represents a good compromise between the aim of producing shorter sentences and not creating too many fluency breaks, as only one split is applied.

(8.7) *De libris apud nos absolutis his nundinis nondum potui*
About books with us appeared this fair not.yet been.able.to
expiscari; sed hac hebdomada id efficiam.
find.out, but this week that I.will.do.

split-MT-de. 'Von den Büchern, die bei uns freigesprochen wurden, konnte ich noch nicht fischen. ; aber in dieser Woche werde ich es tun .'

MT-de. 'Was die bei uns abgeschlossenen Bücher angeht, so konnte ich noch nicht herausfinden, aber ich werde es in dieser Woche tun.'

ref-de. 'Über die zu dieser Messe bei uns erschienenen Bücher konnte ich noch nichts in Erfahrung bringen; aber in dieser Woche werde ich das tun.'

en. 'I have not yet been able to find out about the books published at this fair, but I will do so this week.'

[accessed 10th October 2022]

(8.8) *Diu nihil ad te scripsi, chare mi Myconi, sed*
long.time nothing to you I.have.written, dear my Myconius, but
modo copiosius tecum colloquar per librum, quem
now more.extensively with.you converse through book, that
mitto.
I.send.

split-MT-de. 'Ich habe dir lange nichts geschrieben. , den ich sende.'

MT-de. 'Ich habe dir lange nichts geschrieben, mein lieber Myconus, aber jetzt werde ich mich ausführlicher mit dir durch das Buch unterhalten, das ich dir schicke.'

ref-de. 'Ich habe lange nicht an dich geschrieben, mein lieber Myconius, aber nun möchte ich mich mit dir durch das Buch unterhalten, welches ich hier schicke.'

en. 'I have not written to you for a long time, my dear Myconius, but now I would like to converse with you through the book I am sending here.'

[accessed 10th October 2022]

(8.9) *Pro benignitate autem tua, qua me quam plurimis annis*
For kindness however your, with.which me for many years
perpetuo complecteris, tibi gratias ago immortales.
perpetually you.embrace, you thanks I.make infinitely.

split-MT-de. 'Für deine Güte, mit der du mich in vielen Jahren immer wieder umarmst , danke Ihnen unsterblich.'

MT-de. 'Für deine Güte, mit der du mich in vielen Jahren immer wieder umarmst, danke ich dir unsterblich.'

ref-de. 'Für deine Güte, mit der du mich <schon> sehr viele Jahre bedenkst, sage ich dir unendlichen Dank.'

en. 'For your kindness, with which you have thought of me for many years, I thank you infinitely.'

[accessed 10th October 2022]

8.2.1.2 Splitting at Conjunctions

While Berrichi and Mazroui [2021] identify semantic segmentation markers, which are words that serve as links between two sentence segments, I use conjunctions as positions to split a sentence at. The approach to extract the subordinating clauses yields sensible clauses. During the translation, however, the separate clauses mostly lack important context information in order for the system to translate the sentence meaning correctly, which is reflected in example 8.10 and in an adequacy score of 47.48%. Better adequacy is reached with the splitting rule where a sentence is segmented at each SCONJ-tag. This segmentation yields sensible subsequences, too. The translation system occasionally has difficulties interpreting the structure of a subsequence due to a lack of context, as can be seen in example 8.11. However, an adequacy score of 58.59% is reached, and also a moderate fluency score of 60.61%. A possible reason for the better performance with this split can be the number of splits applied. With normally one to three splits, most of the context in long sentences is still preserved. Even though coordinating conjunctions combine two clauses that are equal to each other, the translation of sentence parts split at CCONJ-tags only reaches an adequacy score of 42.42%, possibly due to occasionally short sequences. The fluency, however, still reaches a reasonable score of 65.66%, which is reflected in example 8.12.

(8.10) *Utinam is sim, mi Bullingere, qui aliquando non*
If.only that be, my.dear Bullinger, that one.day not
ingratum me declarare possim!
ungrateful myself prove I.could!

split-MT-de. 'Utinam is sim, Mein Bullinger, der ich eines Tages nicht undankbar erklären kann!'

MT-de. 'Ich wünschte, mein Bullinger, dass ich mich eines Tages nicht undankbar erklären kann!'

ref-de. 'Wenn ich mich doch eines Tages ganz dankbar erweisen könnte, mein lieber Bullinger!'

en. 'If only one day I could prove myself quite grateful, my dear Bullinger!' [accessed 10th October 2022]

(8.11) *Faxit gratia Christi salutaris, ut filio dei*
May.have.caused grace of.Christ saving, that Son of.God
syncere cognito in ipso aeternum vivas.
through.right knowledge in Him eternally you.live.

split-MT-de. 'Gebe die Gnade Christi des Heils, Wie du im Sohne Gottes aufrichtig erkennst und in ihm ewig lebst.'

MT-de. 'Gebe dir die heilsame Gnade Christi, damit du, nachdem du den

Sohn Gottes aufrichtig erkannt hast, in ihm ewig lebest.'

ref-de. 'Die heilsame Gnade Christi möge bewirkt haben, dass du nach der rechten Erkenntnis des Sohnes Gottes in ihm ewig lebst.'

en. 'May the saving grace of Christ have caused you to live eternally according to the right knowledge of the Son of God in Him.'

[accessed 10th October 2022]

(8.12) *Roga igitur dominum, ut facis, pro misera et afflicta ecclesia.*
Pray so to.the.Lord, as you.do, for miserable and afflicted church.

split-MT-de. 'Bete also, wie du es tust, für elend Und die heimgesuchte Kirche'

MT-de. 'Bete also, wie du es tust, für diese elende und übel mitgenommene Kirche zum Herrn.'

ref-de. 'Bete also, wie du es tust, für diese elende und übel mitgenommene Kirche zum Herrn.'

en. 'Pray to the Lord, then, as you do, for this miserable and afflicted church.'

[accessed 10th October 2022]

8.2.2 Error Analysis of Bullinger NMT Sentence Translations

After the translation of the Latin clauses, the German output is recomposed to sentences by the script `splits2sents.py`. As mentioned in Section 8.2, Figure 16 presents an overview of the results obtained from the qualitative manual evaluation on the files with sentences of 25 or more tokens. Specific errors occurred more frequently in some splitting scenarios than in others. In this section, I discuss the most important observations.

1. Splitting at punctuation markers occasionally results in MT output that lacks fluency.

   As mentioned in Section 8.2.1.1, Bullinger NMT has difficulties translating short and incomplete sentences, as is shown in example 8.7. An advantage of splitting at each punctuation marker and separately translating each clause is the completeness of context of the sentence. Less frequently than in other settings, sentence parts or words are omitted or lost with this splitting rule.

   Nevertheless, this rule most often lacks fluency of the German output, and the clauses do not always represent well-connected sentence parts, which is reflected by the low fluency score in the manual evaluation of 26.26%. A

possible explanation for this observation could be the training process of the model, as the Bullinger NMT system is mainly trained on complete sentences. This is as well suggested by the translation output of example 8.12 with a CCONJ POS-tag. This tag normally combines complete sentences and results in a better partial translation in these examples.

2. Splitting at all punctuation markers and at conjunctions often produces short clauses, which are occasionally copied.

   For the same reasons, possibly, Latin words are occasionally not translated at all, which can be seen in examples 8.10. A plausible explanation for the copying of Latin words instead of their translation could be the lack of context. However, translations of the splitting rule at each punctuation marker still reach 60.61% in adequacy, as most of the sequences are translated accurately.

3. Only using the first and last punctuation split generally loses information and lacks fluency and context.

   While shorter sentences can be produced by this splitting rule, in most cases, the context is not sufficient to produce an accurate translation. This is shown in example 8.8 above. Additionally, with short clauses, Latin words are not translated, possibly also because these sentences tend to lack context. Consequently, translations of this splitting rule reach 45.46% in both adequacy and fluency.

4. Splitting at the punctuation marker in the middle of the sentence produces a fluency break.

   Sentences composed after splitting at the punctuation marker in the middle of the sentences generally show a fluency break at this point of the translation, as shown in example 8.9. On the other hand, most of the sentence information is even obtained in the translation of long sentences of more than 25 tokens. The accuracies of 79.80% in adequacy and 76.77% in fluency exceed those of the splitting rule at each punctuation marker with even more fluency breaks.

5. Several conjunctions in one sentence make the structure obscure.

   The translation of split segments at SCONJ-tags does not always yield sensible translations. This could be caused by the fact that a subordinating clause is usually not placed at the end of a sentence, but only in certain cases. The same is true for sequences split at CCONJ-tags, as these splits can be very short occasionally. Therefore, the NMT system has difficulties interpreting the structure of a subsequence starting with a token tagged as SCONJ or CCONJ.

An example is provided in 8.10.

8.2.3 Error Analysis of GoogleTranslate Clause Translations

This experiment consists of extracting a shuffled sample of sentences longer than 15 tokens using the script `extraction_all.py`, translating separate splits obtained from `traintest_google_postagger.py`, `traintest_google_shuffle.py`, and `traintest_google_splits.py` into German using the GoogleTranslate API, and adding them to the training data. The automatic evaluation of the model performance on the test set with the BLEU score suggests that the translation quality cannot be increased by adding the split clauses. Before adding the GoogleTranslate data to the training data, the MT system reaches 21.16 BLEU, and afterwards 21.06 BLEU. However, the BLEU score increases to 21.96 when the names contained in the translated test set file are postprocessed semi-automatically with find-and-replace commands, and the BLEU score thus exceeds the one of the former model. As the Bullinger NMT system performs well on the translation of names contained in the Bullinger letters, it is obvious that using GoogleTranslate can reduce the overall translation quality due to this issue. Table 3 shows the different BLEU scores reached by the systems with and without the additional training data translated by GoogleTranslate, as well as with postprocessing.

Model	Description	BLEU score
e38	without GoogleTranslate	21.16
e39	with GoogleTranslate	21.06
e39	with GoogleTranslate and postprocessed names	21.96

Table 3: BLEU of the GoogleTranslate Experiment

Besides the automatic metric of BLEU, it is crucial to carry out a human evaluation, especially because a rather small testset is used for the Bullinger letters. The comparison between texts translated by the newly trained and the previous model provide information about the strengths and weaknesses of the new model trained on texts containing splits. Compared texts include the testset as well as files containing segmented sentences that are not included in the training data. The following error analysis lists the most frequently occurring issues with these translations:

1. Bad translation of names.

   In the model trained on data which includes segmented sentences translated by GoogleTranslate, it is noticeable that names are not translated as well as

in the previous model. Sometimes names are even anglicised. As the Bullinger NMT system has not been trained on English data, this is rather due to the use of GoogleTranslate, which might be trained on multi-lingual data, than due to the sentence segmentation. Examples for such problematic name translations are visible in example 8.13.

(8.13) , *per* *quae* *non* *solum* *Tigurinae*
, through which not only in.Zurich

split-MT-de. ', durch die nicht nur Tigurinae'
MT-de. 'durch die du nicht nur in Zürich'
ref-de. 'durch die nicht nur in Zürich'
en. 'through which not only in Zurich'
[accessed 6th November 2022]

2. High occurrence of English sentence parts.

Besides the anglicising of names on the Bullinger letters, there are a striking number of English phrases in the translated sentences, as shown in example 8.14. This observation could as well be due to a multilingual model Google-Translate could be using [Fischer et al., 2022].

(8.14) *Contra* *ego* *tibi* *mitto* *Epitome* *reformationis* *Anglicae*
In.return I you send Epitome of.the.Reformation English

split-MT-de. 'Andererseits sende ich Ihnen den Epitome of the English Reformation'
MT-de. 'Dagegen schicke ich dir das Epitome der englischen Reformation'
ref-de. 'Dagegen schicke ich dir das Epitome der englischen Reformation'
en. 'In return I am sending you the Epitome of the English Reformation'
[accessed 6th November 2022]

3. Form of politeness.

In some sentences, the form of politeness is chosen, even though it is an informal sentence. Example 8.15 shows such a form of politeness, where **de.** "Sie" is used instead of **de.** "du".

(8.15) , *hoc* *senciamus*
, this we.feel

split-MT-de. ', lassen Sie uns das verstehen'

MT-de. ', wollen wir das denken'
ref-de. ', wir fühlen das'
en. ', we feel this'
[accessed 6th November 2022]

4. Misinterpretation of short clauses.

   Occasionally, short clauses are treated as complete sentences and interpreted as questions instead of phrases belonging to a longer sentence. One instance of this issue can be seen in example 8.16. Generally, the new model has the tendency to form shorter translations than the previous model, which is likely due to both the training on shorter sentence segments, and possibly also the use of GoogleTranslate.

   (8.16) *Quis enim hoc*
   Who for this

   split-MT-de. 'Wer ist das?'
   MT-de. 'Denn wer würde das'
   ref-de. 'Denn wer [würde] das'
   en. 'For who [would] this'
   [accessed 6th November 2022]

5. Copying of short clauses.

   In other cases, short clauses are copied instead of translated, possibly because of lack of context. Example 8.17 presents this copying issue. This can be observed more often in the model trained on shorter segments as well. A possible explanation for this issue could lie in the additional training data. If short sequences of the segmented Latin data are not recognized by GoogleTranslate and therefore copied instead of translated to German, these copied sequences also occur in the training data, consequently.

   (8.17) *Nam is ea*
   For is that

   split-MT-de. 'Nam ist EA'
   MT-de. 'Denn das'
   ref-de. 'Denn das'
   en. 'For that'
   [accessed 6th November 2022]

6. Tendency to paraphrase expressions containing nouns.

In addition, in the translated text of the model trained on segmented sentences, a tendency to use fewer nouns than its predecessor can be observed. Example 8.18 shows an instance of this difference in the two models.

(8.18) *Porro non dubium est ,*
Further no doubt is ,

split-MT-de. 'Ferner ist es nicht zweifelhaft.'
previous-MT-de. 'Es gibt keinen Zweifel daran,'
MT-de. 'Es ist kein Zweifel daran'
en. 'There is no doubt about it,'
[accessed 6th November 2022]

8.2.4 Discussion of the Splitting Process

Assessing the methods used for the first experiment of recomposing translated sentence splits to complete sentences, the results suggest that splitting at the middle punctuation marker is the most successful splitting method. The method represents a good optimum between the goal of producing shorter sentences and not creating too many fluency breaks, as only one split is applied. While other methods such as splitting at each punctuation marker or extracting subordinating clauses yields sensible sentence splits, the translation quality often drops due to a lack of context in very short clauses.

The GoogleTranslate experiment showed that it is possible to add shorter sentence sequences consisting of splits to the training material in order to augment the training data. The BLEU score, as an automatic metric of the quality of a translation, remained similar to the system without the additional GoogleTranslate data with 21.06 compared to 21.16. With additional postprocessing of the translation of the testset, consisting of corrections of incorrectly translated names due to the use of GoogleTranslate, BLEU increases to a score of 21.96. While automatic metrics are a useful measure of quality, it is still important to perform a human evaluation. This is because these two types of evaluation do not always coincide for high-quality translation systems, and automatic metrics should generally reflect the human assessment of systems. The error analysis shows that with segmented sentences translated by GoogleTranslate in the training data, the model tends to copy Latin words more frequently, including names. Occasionally, English sentence parts occur in German translations. Additionally, the form of politeness is used incorrectly in certain cases.

Thus, considering the BLEU score, the answer to the research question whether the translation quality of 16th century epistolary Latin into German can be improved by

splitting long sequences into smaller units is yes. The evaluation on the testset of the Bullinger data suggest that Data Augmentation by adding split sentence clauses translated by GoogleTranslate could increase the performance of the system. The splitting rules including the segmentation of sentences on the middle punctuation marker as well as before tokens annotated with SCONJ-tags proved to be the most successful ones. This follows the observation of Su et al. [2018] and Zhang and Matsumoto [2019] that sensible splits of long sentences can be applied on punctuation markers, and is supported by the declaration of Horrocks [2011] that sentences consisting of several clauses are syntactically composed through coordination or subordination. These two splitting rules segment a long sentence into two to four parts. While the translation of short segments occasionally obtained from splitting at each punctuation marker yields an unsatisfying quality, the results of the two methods mentioned above suggest that the obtained moderate sentence length is optimal to deal with long sentences and to avoid separations of discontinuously represented constituents [Horrocks, 2011].

9 Conclusion

9.1 Summary and Main Splitting Results

In this thesis, I aimed to optimize the translation quality from Latin to German as part of the Bullinger Digital Project[1]. The Bullinger Digital Project has the goal to make Heinrich Bullinger's correspondence accessible to the public and create a database with metadata and links to the digital recordings for each individual letter. The NMT system used in the Bullinger Digital Project is based on the Transformer architecture. A challenge for state-of-the-art NMT systems of today is the translation of long sentences. As the NMT system developed in the Bullinger Digital Project performs well on short and medium Latin sentences, long sequences still pose challenges to the NMT model [Fischer et al., 2022].

Thus, this thesis investigated Data Augmentation methods to improve translations of such long sentences. With use of POS-tagging, I applied sentence segmentation on long sentences based on difference splitting methods, similar to Su et al. [2018] and Zhang and Matsumoto [2019]. For the POS-tagging step, the tagger from the CLTK showed the best performance and was used subsequently. The most promising splitting methods proved to be segmentation of a sentence on the middle punctuation marker, as well as segmenting a sentence before each subordinating conjunction. I conducted different experiments on the segmented Latin sentences. The first experiment consisted of the translation and recomposition of complete sentences. The second one involved the translation of the segments using GoogleTranslate, and the subsequent addition of these translations to the training data in order to train a new NMT model. With an additional semi-automatic postprocessing step, incorrectly translated names could be corrected. With this experiment, the evaluation on the testset achieved 21.96 BLEU, exceeding the performance of the previous model with 21.16 BLEU.

These results answer my first research question at the beginning of this thesis. The quality of the Neural Machine Translation of 16$^{\text{th}}$ century epistolary Latin into Ger-

[1] https://www.bullinger-digital.ch/about

man can indeed be slightly improved by splitting long sentences into smaller units. An important reason for the improvement is the addition of more in-domain training data. The second research question is whether the improvement can be carried out by automatically splitting sentences and recomposing the translated subsequences to complete sentences after separate translation into German. This approach proved to be feasible as well. Besides the qualitative manual evaluation in this thesis, it would nevertheless be interesting to evaluate this approach quantitatively. The third research question can be answered positively, too. In terms of the BLEU score, adding the split sequences separately to the training material improved the model performance when including a postprocessing step. The investigation and evaluation of sensible sentence positions to perform splits showed that the most promising splitting methods are splitting a sentence at the middle punctuation marker, as well as splitting it before each subordinating conjunction. Both methods yield a reasonable sentence length to maintain the sentence fluency while keeping most of the context needed for an adequate translation. For further research, these two methods prove to be the most interesting ones.

9.2 Outlook

While, for high-resource language pairs, long sentences are translated in good quality, the model performance drops for low-resource language-pairs such as those containing Latin [Neishi and Yoshinaga, 2019; Kondo et al., 2021]. This makes long sentence translation a major issue in low-resource scenarios [Kondo et al., 2021]. Future research to tackle this problem should be carried out in the field of Data Augmentation. Other promising sentence simplification and segmentation techniques appear to be attention-based methods to translate sentence clauses separately [Kuang and Xiong, 2016; Su et al., 2018; Li et al., 2020], segmenting and translating separate clauses using statistical tools such as the Moses toolkit [Koehn et al., 2007; Tien and Minh, 2019; Berrichi and Mazroui, 2021], corpus augmentation via segmentation of long sentences using back-translation [Zhang and Matsumoto, 2019], clause substitution by maintaining the same label [Shi et al., 2021], and exploring possibilities of positional encoding when using Transformer architectures for translation [Neishi and Yoshinaga, 2019].

For further work related to this thesis, I consider it most interesting to split and translate Latin sentences from the Bullinger texts using GoogleTranslate by only applying the two best-working splitting methods, and then adding the translations to the training data in order to train another Transformer model. Furthermore, it

can be promising to evaluate the methods used in this thesis on other low-resource language pairs where the source language has the tendency to feature long sentences, as long sentences tend to be an issue for several low-resource language pairs in NMT. With further research in low-resource NMT, more sparse-data languages such as Latin can be made accessible to speakers of other languages, and historical and linguistic knowledge can be preserved and spread for further research and learning.

Glossary

A glossary of this thesis' most important terms is provided in this section.

accuracy A basic score for evaluating automatic **annotation tools** such as **parsers** or **part-of-speech taggers**. It is equal to the number of **tokens** correctly tagged, divided by the total number of tokens.

adequacy A measure on whether the of a **machine translation system** conveys the same meaning as the input sentence and whether part of the message is lost, added, or distorted.

BLEU score *(bilingual evaluation understudy)* An algorithm for evaluating the quality of text which has been **machine-translated** from one natural language to another. It is equal to a weighted geometric mean of all the modified **n-gram precisions**, multiplied by the **brevity penalty**.

context vector A representation of the weighted sums of **source sentence annotations**. It is equal to the multiplication of the **encoder hidden states** and their respective **alignment scores**.

data augmentation In **natural language processing**, an approach to generate or collect more data in order to obtain better results in **low-resource scenarios**.

fluency A measure on whether the output of a **machine translation system** is fluent in the **target language**. This involves both grammatical correctness and idiomatic word choices.

gold standard A **data set** which has been **annotated** (either manually or automatically) and then manually corrected. Gold standards are used for testing and evaluating **automatic NLP tools**.

hidden state A **vector** representing an intermediate form of the original input data in a **neural network**. It captures previous information and gets updated with each new data piece such as a new word in the sentence to translate by an **NMT model**.

hyperparameter A **parameter** whose value is used to control the **learning process** of a model. By contrast, other parameters are usually **node weights** and derive their values via **training**.

low-resource language A language which has few or no labelled or unlabelled data. Oftentimes a low-resource language is endangered, less computerized, less digitized, less studied, less commonly taught and sometimes also has a low density and low prestige.

part-of-speech tagging A **process in corpus linguistics** of **annotating** a word in a text or corpus with its corresponding **part of speech** based on both its definition and its context.

source language In **machine translation**, the language to translate into the **target language**.

target language In **machine translation**, the language to translate the **source language** into.

References

M. M. I. Alam and A. Anastasopoulos. Fine-Tuning MT Systems for Robustness to Second-Language Speaker Variations. In *Proceedings of the Sixth Workshop on Noisy User-Generated Text (W-NUT 2020)*, pages 149–158, 2020.

D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014.

D. Bamman and P. J. Burns. Latin BERT: A Contextual Language Model for Classical Philology. *CoRR*, abs/2009.10053, 2020.

S. Berrichi and A. Mazroui. Addressing Limited Vocabulary and Long Sentences Constraints in English–Arabic Neural Machine Translation. *Arabian Journal for Science and Engineering*, 46(9):8245–8259, 2021.

P. J. Burns. Building a Text Analysis Pipeline for Classical Languages. *Digital Classical Philology: Ancient Greek and Latin in the Digital Revolution*, 10: 159–176, 2019.

F. M. Cecchini, M. Passarotti, P. Marongiu, and D. Zeman. Challenges in Converting the *Index Thomisticus* treebank into Universal Dependencies. *Proceedings of the Universal Dependencies Workshop 2018 (UDW 2018)*, 2018.

J. Chen, D. Tam, C. Raffel, M. Bansal, and D. Yang. An Empirical Survey of Data Augmentation for Limited Data Learning in NLP. *CoRR*, abs/2106.07499, 2021.

T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *CoRR*, abs/1512.01274, 2015.

A. Chernyavskiy, D. Ilvovsky, and P. Nakov. Transformers:"The End of History" for Natural Language Processing? In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 677–693, Bilbao, 2021.

K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR*, abs/1409.1259, 2014.

C. Christodouloupoulos and M. Steedman. A Massively Parallel Corpus: The Bible in 100 Languages. *Language Resources and Evaluation*, 49(2):375–395, 2015.

T. Clérice, L. Cerrato, A. Babeu, B. Almas, N. Jovanović, A. Gessner, P. J. Burns, Stella, mkonieczny9805, M. Munson, M. Jøhndal, maryam foradi, zachhimes, MMernitz, M. Seydi, G. G. A. Celano, S. Scott, S. J. Huskey, and TDBuck. PerseusDL/canonical-latinLit: None, Apr. 2022. URL `https://doi.org/10.5281/zenodo.6418631`.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988. Association for Computational Linguistics, 2019.

D. Dalva, U. Guz, and H. Gurkan. Effective Semi-Supervised Learning Strategies for Automatic Sentence Segmentation. *Pattern Recognition Letters*, 105:76–86, 2018.

A. Erdmann, C. Brown, B. Joseph, M. Janse, P. Ajaka, M. Elsner, and M.-C. de Marneffe. Challenges and Solutions for Latin Named Entity Recognition. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 85–93, 2016.

S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy. A Survey of Data Augmentation Approaches for NLP. *CoRR*, abs/2105.03075, 2021.

L. Fischer, P. Scheurer, R. Schwitter, and M. Volk. Machine Translation of 16th Century Letters from Latin to German. In *Proceedings of the Second Workshop on Language Technologies for Historical and Ancient Languages*. LREC, 2022.

G. Franzini, A. Peverelli, P. Ruffolo, M. Passarotti, H. Sanna, E. Signoroni, V. Ventura, and F. Zampedri. Nunc Est Aestimandum: Towards an Evaluation of the Latin WordNet. In *CLiC-it*, 2019.

E. M. Garcia and Á. G. Tejedor. Latin-Spanish Neural Machine Translation: from the Bible to Saint Augustine. In *Proceedings of LT4HALA 2020-1st Workshop on Language Technologies for Historical and Ancient Languages*, pages 94–99, 2020.

D. Garrette, H. Alpert-Abrams, T. Berg-Kirkpatrick, and D. Klein. Unsupervised Code-Switching for Multilingual Historical Document Transcription. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1036–1041, 2015.

J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin. A Convolutional Encoder Model for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135. Association for Computational Linguistics, 2016.

R. Giacomelli. *Storia della lingua latina*. Jouvence, 1996.

R. Gleim, S. Eger, A. Mehler, T. Uslu, W. Hemati, A. Lücking, A. Henlein, S. Kahlsdorf, and A. Hoenen. A Practitioner's View: A Survey and Comparison of Lemmatization and Morphological Tagging in German and Latin. *Journal of Language Modelling*, 7(1):1–52, 2019.

S. T. Gries and A. L. Berez. Linguistic Annotation in/for Corpus Linguistics. *Handbook of Linguistic Annotation*, pages 379–409, 2017.

R. Guarasci. Developing an Annotator for Latin Texts using Wikipedia. *Journal of Data Mining and Digital Humanities*, 2017.

M. A. Hedderich, L. Lange, H. Adel, J. Strötgen, and D. Klakow. A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2545–2568. Association for Computational Linguistics, 2020.

F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, and M. Post. Sockeye: A Toolkit for Neural Machine Translation. *CoRR*, abs/1712.05690, 2017.

V. C. D. Hoang, P. Koehn, G. Haffari, and T. Cohn. Iterative Back-Translation for Neural Machine Translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, 2018.

G. Horrocks. Latin Syntax. *A Companion to the Latin Language*, pages 118–143, 2011.

K. P. Johnson, P. J. Burns, J. Stewart, and T. Cook. CLTK: The Classical Language Toolkit, 2014–2021. URL `https://github.com/cltk/cltk`.

K. P. Johnson, P. J. Burns, J. Stewart, T. Cook, C. Besnier, and W. J. Mattingly. The Classical Language Toolkit: An NLP Framework for Pre-Modern Languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 20–29, 2021.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. 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, 2007.

S. Kondo, K. Hotate, M. Kaneko, and M. Komachi. Sentence Concatenation Approach to Data Augmentation for Neural Machine Translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 143–149. Association for Computational Linguistics, 2021.

S. Kuang and D. Xiong. Automatic Long Sentence Segmentation for Neural Machine Translation. In *Natural Language Understanding and Intelligent Applications*, pages 162–174. Springer, 2016.

V. Kumar, A. Choudhary, and E. Cho. Data Augmentation using Pre-Trained Transformer Models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26. Association for Computational Linguistics, 2020.

D. Li, I. Te, N. Arivazhagan, C. Cherry, and D. Padfield. Sentence Boundary Augmentation for Neural Machine Translation Robustness. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7553–7557. IEEE, 2021.

Z. Li, R. Wang, K. Chen, M. Utiyama, E. Sumita, Z. Zhang, and H. Zhao. Explicit Sentence Compression for Neural Machine Translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34.05, pages 8311–8318, 2020.

E. Litta, M. Passarotti, and C. Culy. Formatio Formosa Est. Building a Word Formation Lexicon for Latin. In *CLiC-it/EVALITA*, 2016.

E. Litta, M. Passarotti, and F. Mambrini. The Treatment of Word Formation in the LiLa Knowledge Base of Linguistic Resources for Latin. In *Proceedings of*

the Second International Workshop on Resources and Tools for Derivational Morphology, pages 35–43, 2019.

A. Magueresse, V. Carles, and E. Heetderks. Low-Resource Languages: A Review of Past Work and Future Challenges. *CoRR*, abs/2006.07264, 2020.

M. Mayrhofer. Zur Gestaltung des etymologischen Wörterbuches einer "Großcorpus-Sprache": Veröffentlichungen der Kommission für Linguistik und Kommunikationsforschung/Österreichische Akademie der Wissenschaften, Philosophisch-Historische Klasse. *Veröffentlichungen der Kommission für Linguistik und Kommunikationsforschung / Österreichische Akademie der Wissenschaften, Philosophisch-Historische Klasse*, 1980.

B. McGillivray. *Methods in Latin Computational Linguistics*. Brill, 2013.

K. Mokhtar, S. S. Bukhari, and A. Dengel. OCR Error Correction: State-of-the-Art vs an NMT-based Approach. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 429–434. IEEE, 2018.

R. Murthy, A. Kunchukuttan, and P. Bhattacharyya. Addressing Word-Order Divergence in Multilingual Neural Machine Translation for Extremely Low Resource Languages. 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 3868–3873. Association for Computational Linguistics, 2018.

M. Neishi and N. Yoshinaga. On the Relation between Position Information and Sentence Length in Neural Machine Translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 328–338, 2019.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

M. Passarotti, M. Budassi, E. Litta, and P. Ruffolo. The Lemlat 3.0 Package for Morphological Analysis of Latin. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, 2017.

M. Passarotti, F. Mambrini, G. Franzini, F. M. Cecchini, E. Litta, G. Moretti, P. Ruffolo, and R. Sprugnoli. Interlinking Through Lemmas. The Lexical Collection of the LiLa Knowledge Base of Linguistic Resources for Latin. *Linguistic Studies and Essays*, 58(1):177–212, 2020.

M. Passarotti, E. Pellegrini, M. Litta, and F. M. G. Moretti. The Two Approaches to Word Formation in the LiLa Knowledge Base of Latin Resources. *Resources and Tools for Derivational Morphology (DeriMo 2021)*, page 105, 2021.

M. C. Passarotti, F. M. Cecchini, G. Franzini, E. Litta, F. Mambrini, and P. Ruffolo. The LiLa Knowledge Base of Linguistic Resources and NLP Tools for Latin. In *LDK (Posters)*, pages 6–11, 2019.

P. Poccetti, D. Poli, and C. Santini. *Una Storia Della Lingua Latina*. Carocci, 1999.

J. Pouget-Abadie, D. Bahdanau, B. Van Merrienboer, K. Cho, and Y. Bengio. Overcoming the Curse of Sentence Length for Neural Machine Translation using Automatic Segmentation. In *roceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 78–85. Association for Computational Linguistics, 2014.

Python Software Foundation. Python. `https://www.python.org/doc/`, 2001-2022. [Online; accessed 09-October-2022].

P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, 2020.

H. Robinson. *The Zurich Letters: Comprising the Correspondence of Several English Bishops and Others, with Some of the Helvetian Reformers, During the Early Part of the Reign of Queen Elizabeth*, volume 3. University Press, 1846.

G. G. Şahin and M. Steedman. Data Augmentation via Dependency Tree Morphing for Low-Resource Languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009. Association for Computational Linguistics, 2019.

T. Schiess and Badische Historische Kommission. *Briefwechsel der Brüder Ambrosius und Thomas Blaurer 1509-1548*. Ernst Fehsenfeld, 1908.

S. Schulz and M. Keller. Code-Switching Ubique Est - Language Identification and Part-of-Speech Tagging for Historical Mixed Text. In *Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 43–51. Association for Computational Linguistics, 2016.

H. Schwenk, K. Tran, O. Firat, and M. Douze. Learning Joint Multilingual Sentence Representations with Neural Machine Translation. *ACL Workshop, Repl4NLP*, 2017.

H. Schwenk, V. Chaudhary, S. Sun, H. Gong, and F. Guzmán. WikiMatrix: Mining 135M Parallel Sentences in 1620 Language Pairs from Wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361. Association for Computational Linguistics, 2021.

R. Sennrich and B. Zhang. Revisiting Low-Resource Neural Machine Translation: A Case Study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221. Association for Computational Linguistics, 2019.

R. Sennrich, B. Haddow, and A. 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. Association for Computational Linguistics, 2016.

H. Shi, K. Livescu, and K. Gimpel. Substructure Substitution: Structured Data Augmentation for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3494–3508. Association for Computational Linguistics, 2021.

S. Singh and A. Mahmood. The NLP Cookbook: Modern Recipes for Transformer Based Deep Learning Architectures. *IEEE Access*, 9:68675–68702, 2021.

P. Sountsov and S. Sarawagi. Length Bias in Encoder Decoder Models and a Case for Global Conditioning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1525. Association for Computational Linguistics, 2016.

R. Sprugnoli, M. Passarotti, C. Flavio Massimiliano, M. Fantoli, and G. Moretti. Overview of the Evalatin 2022 Evaluation Campaign. In *Proceedings of the Second Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA 2022), Language Resources and Evaluation Conference (LREC 2022)*, pages 183–188, 2022.

F. Stahlberg. Neural Machine Translation: A Review. *Journal of Artificial Intelligence Research*, 69:343–418, 2020.

M. Stoeckel, A. Henlein, W. Hemati, and A. Mehler. Voting for POS Tagging of Latin Texts: using the Flair of FLAIR to Better Ensemble Classifiers by Example of Latin. In *Proceedings of 1st Workshop on Language Technologies for Historical and Ancient Languages*, pages 130–135, Marseille, 2020.

M. Straka. UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium, 2018. Association for Computational Linguistics.

M. Straka, J. Náplava, J. Straková, and D. Samuel. RobeCzech: Czech RoBERTa, a Monolingual Contextualized Language Representation Model. In *International Conference on Text, Speech, and Dialogue*, pages 197–209. Springer, 2021.

J. Su, J. Zeng, D. Xiong, Y. Liu, M. Wang, and J. Xie. A Hierarchy-to-Sequence Attentional Neural Machine Translation Model. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(3):623–632, 2018.

B. Thompson and P. Koehn. Vecalign: Improved Sentence Alignment in Linear Time and Space. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1342–1348, 2019.

J. Tiedemann. OPUS–Parallel Corpora for Everyone. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation: Projects/Products*, 2016.

H. N. Tien and H. N. T. Minh. Long Sentence Preprocessing in Neural Machine Translation. In *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*, pages 1–6. IEEE, 2019.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you Need. *Advances in Neural Information Processing Systems*, 30, 2017.

M. Volk, L. Fischer, P. Scheurer, R. Schwitter, P. Ströbel, and B. Suter. Nunc Profana Tractemus. Detecting Code-Switching in a Large Corpus of 16th Century Letters. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2901–2908. LREC, European Language Resources Association, 2022a.

M. Volk, P. Scheurer, B. Schroffenegger, and R. Müller. Bullinger Digital. `https://www.bullinger-digital.ch/about`, 2022b. [Online; accessed 10-October-2022].

X. Wang, M. Utiyama, and E. Sumita. Online Sentence Segmentation for Simultaneous Interpretation using Multi-Shifted Recurrent Neural Network. In *Proceedings of Machine Translation Summit XVII: Research Track*, pages 1–11, 2019.

A. Way and M. Hearne. On the Role of Translations in State-of-the-Art Statistical Machine Translation. *Language and Linguistics Compass*, 5(5):227–248, 2011.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

M. Xia, X. Kong, A. Anastasopoulos, and G. Neubig. Generalized Data Augmentation for Low-Resource Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5786–5796. Association for Computational Linguistics, 2019.

O. Zennaki, N. Semmar, and L. Besacier. Utilisation des Réseaux de Neurones Récurrents pour la Projection Interlingue d'Étiquettes Morpho-Syntaxiques à Partir d'un Corpus Parallèle. In *Proceedings of TALN 2015*, pages 213–220, Caen, France, 2015.

J. Zhang and T. Matsumoto. Corpus Augmentation by Sentence Segmentation for Low-Resource Neural Machine Translation. *CoRR*, abs/1905.08945, 2019.

B. Zoph, D. Yuret, J. May, and K. 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. Association for Computational Linguistics, 2016.

Lebenslauf

Persönliche Angaben

Sabrina Brändle
Rebbergstrasse 21B
8422 Pfungen
sabrina.braendle@uzh.ch

Schulbildung

seit 2020	Master-Studium Digitale Linguistik an der Universität Zürich
2019-2021	Master-Studium Vergleichende germanische Sprachwissenschaft und Slavische Sprachwissenschaft/Literaturwissenschaft an der Universität Zürich
2016-2019	Bachelor-Studium Vergleichende germanische Sprachwissenschaft und Slavische Sprachwissenschaft/Literaturwissenschaft an der Universität Zürich

Berufliche und nebenberufliche Tätigkeiten

seit März 2022	IT und Sprachtechnologie beim Schweizerischen Idiotikon
Mai-Juli 2020	Human Resources Praktikum Pflegezentren Stadt Zürich
Oktober 2013-Juli 2017	Sachbearbeitung Leistungen SWICA AG