Universität Zürich[UZH]

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Arts
der Philosophischen Fakultät der Universität Zürich

Part-of-Speech Tagging on 16th Century Latin

Verfasserin: Elina Stüssi
Matrikel-Nr: 21-722-202

Betreuer: Dr. Phillip Ströbel

Institut für Computerlinguistik

Abgabedatum: 01.12.2023

Abstract

Part-of-speech (POS) tagging is a fundamental aspect of natural language processing (NLP) and its capability extends beyond basic linguistic tasks. However, existing taggers trained primarily on Classical Latin have limitations when applied to texts from different eras due to the historical evolution of the language. This thesis examines the performance evaluation of pretrained taggers and explores the potential of large language models (LLMs), exemplified by GPT, in conducting POS tagging. The investigation encompasses 16th century epistolary Latin sourced from the Bullinger Digital project and data extracted from five Latin treebanks. Employing a self-created gold standard, I compare tagger accuracy on the Bullinger data and employ fine-tuning on GPT models with the aim of enhancing accuracy. The findings illustrate the viability of LLMs as taggers, which demonstrate competitive performance compared with pretrained models. Further accuracy improvements are discernible through fine-tuning, with the method of prompting emerging as a pivotal factor exerting a substantial influence on results. This research contributes to the advancement of POS taggers, fostering their adaptability across various epochs for comprehensive content analysis. Additionally, it establishes a foundation for leveraging LLMs in specific NLP tasks.

Zusammenfassung

Part-of-Speech (POS) Tagging ist ein grundlegender Aspekt der natürlichen Sprachverarbeitung (NLP) und geht weit über grundlegende linguistische Aufgaben hinaus. Bestehende Tagger, die hauptsächlich auf klassischem Latein trainiert sind, haben jedoch aufgrund der historischen Entwicklung der Sprache Einbussen bei der Genauigkeit, wenn sie auf Texte aus verschiedenen Epochen angewendet werden. Diese Arbeit untersucht die Leistung von bestehenden Taggern und erforscht das Potenzial grosser Sprachmodelle (LLMs), wie beispielsweise GPT, bei der Durchführung von POS Tagging. Die Arbeit umfasst Latein aus dem 16. Jahrhundert, das aus dem Bullinger Digital Projekt stammt, sowie Daten aus fünf lateinischen Treebanks. Ich vergleiche die Tagger-Genauigkeit anhand der Bullinger-Daten unter Verwendung eines selbst erstellten Goldstandards und wende Fine-Tuning auf GPT-Modellen an, um die Genauigkeit zu verbessern. Die Ergebnisse zeigen die Nutzbarkeit von LLMs als Tagger, die im Vergleich zu bestehenden POS Taggern eine wettbewerbsfähige Leistung aufweisen. Weitere Genauigkeitsverbesserungen sind durch Fine-Tuning erkennbar, wobei sich die Prompting-Strategie als entscheidender Faktor erweist, der einen erheblichen Einfluss auf die Ergebnisse ausübt. Diese Forschung trägt zur Weiterentwicklung von POS Taggern bei und fördert deren Anpassungsfähigkeit über verschiedene Epochen hinweg für eine umfassende Inhaltsanalyse. Darüber hinaus legt sie einen Grundstein für die Nutzung von LLMs in spezifischen NLP-Aufgaben.

Acknowledgement

I am immensely grateful to all those who contributed to the completion of this bachelor's thesis. Their support and inspiration have been truly invaluable to me.

I would like to express my gratitude to my supervisor, Phillip Ströbel, for his consistent support. Whenever I encountered technical issues or had other questions, he provided guidance and assistance. His valuable input and support were instrumental throughout the entire process.

I also extend my thanks to the Department of Computational Linguistics and the contributors of the Bullinger Digital project for granting me permission to utilize their resources.

A heartfelt thank you goes to my former Latin teacher, Urs Schwarz, for his assistance in creating the gold standard.

Of course, I want to express my deepest appreciation to my family and friends for their unwavering belief in me and their support in every way possible.

Without the support, guidance, and encouragement of all of you, this journey would not have been feasible. Thank you.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CLTK	The Classical Language Toolkit
DL	Deep Learning
GPU	Graphics Processing Unit
GS	Gold Standard
IQR	Interquartile Range
ITTB	Index Thomisticus Treebank
LiLa	Linking Latin
LLCT	Late Latin Charter Treebank
LLM	Large Language Model
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
OCR	Optical Character Recognition
POS	Part-Of-Speech
PROIEL	Pragmatic Resources in Old Indo-European Languages
RDF	Resource Description Framework
SCRDR	Single Classification Ripple-Down Rule
UD	Universal Dependencies
UPOS	Universal Part-Of-Speech Tags
XML	eXtensible Markup Language
XPOS	language-specific POS tags

1 Introduction

1.1 Motivation

Understanding the essence of parts-of-speech (POSs) is vital in linguistic analysis [Jurafsky and Martin, 2019]. POS tagging, which discerns whether a word functions for example as a noun or a verb, contributes significantly to parsing, establishing its fundamental role in language analysis.
Among historical languages within the Universal Dependencies treebanks[1], Latin boasts the most extensive dataset, particularly in Classical Latin. However, significant portions of Latin literature lack syntactic analysis [Nehrdich and Hellwig, 2022]. POS tagging in historical texts presents unique hurdles due to spelling variations and the absence of extensive datasets, particularly evident in 16th century Latin compared to modern or resource-rich languages like English [Schmid, 2019].
Despite its historical importance in scientific, religious, and communication domains, Latin is classified as a low-resource language due to the lack of digitized texts and annotations, stemming from its status as an extinct language [Hedderich et al., 2021]. Projects like Bullinger Digital[2] have contributed 16th century Latin texts for analysis, such as the correspondence of Heinrich Bullinger [Fischer et al., 2022].
Most POS taggers for Latin rely on Classical Latin texts, yet nuances in 16th century epistolary Latin pose challenges for these systems. To exemplify these nuances, consider the tagging of a sentence from the Bullinger corpus in Figure 1, using different taggers. The Bullinger corpus is a subset of the Bullinger letters and will be introduced later in Section 3.2.1. The English translation for this example is presented in cursive beneath the Latin sentence. The first line represents the established gold standard (GS), the second is tagged by LatinCy (LC), the third by RDRPOSTagger (RDR), and the last row by GPT-4.

The example sentence already shows disparities in tagging among the various taggers. The RDRPOSTagger erroneously classified punctuation and the name “Eras-

[1] `https://universaldependencies.org`
[2] `https://www.bullinger-digital.ch`

	Dominus	Erasmus	plurimam	salutem	tibi	adscribere	iussit	.
en:	*Lord*	*Erasmus*	*has*	*ordered*	*me*	*to write*	*many greetings*	*to you* .
GS:	NOUN	PROPN	ADJ	NOUN	PRON	VERB	VERB	PUNCT
LC:	NOUN	PROPN	ADJ	NOUN	PRON	NOUN	VERB	PUNCT
RDR:	NOUN	VERB	ADJ	NOUN	PRON	VERB	VERB	VERB
GPT-4:	NOUN	PROPN	ADJ	NOUN	PRON	VERB	VERB	PUNCT

Figure 1: A Sentence from the Bullinger corpus tagged with different taggers

mus" as verbs. Additionally, LatinCy labeled "adscribere" as a noun, whereas the RDRPOSTagger and GPT-4 identified it correctly as a verb. Connecting the tagging similarities identified between GPT-4 and the GS for this sentence, it underscores the potential of large language models (LLMs). These models offer a promising avenue to mitigate the inaccuracies exemplified here, showcasing their capacity to enhance accuracy in language processing tasks. Emphasizing the prowess of Large Language Models (LLMs) in natural language processing (NLP) due to their capacity to learn from extensive datasets [Radford et al., 2019], the primary goal of this thesis is to assess their effectiveness in the specific context of POS tagging. Additionally, I plan to fine-tune a LLM for this purpose, with the explicit aim of investigating potential accuracy improvements.

Resolving the detected issue with POS tagging holds great potential in various fields. Enhanced accuracy in POS tagging for historical languages stands to profoundly impact linguistic analysis, aiding historians, linguists, and researchers in analyzing ancient texts.

The technological advances resulting from these efforts go beyond the field of historical linguistics. Enhancing POS tagging algorithms can spur the creation of sophisticated language processing instruments. These developments promise increased accuracy and effectiveness in NLP applications, particularly for low-resource languages and historical documents.

1.2 Research Questions

This thesis investigates the effectiveness of different POS taggers in handling 16th century Latin texts and explores the potential use of generative pretrained transformer (GPT) models in POS tagging tasks within the above mentioned historical linguistic context.

One focus lies on comparing various pretrained POS taggers in order to determine

which ones are best suited for processing data pertaining to the 16th century Latin. Hence, the first research question is formulated as follows:

1. How accurately do pretrained POS taggers perform on the 16th century Bullinger data?

This inquiry aims to evaluate their accuracy, adaptability to historical linguistic nuances, and overall robustness when applied to texts from this time period. An essential question pertains to the viability and limitations of employing LLMs, exemplified by GPT models, as effective tools for POS tagging in the context of historical Latin texts. The second research question is defined as follows:

2. Can language models such as GPT models be effectively used as POS taggers?

Furthermore, the investigation examines the impact of fine-tuning LLMs, particularly GPT models, to enhance their performance in POS tagging for historical Latin data. Thus, the third research question is proposed:

3. Can fine-tuning significantly increase the accuracy of GPT models in POS tagging tasks?

This inquiry aims to determine whether fine-tuning these models can substantially increase accuracy and resolve challenges encountered in historical language analysis. Lastly, I endeavor to assess the generalizability of POS taggers and LLMs across diverse genres, authors, and linguistic variations within the Latin corpora. For this purpose, the final research question is formulated:

4. Is there a notable discrepancy in the performance of POS taggers when applied to Classical Latin data?

By addressing these four questions, I aim to ascertain the adaptability of the POS taggers to various text eras within the Latin corpus.

1.3 Thesis Structure

The structure of the thesis is as follows: Chapter 2 examines both the theoretical underpinnings and linguistic background and presents an overview of relevant prior studies that form the foundation of my research. In Chapter 3, I present the data, methodologies, and tools employed in my experiments, along with a comprehensive exposition of the experimental setup. The outcomes of these experiments are then stated in Chapter 4. Chapter 5 gives an analysis of the results obtained and brings them in the context of the research questions. It includes a discussion, a comparative

analysis, and an acknowledgment of potential limitations encountered during the study. Finally, Chapter 6 unifies the disparate components of my work to give a summary of my findings. It encapsulates final conclusions drawn from this thesis while highlighting avenues for future research.

2 Background

In this chapter, I aim to provide an understanding of the theoretical and technical foundations pertinent to my research questions. Section 2.1.1 briefly traces the evolution of the Latin language, Section 2.1.2 defines the term "low resource" in the context of languages, and Section 2.1.3 elucidates the concept of POS ambiguity, a key factor reducing the accuracy in POS tagging. Next, Section 2.2 examines the technical aspects such as POS tagging and LLMs. Finally, I introduce various works aligned with my research questions in Section 2.3.

2.1 Linguistic Background

I will provide an introduction of the linguistic background relevant to my thesis in this section. In particular, this covers the evolution of the Latin language and is consistent with the thesis's analysis of discrepancies in performance between Classical Latin and Latin of the 16th century. The section will also discuss Latin's designation as a low-resource language and explain POS ambiguity, which has a big impact on the accuracy in the context of the experiments.

2.1.1 Evolution of the Latin language

The Latin language has experienced significant changes over the course of its long historical evolution, creating a challenging environment for scholars. Changes in case endings and lexical transformations are part of this linguistic journey. This is most evident in the transition from Old Latin to Classical Latin, where endings like *-om* and *-os* became *-um* and *-us*, respectively [Bennett, 1907]. An example of this transformation is the Old Latin word *Romanom* (meaning "Roman" in English), which evolved into *Romanum* in Classical Latin. These modifications involve more than just word replacements; they also involve adjustments to meanings and structures. Due to these modifications, scholars are faced with a range of linguistic variances, from regional to global forms, the contrast between extinct and extant

Latin variants, and the differences between Vulgar Latin and its literary equivalent [Poccetti et al., 1999].
Latin, despite its decline since 1800, has persisted in certain domains like the Catholic Church and classical studies. Over 99.99% of extant Latin texts originate from later periods, particularly the Renaissance and Neo-Latin eras [Leonhardt, 2009]. Humanists have made significant efforts to purify written Latin from medieval orthographic alterations, advocating for a return to Classical Latin spellings and distinguishing between "t" and "c," and thus distinguishing between "eciam" (meaning "and also" in English) and "etiam", which were interchangeable for medieval scribes [Leonhardt, 2009].
Understanding the linguistic evolution in Latin is crucial, especially contextualizing the Bullinger letters, composed in 16th century epistolary Latin. The disparity in Latin utilized within this corpus compared to other corpora poses a challenge for language processing systems. Given that these systems are predominantly trained on datasets from distinct temporal periods, their performance in analyzing texts within this corpus may exhibit considerable divergence from those texts derived from alternate epochs.

2.1.2 Low-resource languages

A low-resource language is a language with little or no labeled data. The delineation of "low resource" depends on the specific NLP task and the associated threshold, as elaborated by Hedderich et al. [2021], who explore the multifaceted nature of the definition of "low-resource". Creating task-specific labels often demands manual annotation, a process hindered by feasibility constraints due to its time and cost-intensive nature [Hedderich et al., 2021].
Despite the existence of texts, Latin is regarded as a low-resource language. One major hurdle in the analysis of Latin stems from the absence of native speakers, rendering interactions and learning from them impossible. This absence poses significant difficulties in creating a gold standard (GS), a process notably more labor-intensive and error prone than that for modern languages. Nonetheless, Latin benefits from a wealth of linguistic expertise derived from its extensive historical legacy, offering substantial aid in overcoming these obstacles [McGillivray, 2015].
The challenges associated with Latin as a low-resource language significantly impact this thesis. Specifically, these hurdles influenced the decision to create a task-specific GS due to the scarcity of labeled data.

2.1.3 POS ambiguity

During POS tagging, accurately assigning tags to wordforms can be an intricate process, particularly for Latin. Its linguistic subtleties, such as the presence of participles, namely verbs that often take on the role of an adjective in English translation, pose significant challenges. Participles are a distinctive category of verbal adjectives, blending characteristics of both verbs and adjectives. Consequently, determining which POS tag to assign, specifically verb or adjective in Latin, relies heavily on contextual cues, necessitating a consensus on their classification [Embick, 2000].
Moreover, discerning whether a (modal) verb operates as an auxiliary verb can be challenging, with a definitive consensus on their treatment lacking[1]. These complexities directly impact the accuracy and reliability of POS tagging systems when applied to Latin texts. In the context of this research, where the evaluation and utilization of pretrained taggers and LLMs in handling 16th century Latin texts are central, addressing these intricacies becomes crucial.
The decisions made regarding the treatment and tagging of complex linguistic instances significantly influence the accuracy of the taggers applied to historical Latin texts. And thus, establishing conventions and guidelines, as presented in Appendix A, is essential for consistency and accuracy in creating a GS for this thesis.

2.2 Technical Background

In this section, I will provide an introductory overview of the technical framework that underpins my thesis. This includes delving into the concept of POS tagging and the various existing approaches, and elucidating the terminology and concept of LLMs.

2.2.1 Part-of-speech tagging

POS tagging, which involves assigning specific tags such as noun or verb to each word in a text, informs word roles, aiding in sentence comprehension, syntactic analysis for parsing, entity labeling, and coreference resolution [Jurafsky and Martin, 2019]. Disambiguating words with multiple potential tags relies on context and inherent word characteristics [Jurafsky and Martin, 2019], a concept tracing back to Dionysius Thrax's work around 100 B.C., which laid the groundwork for POS

[1] See `https://universaldependencies.org/u/feat/VerbType.html`

tagging by delineating eight enduring POS concepts [Jurafsky and Martin, 2019]. Various languages utilize different POS tag sets tailored to their specific features. For highly inflected languages like Greek and Latin, these tag sets tend to be more complex [Petrov et al., 2011]. In this thesis, I used the universal POS tags (UPOS tags) for consistency[2].

Automation of POS tagging has been a focal point due to the tedium associated with manual assignment. Different NLP methods, including rule-based [Borin, 2000], stochastic, transformation-based, and machine learning (ML) approaches [Chiche and Yitagesu, 2022], have been explored for POS tagging.

Rule-based methods offer simplicity but lack adaptability to new expressions or language changes [Chiche and Yitagesu, 2022] due to their reliance on predefined linguistic rules and patterns. Conversely, stochastic models calculate word POS probabilities based on context, effectively managing ambiguities but demanding substantial training data. Transformation-based tagging, such as the Brill tagger, iteratively refines tags based on identified errors and proves to be efficient and language-independent. ML approaches, using decision trees and neural networks, leverage annotated data for robust performance [Chiche and Yitagesu, 2022].

Despite advancements, POS tagging remains challenging, necessitating high accuracy, minimal false positives, and adept handling of contextual ambiguities. Current strategies, empowered by increased GPU capabilities, lean towards deep learning (DL) and ML methods. ML utilizes feature engineering from corpora, while DL captures intricate features directly from data [Chiche and Yitagesu, 2022].

2.2.2 Large language models

LLMs excel in comprehending and generating language due to their extensive training on massive datasets, learning billions of parameters through significant computational resources [Hadi et al., 2023].

Primarily built on transformer architectures, LLMs function as autoregressive models predicting subsequent tokens from input text[Liu et al., 2024]. While fine-tuning was conventional for task-specific adaptation, newer large-scale models like GPT-3 show comparable performance with engineered prompts. These models are believed to grasp syntax, semantics, and the fundamental structure of human language from corpora. However, they may inherit biases and inaccuracies embedded in these datasets [Liu et al., 2024].

[2] `https://universaldependencies.org/u/pos/`

2.3 Related Work

Having discussed the key concepts underlying this thesis, I finally outline some related work that will be helpful in understanding its broader context.
Despite Latin not being extensively studied in NLP, existing literature offers various methodologies that enhance its efficient processing. Initiatives such as the inaugural Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA) in 2020 marked a significant step in this direction. This workshop focused on uniting scholars who invested in developing and employing language technologies tailored for historically documented languages, including Latin [Sprungoli and Passarotti, 2020]. The EvaLatin[3] campaign within LT4HALA was instrumental, being the first dedicated effort to assess NLP tools specifically designed for Latin. Given the great diversity across Latin texts spanning two millennia, EvaLatin examined lemmatization and POS tagging, assessing how genre and diachronic variations influenced tool performance [Sprungoli and Passarotti, 2020]. Understanding the methodologies and challenges outlined in EvaLatin aligns with the goals of this thesis, offering nuanced insights into evaluating NLP tools on historical language data across diverse temporal contexts.
Submissions within EvaLatin, such as LSTMVoter proposed by Stoeckel et al. [2020] and the UDPipe2-based system by Straka and Straková [2020], showcased advancements in lemmatization and POS tagging techniques tailored for historical Latin texts, expanding the methodological breadth available for analysis. Achieving notable success, Straka and Straková [2020] attained an accuracy of 91.01% on the cross-time lemmatization task, and an even more impressive accuracy of 96.19% on the classical lemmatization task. In the classical POS tagging task, an accuracy of 96.74% was attained, while on the cross-time task, they achieved an accuracy of 87.69% [Straka and Straková, 2020].
Additionally, the Linking Latin (LiLa) Knowledge Base of Linguistic Resources project, introduced by Passarotti et al. [2019], established a robust lexical foundation for Latin using the Resource Description Framework (RDF), fostering synergy between textual and lexical resources [Pellegrini et al., 2023; Fantoli et al., 2022]. LiLa's Text Linker tool offers POS tagging capabilities and interconnected lexical information in RDF format, facilitating comprehensive linguistic analysis. Figure 2 illustrates an example entry in the Text Linker for the Latin verb "dabo" (meaning "I will give" in English).

Further advancements in this domain include Latin BERT Bamman and Burns [2020], a Latin-tailored version of BERT Devlin et al. [2019] trained across Classi-

[3] `https://circse.github.io/LT4HALA/2020/EvaLatin`

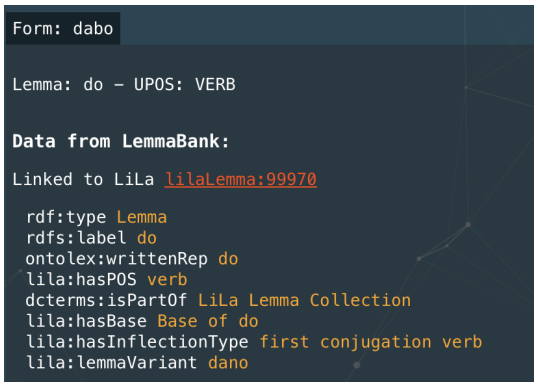


Figure 2: Example of an entry of the Text Linker

cal to contemporary sources. Latin BERT demonstrates exceptional POS tagging accuracy without task-specific training, achieving high accuracy on various Latin datasets [Bamman and Burns, 2020].
Additionally, while studies such as the investigation into low-resource languages like Hongkonese highlight the nuanced strengths of GPT models, showcasing their superior performance over existing POS taggers, there remains a notable gap in research specifically examining LLMs as POS taggers[4]. Drawing from these insights, the methodologies applied in this thesis to assess language models and POS taggers within the domain of historical Latin texts are shaped by these previous studies.

[4]See "GPT-4 is a very good Hongkongese POS Tagger" `https://medium.com/@kyubi_fox/gpt-4-is-a-very-good-hongkongese-pos-tagger-feeae1b44868`

3 Methods

This thesis builds upon the foundational knowledge established in the background and related work sections, emphasizing the challenges of POS tagging in historical texts and the limitations of traditional taggers. Notably, there's a gap highlighted in leveraging contemporary language models, like GPT, for POS tagging historical texts. Addressing this gap, this thesis focuses on assessing the effectiveness of such models, both pretrained and fine-tuned, on Bullinger's 16th century writings. The methods chapter outlines the experimental design, data collection, and evaluation metrics, aiming to explore the capabilities and limitations of these models in handling language structures in historical linguistic studies. It covers the acquisition, curation, and tools used alongside the data sources, focusing on POS tagging, datasets, POS tagging models, and the experimental setup outlined in Sections 3.1, 3.2, 3.3, and 3.4 respectively. All code utilized in these experiments is accessible through my GitHub repository[1].

3.1 Part-of-speech Tagging

This thesis centers on POS tagging, utilizing the UPOS tags from the Universal Dependencies Project[2]. The UPOS system offers a standardized set of tags for POS tagging in NLP. These 17 UPOS tags, given in Table 1 along with their meanings, aim for cross-linguistic applicability, enabling comparative linguistic analyses across various languages and tagging systems. However, I excluded the UPOS tag PUNCT from the GS in this thesis due to errors produced by RDRPOSTagger when processing punctuation symbols in the used data. However, since this is only a single tag, taggers should be able to learn this tag very quickly and be able to tag the punctuation correctly. Not all tags appear in all the taggers used, as discussed in more depth in Section 4.4.

[1] `https://github.com/Eljuanina/BA`

[2] See `https://universaldependencies.org/u/pos/`

ADJ	Adjective	PART	Particle
ADP	Adposition	PRON	Pronoun
ADV	Adverb	PROPN	Proper noun
AUX	Auxiliary	PUNCT	Punctuation
CCONJ	Coordinating conjunction	SCONJ	Subordinating conjunction
DET	Determiner	SYM	Symbol
INTJ	Interjection	VERB	Verb
NOUN	Noun	X	Other
NUM	Numeral		

Table 1: UPOS tags and their meanings

3.2 Corpus and Training Data

In this subsection, I discuss the data and treebanks used in the experiments performed in this thesis. I used samples from the Bullinger Digital corpus as well as texts from five treebanks, namely ITTB [Cecchini et al., 2018] [Passarotti and Dell'Orletta, 2010], LLCT [Korkiakangas, 2021a], UDante [Cecchini et al., 2020b], PROIEL [Eckhoff et al., 2018], and Perseus [Bamman and Crane, 2006]. An overview of the different datasets used in this thesis is displayed in Table 2.

Dataset	**time**	**# of sentences**	**# of token-tag pairs**
Bullinger	c. 16	200	3664
ITTB	c. 13	24,876	420,672
LLCT	c. 8 – c. 10	8,173	218,223
PROIEL	c. 1 BCE – c. 4	11,851	110,774
UDante	c. 13 – c. 14	1,157	38,086
Perseus	c. 1 BCE – c. 4	4,236	68,283
Total		**50,493**	**859,702**

Table 2: Overview of different datasets (c. = century).

3.2.1 Bullinger corpus

The Bullinger data, a subset of 200 sample sentences from the Bullinger Digital project[3] dataset, forms the basis of this research, referred to as the "Bullinger cor-

[3] `https://www.bullinger-digital.ch`

pus" in this thesis. It stems from the extensive correspondence of Heinrich Bullinger, a Swiss reformer living from 1504 to 1575, encompassing around 12,000 preserved letters from the 16th century [Fischer et al., 2022]. These letters, spanning over 50 years and involving communication with more than 1,000 correspondents across diverse social strata and geographical regions. They offer insights into the political, theological, economic, and societal aspects of the early modern period. While multilingual, this thesis specifically focuses on the Latin sections within this corpus [Bullinger Digital, 2023]. The Bullinger corpus consists of both edited and unedited text editions, incorporating 100 samples from the Heinrich Bullinger Briefwechsel (HBBW) edition and another 100 samples from unedited transcriptions.
The digitization process involved scanning the original letters followed by optical character recognition (OCR) software to convert them into digital text [Ströbel et al., 2023]. These digitized letters were then structured into a standardized XML format, with individual XML files representing each letter. These files contain metadata such as sender, recipient, date, and a German summary, maintaining a sentence-per-line structure with language attribute tags [Bullinger Digital, 2023].

3.2.2 Treebanks

This section describes the five utilized treebanks, which served as training material for the POS tagging models used in this work and as data to fine-tune the GPT models. I further elaborate on the taggers in Section 3.3, and in Section 3.4.3 I explain the process of fine-tuning the models.

3.2.2.1 ITTB

The Index Thomisticus project, widely regarded as a pioneering endeavor in the field of computational linguistics, was spearheaded by Father Roberto Busa SJ in the second half of the 1940s. Its aim was to compile the complete Latin works of Thomas Aquinas, encompassing 118 texts by Aquinas himself and an additional 61 texts by related authors. This corpus totals approximately 11 million words, meticulously annotated with morphological tagging and lemmatization.
An offshoot of this project, the Index Thomisticus treebank (ITTB), initiated planning in the early 1970s under Father Busa's guidance. Its purpose was to incorporate morphosyntactic disambiguation and sentence-level syntactic annotation into the Index Thomisticus corpus. Beyond these primary goals, the ITTB project also aims to create diverse and sophisticated language resources specifically tailored for Latin [Passarotti, 2019].

In this thesis I used test and training sets retrieved in CoNLL-U format from a GitHub repository[4] comprising 24,876 tagged sentences and totaling 420,627 token-tag pairs.

3.2.2.2 LLCT

I utilized the Universal Dependencies (UD) version of the Late Latin Charter treebank (LLCT), which is an automated conversion of the LLCT2 treebank from the Latin Dependency treebank format to UD format[Korkiakangas, 2021b]. LLCT2, developed from 2016 to 2018, encompasses 521 Early Medieval Latin records (charters) written in Tuscany during the period from A.D. 774 to 897 [Korkiakangas, 2021b]. These charters belong to the legal (documentary) genre, utilizing a non-standard Latin variety that differs from Classical and Medieval Latin in spelling, morphology, and syntax [Korkiakangas, 2021b].
Charters offer rich non-literary content, contain extensive metadata such as scribe names, places and dates, and represent original documents. However, they also pose a challenge because they are formulaic, have a complicated relationship with spoken language (manifesting as a mixture of archaic, misunderstood, and spoken language features), and repeat or lack certain linguistic phenomena[5].
The data used in this thesis was retrieved from a GitHub repository[6]. The dataset encompasses 8,173 tagged sentences, which correspond to 218,223 token-tag pairs in total.

3.2.2.3 UDante

The UDante treebank encompasses all Latin texts authored by Dante Alighieri, a prominent figure in Italian literature and politics during the 13th and 14th centuries, notably known for his *Divina Commedia* [Cecchini et al., 2020a]. This treebank project arose from the absence of syntactic annotations in Dante's texts despite their significance in Italian literature. To commemorate the 700th anniversary of Dante's death, UDante aimed to syntactically annotate all his Latin works according to UD standards [Cecchini et al., 2020b].
UDante focuses on literary Medieval Latin (14th century) and includes works such as "De vulgari eloquentia," which discusses the identification of the best vernacular language in Italy; *Monarchia*, a political treatise on the Empire–Church relation-

[4] `https://github.com/UniversalDependencies/UD_Latin-ITTB/tree/master`
[5] See `https://lila-erc.eu/wp-content/uploads/2019/06/korkiakangas.pdf`
[6] `https://github.com/UniversalDependencies/UD_Latin-LLCT`

ship; and a collection of Dante's mostly politically themed letters [Cecchini et al., 2020a].
The UPOS tagging in UDante was obtained through an automatic conversion from the original DanteSearch[7] corpus tagset, with manual and partial automatic adjustments to align it with UD annotation guidelines [Cecchini et al., 2020a].
The UDante data included in my experiments, obtained from GitHub[8], includes 1,157 tagged sentences, comprising 38,086 token-tag pairs.

3.2.2.4 PROIEL

The Pragmatic Resources in Old Indo-European Languages (PROIEL) project, affiliated with the University of Oslo's Department of Philosophy, Classics, History of Art, and Ideas, focuses on exploring morphological and syntactic structures in ancient Indo-European languages by creating an annotated corpus. This corpus includes the Koine Greek original of the New Testament and its translations into Latin (Jerome's *Vulgata*), Gothic (Wulfila's translation), and Slavic (from *Codex Marianus*) [Eckhoff et al., 2009].
The New Testament was chosen as a primary source for comparative research due to its status as a naturally occurring parallel corpus and its inclusion of some of the oldest and most substantial texts in Germanic, Slavic, and Armenian. Alongside religious texts, the corpus incorporates Latin works including Julius Caesar's *Commentarii de bello Gallico*, Cicero's "Epistulae ad Atticum," and "De officiis," as well as Palladius' "Opus agriculturae" [Eckhoff et al., 2009].
The PROIEL treebank's creation underwent various linguistic annotation stages, including sentence boundary verification and correction, followed by manual assignment of morphological data, often guided by system-generated suggestions and predefined rules [Eckhoff et al., 2009].
The PROIEL dataset used in this thesis, obtained from GitHub[8], comprises 11,851 tagged sentences, totaling 110,774 token-tag pairs.

3.2.2.5 Perseus

The Perseus dataset, specifically version 2.1, contains semi-automatically annotated texts. Each word in this dataset is detailed with several attributes, including a 9-character POS attribute. The positional significance of each character within this

[7] `https://dantesearch.dantenetwork.it`

[8] `https://github.com/UniversalDependencies/UD_Latin-UDante/blob/master`

[8] `https://github.com/proiel/proiel-treebank/`

string determines the word's POS [Babeu, 2019].
However, the dataset did not initially use UPOS tags, necessitating a mapping process to align the tags used in Perseus with the UPOS tagset before proceeding with the analysis. The version 2.1 dataset includes works such as Cicero's *In Catilinam*, Ovid's "Metamorphoses," and Augustus' "Res Gestae."[9]
I retrieved this dataset from the GitHub repository[10]. All files were utilized except those containing Caesar's *Commentarii de Bello Gallico* and Jerome's *Vulgata*, as these were already present in the PROIEL dataset (see Section 3.2.2.4). The shortened Perseus dataset employed in this thesis encompasses 4,236 tagged sentences, totaling 68,283 token-tag pairs.

3.3 POS Tagging Models

In this thesis, I evaluated the performance of various POS tagging models on 16th century epistolary Latin sourced from the Bullinger letters. Additionally, I assessed their effectiveness on the training data used for these models. My comparative analysis involved five established POS tagging models: LatinCy, CLTK, UDPipe, RDRPOSTagger, and TreeTagger. Attempts to include RNNTagger were discontinued due to challenges in aligning its output tags with the UPOS tagset. Moreover, I examined the capabilities of LLMs for POS tagging, employing the GPT-3.5-Turbo and GPT-4 models. Additionally, I fine-tuned eight models based on GPT-3.5-Turbo with different volumes of training examples. Detailed descriptions of each utilized tagger are presented in the following subsections.

3.3.1 LatinCy

LatinCy, a spaCy-based toolkit for Latin NLP, emerged in 2023 as a set of trained general purpose "core" pipelines [Burns, 2023]. The LatinCy toolkit includes three main models: "la_core_web_sm," "la_core_web_md," and "la_core_web_lg." The "la" prefix signifies Latin, while "core" denotes a comprehensive pipeline encompassing various NLP components and the "web" label indicates their training data origin. The larger "md" and "lg" models utilize subword vectors [Burns, 2023], enabling them to grasp extensive vocabularies and generalize beyond their training data [Ács et al., 2021].

[9] A complete list of the works is provided here `https://universaldependencies.org/treebanks/la_perseus/index.html`

[10] `https://github.com/PerseusDL/treebank_data/tree/master/v2.1/Latin/texts`

The LatinCy models underwent training using diverse sources such as Latin Universal Dependencies treebanks [Celano, 2019], Wikipedia, the cc100-latin corpus [Ströbel, 2023], and NER datasets from the Herodotos project [Erdmann et al., 2019]. The top-performing LatinCy model achieves an impressive 97.41% accuracy for POS tagging [Burns, 2023]. In this thesis, the focus was specifically on evaluating the largest model, namely the "la_core_web_lg" model.

3.3.2 CLTK

The Classical Language Toolkit (CLTK), introduced in 2014 by Johnson et al. [2021], stands as an open-source Python library designed for NLP tasks concerning ancient and premodern languages, focusing on historically under-resourced languages like Latin and Greek. It offers functionalities for tokenization, lemmatization, POS tagging, and morphological analysis [Burns, 2019]. With preconfigured modular pipelines for 19 languages, including Classical Chinese, Old Norse, Middle High German, Sanskrit, Greek, and Latin[11], CLTK's architecture caters to the unique requirements of premodern languages [Johnson et al., 2021]. In the realm of Latin morphological parsing, CLTK integrates Stanza, a Python package leveraging neural networks for morphological analysis. Stanza utilizes advanced neural network components and PyTorch-based modules, employing a bidirectional LSTM network for its POS tagging [Qi et al., 2020]. CLTK showcases average accuracies of 68% for unigrams, 10% for bigrams, and 75% for trigrams on the Perseus test data. With the 1, 2, 3-gram backoff tagger iterated over 10 times, it reaches an accuracy of 82%, employing a cascading approach to tagging resolution when the primary tagger lacks adequate information[12].

3.3.3 UDPipe

UDPipe2 functions as a comprehensive pipeline for sentence segmentation, tokenization, dependency parsing, lemmatization, and POS tagging [Straka, 2018], employing CoNLL-U format data and pretrained word embeddings for training. With over 50 language models, including Latin and non-Indo-European languages like Arabic, UDPipe builds its models based on Universal Dependencies treebanks, featuring specific ones for Latin: "Latin-ITTB," "Latin-Perseus," and "Latin-PROIEL" [Straka, 2018].

[11] See `https://github.com/cltk/cltk`

[12] See `http://cltk.org/blog/2015/08/02/updated-accuracies-pos-taggers.html`

The system utilizes an artificial neural network for POS tagging, lemmatization, and dependency parsing, employing shared trained word embeddings to evade treebank-specific hyperparameter tuning. For POS tagging, it processes embedded words through a multilayer bidirectional LSTM and a softmax classifier to generate tags. Advanced features encompass residual connections and a dense layer with tanh non-linearity for intricate nonlinear processing [Straka, 2018].
UDPipe2 excelled in the CoNLL 2018 UD Shared Task, achieving top rankings across various evaluations [Straka, 2018]. UDPipe is available for download as a program compatible with all operating systems and can also be imported as a library into programming languages such as Python, Perl, and R. In this thesis, UDPipe2 was utilized as a library within the R programming environment, employing the Latin-PROIEL model. Table 3 presents UDPipe's results for different UPOS tags as documented in Straka [2018].

Model	% accuracy
Latin-ITTB	98.28
Latin-PROIEL	96.75
Latin-Perseus	87.64

Table 3: Accuracies of UDPipe for Latin models

3.3.4 RDRPOSTagger

RDRPOSTagger specializes in POS and morphological tagging, employing a Single Classification Ripple Down Rules (SCRDR) tree, generated via an error-driven method [Nguyen et al., 2014]. Covering approximately 80 languages, it provides pretrained models for UPOS, XPOS, and morphological tagging[13]. Unlike Brill's approach Brill [1995], RDRPOSTagger automatically organizes transformation rules into a structured SCRDR tree, allowing controlled interactions between rules and incorporating new exception rules to rectify errors [Richards, 2009; Nguyen et al., 2014].
The SCRDR tree structure integrates edges labeled as "except" and "if-not," housing rules in a conditional form "if α then β," with α as the condition and β as the conclusion [Richards, 2009]. During evaluation, the case traverses the tree path, culminating at the last node whose rule aligns with the case, resulting in a definitive determination [Nguyen et al., 2014].
Proven successful for English and Vietnamese, RDRPOSTagger's language indepen-

[13]See `https://rdrpostagger.sourceforge.net/`

dence allows adaptation to other languages by providing word lexicons with their most frequent associated tags [Nguyen et al., 2014]. For Latin, three available models showcase slightly varying accuracies[14], outlined in Table 4. In this thesis, I utilized RDRPOSTagger within the R environment, employing the UD_Latin-ITTB model.

Model	% accuracy
UD_Latin-ITTB	96.85
UD_Latin-PROIEL	93.98
UD_Latin-Perseus	84.44

Table 4: Accuracies of RDRPOSTagger for Latin models

3.3.5 TreeTagger

TreeTagger, emerging from the University of Stuttgart's textual corpora (TC) project, excels in annotating text with POS and lemma information across various languages, including German, English, Chinese, Russian, Greek, and Latin. Adaptable to new languages given a lexicon and a tagged training corpus, it remains versatile[15].
There are different parameter files available for download for each supported language in TreeTagger. For Latin, there are two files: one by Gabriele Brandolini and another trained on the ITTB by Marco Passarotti. Initially using the latter, I encountered issues mapping the output tags to the UPOS tagset, prompting a shift to Brandolini's parameter file for TreeTagger evaluation.
Functionally akin to conventional n-gram taggers outlined by Church [1988] and Kempe [1993], TreeTagger models the probability of tagged word sequences. While n-gram taggers rely on maximum likelihood estimation and encounter difficulties with zero frequencies, TreeTagger estimates transition probabilities by utilizing a binary decision tree [Schmid, 1994].
TreeTagger's performance was assessed using data from the English Penn Treebank corpus by employing approximately 2 million training words and 100,000 testing words from a distinct part of the treebank. Two versions, bigram, and trigram, achieved accuracies of around 95.8% and 96%, respectively [Schmid, 1994]. The output of TreeTagger does not use UPOS tags, making it necessary to map the tags into UPOS tags. Below is an example of the unmapped TreeTagger output:

[14]See `https://github.com/datquocnguyen/RDRPOSTagger/tree/master/Models`
[15]See `https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`

```
Augustinum N:acc Augustinus
vero ADV vero|verus
suam POSS suus
probasse V:INF probo
```

3.3.6 RNNTagger

RNNTagger, a tool for POS tagging and lemmatization, supports 49 languages, encompassing both modern and ancient languages like English, German, Latin, and Old Greek. Implemented in Python3 using PyTorch, it was developed by Helmut Schmid, known for implementing TreeTagger (refer to Section 3.3.5). Unlike TreeTagger, RNNTagger aims for higher tagging accuracy and ensures lemmatization for all tokens. However, it tends to be slower, especially without a GPU, and requires larger Python and PyTorch parameter files. In the case of Latin, its training relied on the Index Thomisticus corpus (see Section 3.2.2.1). The POS tagger utilizes bidirectional LSTM networks to process character sequences of words and generate word representations, followed by fully connected layers and softmax operations to predict POS tags on tokenized text using its own tokenizer [Schmid, 2019].
However, in this thesis, RNNTagger, while producing output, did not format tags into UPOS, making it challenging to map them as no manual for mapping was available. Consequently, this tagger was excluded from further evaluation. Below is an example of its unmapped output.

```
Augustinum 11B---D1---augustinus
vero 11B---G---1 verus
suam 11A---D2---suus
probasse 3-JH4------probo
```

3.3.7 GPT

In this thesis, I evaluated two OpenAI-developed LLMs: GPT-3.5-Turbo and GPT-4. GPT-3.5, an advanced version of GPT-3 launched in 2022, utilizes 175 billion parameters and was trained on a vast 570GB text corpus, ranking among the largest language models [OpenAI, 2023]. Operating on a transformer neural network architecture, it excels in tasks like translation, text completion, and question-answering due to its deep comprehension of complex language patterns and nuances [Topal et al., 2021].
GPT-4, a newer model, shares some limitations with previous versions, such as oc-

casional unreliability, potential "hallucinations," and constraints within its context window. Trained on diverse datasets including publicly available internet data and third-party licensed data, GPT-4 was fine-tuned by reinforcement learning based on human feedback [OpenAI, 2023].
While GPT-4 demonstrates superior performance across various tasks and can process visual input data, as of this thesis, only GPT-3.5-Turbo supports fine-tuning with custom training data. Fine-tuning involves updating pretrained weights with task-specific supervised datasets. Although this method yields strong performance on benchmarks, it requires substantial datasets for each task and may lead to overfitting or reliance on irrelevant features in the training data, potentially impacting comparisons with human performance [Brown et al., 2020]. Section 3.4.3 covers the fine-tuning process, while Section 4 details the model evaluations.

3.4 Experimental Setup

In this section, I delve into the setup of various experiments, detailing the establishment of the GS is discussed in Section 3.4.1, the methodology behind the POS tagging process described in Section 3.4.2, and the intricate fine-tuning procedure outlined in Section 3.4.3. These steps were crucial in aligning with my research goals, facilitating the evaluation of pretrained POS taggers and language models on 16th century Latin texts, particularly within the context of Heinrich Bullinger's correspondence.

3.4.1 Gold Standard

The process began by extracting 200 sample sentences from the Bullinger correspondence, forming a smaller Bullinger corpus. I collected 100 sentences from both the editions/hbbw and unedited/transcribed directories within the project's GitHub repository [16]. These samples contained Latin segments from letters, accompanied by metadata like source file, author, year, and sentence number, stored in an XML file. This structure ensured representation across different editions, authors, and years. After confirming diverse representation, I saved the sentences in txt files, tokenized them using the spaCy tokenizer with the "la_core_web_lg" model, and cleaned the text by removing punctuation, including internal parentheses within words.
Subsequently, I employed the UDPipe tagger to assign reference tags to each token in the file. Creating the GS involved meticulously reviewing each token-tag pairing,

[16] `https://github.com/bullinger-digital/bullinger-korpus`

resolving discrepancies between assigned tags and my interpretation of the correct classifications. To ensure reliability, I collaborated with my former Latin teacher, Urs Schwarz, to establish a GS based on agreed tagging principles (Appendix A). Seeking validation, I consulted Dr. Philipp Roelli from the Fachstelle Latein at the University of Zurich, ensuring the sensibility and reliability of our interpretation process. Finally, we measured inter-annotator agreement (IAA) using Cohen's Kappa. We achieved an IAA of 0.97, indicating a high level of agreement between our annotations.

3.4.2 POS tagging

I used various POS tagging models presented in Section 3.3 on the tokenized sample senteces as well as the treebank data. Different methods were employed for each tagger, generating separate output txt files. Python scripts handled LatinCy and CLTK, command line commands managed TreeTagger and RNNTagger, and R scripts were used for UDPipe and RDRPOSTagger. The scripts used for this process are accessible in my GitHub repository[17]. I standardized token placement across all files to ensure consistency, enabling accurate POS tag analysis. In cases where taggers failed to assign a tag or presented formatting issues, hindering tag assignments, I replaced the tags in these areas with "X." Accuracy served as the primary metric for evaluation after correcting the output. I also experimented with both GPT-3.5-Turbo and GPT-4 models using the GPT API. Initially, providing only a sentence from the Bullinger corpus in the prompt led to tokenization issues. To fix this, I adjusted the input prompt, including the example sentence and a tokenized version of the sentence that matches the tokenization in the GS to ensure proper tokenization matching. The prompt in Figure 3 included system role definition and output formatting instructions. Additionally, I tested a token-only approach for the Bullinger corpus, submitting only tokens without a reference sentence in the prompt.

To use GPT models for POS tagging on the treebank data, I devised an alternative approach due to missing sentence boundaries in some datasets. Instead of providing full sentences, I used sets of 65 tokens in the prompts. This approach substituted the need for punctuation to delineate sentences. However, the output required extensive manual adjustments due to formatting errors. The method's time-consuming nature was particularly challenging, especially with larger treebank datasets. As a result, I could only conduct a single run for each model and dataset, limiting the assessment to a singular evaluation without averaging accuracy across multiple runs.

[17] `https://github.com/Eljuanina/BA`

```
completion = openai.ChatCompletion.create(
    model = model,
    messages=[
        {"role":"system","content": """You are a Latin linguist and part-of-speech tagging expert. You are
        using UPOS (universal part of speech tags). UPOS tags are ADJ,ADP,ADV,AUX,CCONJ,DET,INTJ,NOUN,NUM,
        PART,PRON,PROPN,PUNCT,SCONJ,SYM,VERB and X. X stands for 'other'."""},
        {"role":"user",
        "content":f"""Return the UPOS tag for the tokens of the sentence: {sentence} The sentence should be
        tokenized like that: {tokens}. Return the tags in the format TOKEN \t Tag. Every Token-Tag pair
        should be on a new line in the output file, so add a newline character after the tags. Only output
        the token and the tag (no explanations, no translations, no additional text)."""}
        ]
    )
```

Figure 3: Prompt employed for POS tagging using the GPT API

3.4.3 Fine-tuning of GPT-3.5-Turbo

I fine-tuned the GPT-3.5-Turbo model using training and test data gathered from the treebanks specified in Section 3.2.2. To maintain a 20/80 split between test and training data, I adjusted dataset sizes accordingly. When the predefined test sets were absent, such as for PROIEL and Perseus, or when the test set exceeded 20%, such as for UDante, I randomly selected sample sentences to maintain a 20% ratio. However, for treebanks such as ITTB and LLCT, with test sets of only 10%, I utilized the existing predefined test and training sets. This resulted in an overall adjustment, resulting in approximately 16.2% of the data used for the test set and 83.8% used for training[18].

I divided the training data into subsets comprising 50, 100, 200, 500, 1,000, 2,000, 5,000, and 10,000 sentences. According to OpenAI, a minimum of 10 examples is required for effective fine-tuning, while substantial improvements typically demand 50 to 100 examples[19]. I maintained consistent proportions of each treebank in the test set across subsets by precisely calculating the number of sample sentences needed for each treebank in every training subset.

Fine-tuning necessitated transforming the examples into a specific format specified by OpenAI. An illustration of this transformed data is available in Appendix D.1. Subsequently, I uploaded these files to OpenAI and initiated the fine-tuning process, following the steps outlined in a Jupyter notebook accessible on my GitHub. Note that I did not write this code myself, as the code was already accessible online.

[18]More details about the exact splits can be found in my Jupyter notebook named "all_training_data.ipynb" on GitHub.

[19]See `https://platform.openai.com/docs/guides/fine-tuning/when-to-use-fine-tuning`

4 Results

In this section, I discuss the outcomes obtained from applying different POS taggers on the Bullinger corpus as well as on the treebank test corpora. This analysis is crucial for evaluating how well pretrained taggers work on 16th century Latin texts. I am primarily looking at their accuracy and adaptability across different types of language data. By comparing the results and evaluating how effective the taggers are, this research aims to contribute to the field of NLP, especially concerning historical Latin analysis and the use of LLMs for POS tagging.

4.1 Output of GPT models

The evaluation of the GPT models revealed several discrepancies between the anticipated and generated outputs. These deviations encompassed repetition and omission of tokens or entire segments, and alterations in token-tag separators, such as the use of a tilde instead of the designated tab. Despite the requests for UPOS tags in the prompts, instances of erroneous tags, including the presence of lowercase variations, were detected. Despite these inconsistencies, I accepted the scenario of lowercase tags and compared the tags in their lowercase version to assess the ability of the GPT models to assign correct tags. For example, in the output "et UNK lato UNK taget UNK," erroneous tags were assigned, highlighting the model's shortcomings. Additionally, there were occasions where the models unexpectedly inserted translations or incorporated text from different languages, such as French prompts or Cyrillic-script text. These anomalies are showcased in the provided examples in Appendix D.2 or in the Jupyter notebook named "clean_output.ipynb" on GitHub. Furthermore, peculiarities were observed in the outcomes, notably in an instance where the generated output displayed unexpected text:

```
Vuido   PROPN
Nopqrstuvwxyz
gratia   NOUN
```

The occurrence of missing or incorrect data varied among the models. Generally, models trained with larger datasets of 5,000 or 10,000 examples displayed more issues compared to those trained with fewer examples. Nevertheless, a consistency in encountering errors in specific passages across different models was noted, indicating persistent issues in certain areas despite variations in the size of the training sets.

4.2 POS tagging on Bullinger corpus

The creation of a reference GS involved collaboration to ensure accuracy. This validation underscored the similarity between my GS and that of my former Latin teacher. Given the near-identical outcomes, I solely evaluated the taggers against my self-generated GS. The taggers' performance, assessed by accuracy, is detailed in Table 5. LatinCy demonstrated the highest accuracy from the pretrained models at 79.8%, slightly surpassing CLTK by 2.7%. However, RDRPOSTagger displayed the lowest accuracy at 64.5%, indicating limitations in processing Bullinger's 16th century data (research question 1).

LatinCy	CLTK	UDPipe	RDRPOSTagger	TreeTagger
79.8	77.1	72.8	64.5	74.3

Table 5: Accuracy of the taggers on the Bullinger corpus in %

In Section 3.4.2, I delineated two evaluation approaches for the GPT models. The results for both the token-only and sentence-included approaches are outlined in Table 6. Here, the fine-tuned models are abbreviated using the prefix "train" followed by the number of training examples. For instance, the GPT-3.5-Turbo model fine-tuned with 50 training examples is labeled as "train50."

prompt	GPT-3.5-Turbo	train50	train100	train200	train500
tokens	62.2	70.3	69.4	68.2	58.8
sentence	80.2	84.8	85.5	84.0	77.2
prompt	train1000	train2000	train5000	train10000	GPT-4
tokens	65.8	78.2	71.9	74.4	74.3
sentence	82.5	78.3	76.6	76.4	83.5

Table 6: Accuracy of the GPT models on the Bullinger corpus in %

The observed accuracies significantly varied between the two tagging methods. Uti-

lizing complete sentences in the prompt consistently yielded higher accuracy compared to the token-only method across all models. The token-only method averaged 69.35% accuracy, while the sentence-inclusive method achieved 80.9%, indicating an over 11% difference.
In the token-only method, the model trained on 2,000 sentences exhibited the highest accuracy at 78.2%. This model demonstrated 1.6% lower accuracy compared to LatinCy but surpassed the remaining models. In contrast, the train500 model showed the weakest performance, achieving only 58.8% accuracy, which was 5.7% lower than RDRPOSTagger—the least accurate tagger from Table 5.
However, the sentence-inclusive method consistently yielded higher accuracy than TreeTagger. The train100 model attained the highest accuracy at 85.5%, surpassing even the non-fine-tuned GPT-4 model. Notably, the train500 model showcased the most significant disparity between the two approaches, with an 18.4% difference, while the train2000 model showed a marginal difference of 0.1%.
The tagging process was time-consuming, particularly for GPT-4, hence each model was tested only once. Pretrained models were faster, yet the GPT models sometimes took several hours, contingent upon the global GPT demand. Testing the models during off-peak hours expedited processing times, whereas usage during evening hours substantially prolonged them.

4.3 POS Tagging on the test data

Among the tested taggers and pretrained models, LatinCy emerged with the highest average accuracy, achieving 83.22% on the treebank test data (see Table 7 in Appendix B). Figure 4 depicts the average accuracy of the taggers, indicating effective performance of the LLMs as POS taggers, addressing research question 2.

Among the evaluated models, the fine-tuned train1000 model showcased the most effective performance, boasting an average accuracy of 88.99%. Conversely, RDRPOSTagger exhibited the lowest performance, recording an average accuracy of 72.36%, marking a 16.63% difference between the highest and lowest performers.
Regarding performance across various test sets, the taggers displayed superior accuracy on the ITTB test set, averaging 84.95%. Conversely, the taggers achieved the lowest average accuracy on the UDante test set at 78.4%. On the LLCT test set, the taggers demonstrated an accuracy of 84.09%, while the taggers achieved on the Perseus and PROIEL datasets accuracies of 79.63% and 84.07%, respectively. The collective average accuracy across taggers and test sets stood at 81.26%.

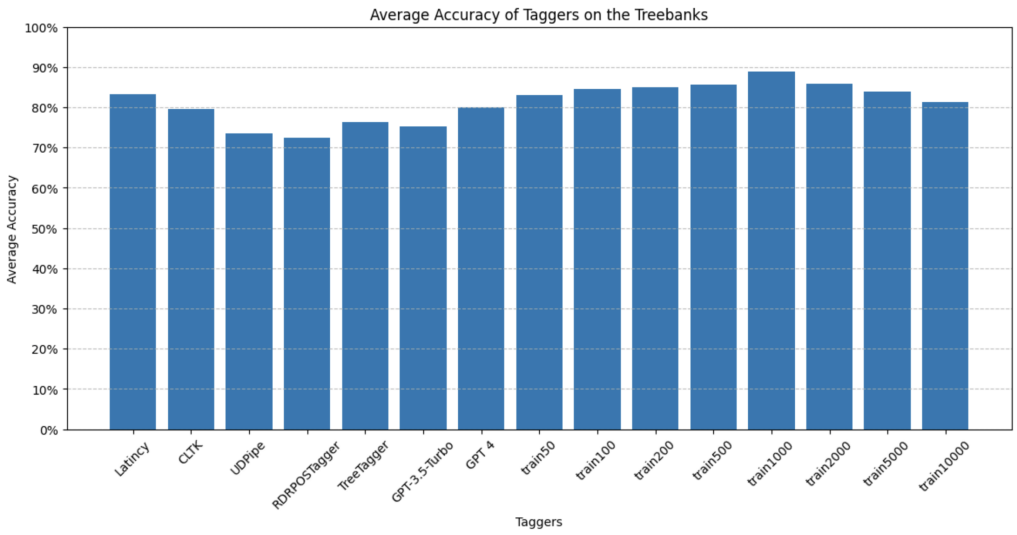
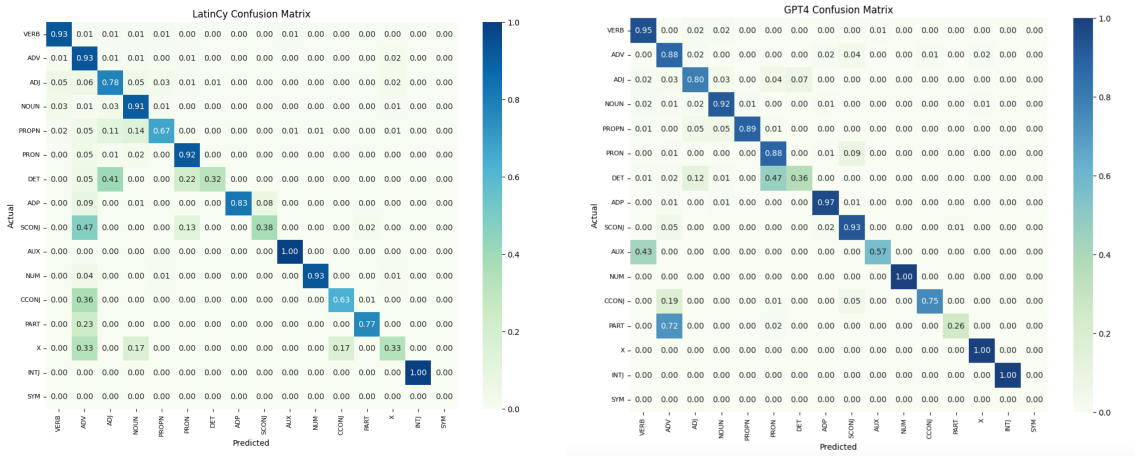


Figure 4: Average accuracy on the treebank test data

4.4 Tag distribution

The models' performance differed depending on the POS tag, showing both accurate and inaccurate assignments. In Figure 5, the confusion matrices for LatinCy and GPT-4 on the Bullinger corpus are displayed. In the Appendix C, the confusion matrices for all of the taggers evaluated in this thesis, along with their respective tags for the Bullinger corpus, are presented in Figures 12 and 13. These matrices highlight the tags assigned by the taggers along the x-axis, representing their predictions, and the tags from the GS along the y-axis, offering insights into their performance.

As can be seen in Figure 5a, LatinCy showcases a nearly diagonal line, suggesting overall accurate predictions. However, it correctly predicts the subordinating conjunctions (SCONJ) tag only 38% of the time, incorrectly predicting ADV 47% of the time and PRON 13% of the time. Conversely, GPT-4 (see Figure 5b) displays a more uniform and darker diagonal line, implying higher accuracy in tag assignments. These matrices reveal variations in the usage of UPOS tags among different taggers. For instance, some tags like SYM are absent in the GS and are not assigned by most taggers, resulting in an empty row. In Figure 12, RDRPOSTagger also omits the use of tags like PART, X, and INTJ. The presence of misalignment between predictions in RDRPOSTagger's results is evident, particularly in the upper left quadrant. This suggests varied and less consistent predictions, a characteristic that becomes apparent due to the scattering of colored fields.



(a) Confusion matrix for LatinCy (b) Confusion matrix for GPT-4

Figure 5: Confusion matrix of two taggers on the Bullinger sample

A striking observation lies in the inverse relationship between the model performance and the quantity of training data, as depicted in Figure 13. For instance, the assignment of the AUX tag remains consistently imperfect across various models. The train50 model assigned it as VERB (34%), AUX (63%), and X (3%). In contrast, train5000, with 100 times more training examples, eliminated the assignment of X entirely, assigning VERB in 94% of cases and AUX in only 6% of cases. Only train200 utilized the INTJ tag, albeit with a 50% correct assignment rate.

In Figures 7 and 8, illustrating the tag distribution of pretrained models across the Bullinger corpus and treebank data, NOUN and VERB emerge as the most prevalent tags, aligning well with the GS for the Bullinger corpus. Notably, UDPipe sparsely assigns the SCONJ tag, while PRON occurrences by UDPipe and LatinCy are relatively fewer compared to other pretrained models or the GS. Conversely, ADV is frequently assigned by these two taggers.

Turning to Figures 9 and 10, showcasing tag distribution among GPT models, a balanced distribution is observed across different models. The train2000 model has many X tags in the treebank data, but it also had relatively many Xs in the Bullinger corpus. However, a peculiar observation surfaces concerning the PUNCT tag. Smaller GPT models exhibit occurrences of this tag in the Bullinger corpus, although theoretically, there should be none due to the absence of punctuation.

5 Discussion

The study analyzed the performance of different models in POS tagging tasks, finding that LatinCy had the highest accuracy of pretrained POS taggers, while train1000 outperformed as the best tagger on treebank test data. The study also revealed significant differences in the effectiveness of different prompting strategies on Bullinger data, emphasizing the importance of context in POS tagging. The contrasting successes of various models highlight the potential effectiveness of language models like GPT models in POS tagging tasks. The study also examined the impact of fine-tuning on GPT models' accuracy in POS tagging tasks.

5.1 Accuracy on Bullinger Corpus

This section compares taggers' performance to the GS to assess how accurate they are tagging the Bullinger corpus. Among the pretrained models, LatinCy was particularly accurate, though the baseline GPT models outperformed it. Remarkably, some fine-tuned GPT models even performed better than the baseline models.

5.1.1 Comparison of accuracy of pretrained taggers'

The assessment of pretrained POS taggers revealed distinct discrepancies in accuracy when confronted with the 16th century Bullinger corpus. LatinCy showcased the highest accuracy at 79.8% against the GS, a significant difference from its 97.41% performance on test data as stated in Burns [2023]. Also the other pretrained models contrasting deviations in accuracy from their performances on other datasets.
The divergence in accuracy among the taggers when applied to the Bullinger corpus hints at potential challenges stemming from unique data characteristics or variations in tag assignments. While the discrepancies observed could partially be attributed to tagging errors, the high IAA suggests broader issues beyond individual tagging inconsistencies.
These findings emphasize the substantial hurdles pretrained taggers face when han-

dling historical corpora like Bullinger's 16th century data. Notably, the EvaLatin shared task evaluated tagger performance across different eras, signifying the inherent difficulty in applying these models to diverse historical periods. The notable discrepancies observed among taggers when compared to both the GS and their performances on other datasets underscore the complexity and distinct nature of historical text analysis.

The evaluation of pretrained POS taggers on the 16th century Bullinger corpus revealed significant discrepancies in word usage or tagging standards among the Bullinger corpus, the gold standard, and the training data. This divergence appears to have notably contributed to reduced performance across all taggers when applied to the Bullinger dataset. In response to the first research question, "How accurately do pretrained POS taggers perform on the 16th century Bullinger data?," these findings underscore the considerable hurdles these taggers face when dealing with this specific historical corpus and indicate that the pretrained taggers only show a moderate performance on 16th century data.

5.1.2 Comparison of accuracy of GPT models

Within the tokens-only approach, the train2000 model showcased superior accuracy at 78.2%, outperforming all pretrained models on the 16th century GS dataset except for LatinCy. Notably, models trained on more examples exhibited decreased accuracy, possibly indicating potential overfitting issues.

The sentence-included approach significantly improved results compared to the tokens-only method, with most models surpassing pretrained counterparts or achieving high performance levels. Enhanced accuracy was attributed to the model's ability to contextualize words within sentences, facilitating more precise tagging based on varied word usages.

The most notable performance leap occurred with the train500 model, which initially had the poorest accuracy in the tokens-only approach. The unexplained difference between the performance of train200 (84.0%) and train500 (77.2%) (refer to Table 6) persists, despite the latter including all the data from train200 along with additional examples. It is plausible that conflicting examples in train500 caused confusion that could be resolved by providing more examples, as implemented in train1000. Surprisingly, the train100 model with limited training examples achieved the highest accuracy at 85.5%, though reproducibility remains uncertain due to limited testing epochs.

The study's comparison with Chu[1] highlights performance similarities and differences between GPT models' POS tagging outcomes. However, direct comparisons were hindered by differences in experimental methodologies, such as the absence of fine-tuning in Chu's study. Unexpected behaviors, like punctuation dropping and the usage of incorrect tags, were encountered, echoing findings from Chu's work.
The analysis underscores the pivotal role of prompting in LLMs and its influence on POS tagging. Solely providing tokens resulted in significantly lower accuracy, underscoring the limitations of disambiguating POS tags without contextual cues. This analysis aligns with the notion that LLMs excel in NLP tasks and highlights the substantial impact of prompting methodologies on LLM performance, surpassing the influence of training data quantity.

5.1.3 Comparison of accuracy on the treebank data

All taggers encountered difficulties with the Bullinger corpus, resulting in none achieving outstanding results. Even the best model did not achieve comparable accuracy with taggers such as Latin BERT, which is referred to in Section 2.3. This raises uncertainty regarding whether the issue stems from my tagging choices or if the Bullinger corpus dataset was an unfortunate selection. Although the dataset was randomly selected, the limited sample size of 200 samples may have contributed to the poor performance, as it may include certain edge cases. To gain a clearer perspective on tagger performance, I tested the taggers on test data sourced from the treebanks they were originally trained on, for which one would expect better performance than with the Bullinger corpus.
Although minimal deviation in average accuracy was observed across different models (Figure 4), the GPT models showcased superior performance. Specifically, the train1000 model outperformed LatinCy by a relative improvement of over 5% (Table 7), excelling particularly on the ITTB and LLCT data. Upon reviewing the outcomes of the experiments, it can be asserted that language models such as GPT can effectively serve as POS taggers, thus providing an affirmative response to the second research question.The illustration of tagger accuracy on treebanks through boxplots (Figure 6) reveals fluctuations among datasets. While UDPipe demonstrated minimal fluctuation but consistent mediocre performance, models like train500 and LatinCy showed significant variability among datasets, performing well on certain data but lower on others. The train1000 model showcased the highest median accuracy, denoted by orange lines in Figure 6, with a more focused range of scores, suggesting

[1]See "GPT-4 is a very good Hongkongese POS Tagger" `https://medium.com/@kyubi_fox/gpt-4-is-a-very-good-hongkongese-pos-tagger-feeae1b44868`

better balance between higher accuracy and reduced variability. Conversely, the train2000 model demonstrated consistency across sets with good average accuracy. Addressing the fourth research question, "Is there a notable discrepancy in the performance of POS taggers when applied to Classical Latin data?" points to a considerable difference in tagger performance between treebank data, largely representing Classical Latin, and the Bullinger 16th century dataset. However, limitations in sample size and potential edge cases in the dataset may have influenced these findings.

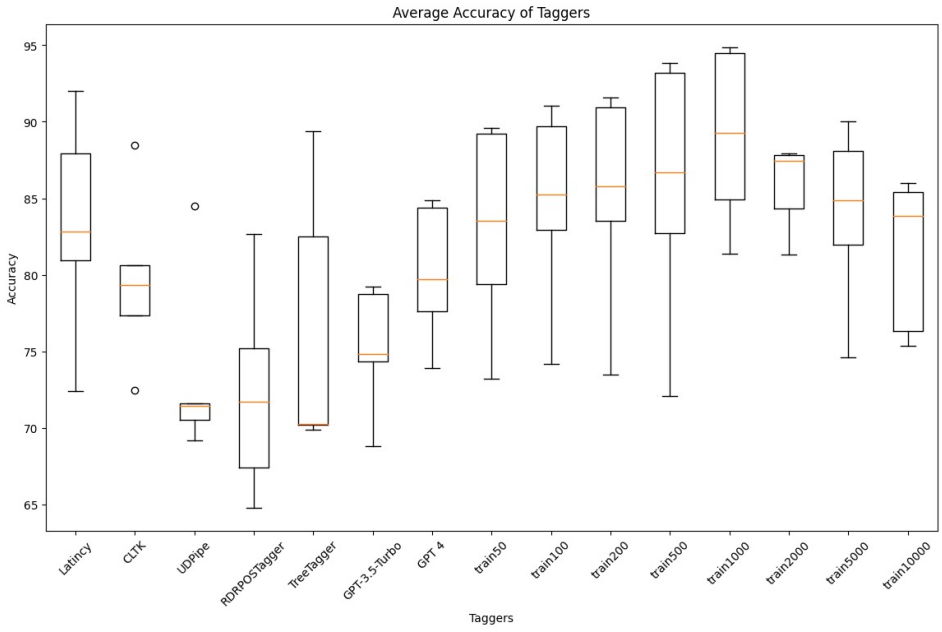


Figure 6: Boxplots illustrating the average accuracy on the treebank test data

5.1.4 Fine-tuning of GPT

The fine-tuned models showed significant improvements over their base counterparts. For instance, the train1000 variant displayed a remarkable increase of 13.81 percentage points in average accuracy compared to the foundational GPT-3.5-Turbo model, even surpassing the more robust GPT-4 by 8.89% in average accuracy. However, while GPT-4 demonstrated higher accuracy than GPT3.5-Turbo, its operational speed was comparatively lower than that of the fine-tuned models. Notably, the expenses associated with utilizing GPT-4 during prompting were higher, whereas fine-tuning incurred costs during the training phase. The decision to engage in fine-tuning depends on individual assessment, weighing whether the resultant accuracy increase justifies the associated costs. For applications involving extensive data usage, fine-tuning appears more justifiable. Conversely, for singular or infrequent

model usage, the necessity of fine-tuning might be debatable, especially considering that comparable accuracy increments could potentially be achieved through alternative prompting strategies. This emphasizes the pivotal role of precise prompts in maximizing the efficacy of fine-tuned models, as varying prompting strategies could also yield accuracy enhancements besides additional training examples. My assessment of the models' cost and speed is based solely on their current status during this thesis, acknowledging the potential for future versions to alter these aspects.
Regarding performance on known datasets versus the unseen Bullinger corpus, the fine-tuned models, along with GPT-3.5-Turbo and GPT-4, showed good results. This supports the rationale for using LLMs in POS tagging and aligns with prior insights presented by Chu, suggesting a positive response to my second and third research questions. However, exploring alternative prompting strategies, such as using complete sentences instead of multiple tokens per prompt, or making adjustments in parameters like temperature, remains an area requiring further investigation.
Regarding the impact of fine-tuning on GPT model accuracy in POS tagging tasks, it was observed that while fine-tuning enhances performance, the extent of improvement might not be as substantial as anticipated. Additionally, the choice of prompting strategy appeared to have a greater influence on GPT model performance than fine-tuning itself. However, limitations in experimentation, including single-epoch testing and formatting complexities, restricted a comprehensive assessment of fine-tuning's true potential.
The study's limitations, such as the restriction to one epoch of assigning POS tags and the inability to retag incorrectly formatted passages, constrained the exploration of fine-tuning's full impact.

5.2 Comparison of the tag distributions

Comparing tag distributions across Figures 8, 10, and 11 highlights potential factors influencing model accuracy. Notably, the train2000 model's frequent use of the X tag correlates well with its prevalence in the Perseus treebank, suggesting a strong correlation and might be a factor for its good performance. Surprisingly, the train1000 model's high accuracy on ITTB and LLCT datasets, despite the absence of distinct peculiarities in tag distribution patterns, underscores the intricate relationship between distribution characteristics and model accuracy on specific data. LatinCy's alignment with tagging distributions, frequently assigning PROPN and PUNCT tags in line with their prevalence in the LLCT dataset, may contribute to its high accuracy. However, discrepancies, like LatinCy's frequent tagging of ADV

and ADJ not prevalent in the LLCT data, exist. For a comprehensive overview, refer to the tag distribution figures in Appendix C and Table 7 in Appendix B for tagger accuracies on treebanks. These findings suggest that a model's performance is shaped by the dataset's characteristics, and mismatches between a tagger's assignments and the dataset's tag usage may impact accuracy.

6 Conclusion

6.1 Summary

This research aimed to evaluate pretrained POS taggers as well as LLMs, exemplified by GPT, on 16th century Latin texts, particularly Heinrich Bullinger's correspondence. The central questions for this research were as follows:

1. How accurately do pretrained POS taggers perform on the 16th century Bullinger data?
2. Can language models such as GPT models be effectively used as POS taggers?
3. Can fine-tuning significantly increase the accuracy of GPT models in POS tagging tasks?
4. Is there a notable discrepancy in the performance of POS taggers when applied to Classical Latin data?

The evaluation of pretrained POS taggers on 16th century Latin texts unveiled a moderate level of competence among the tested models, with LatinCy demonstrating the highest performance at 79.8% accuracy.
Exploring the potential of LLMs such as GPT models revealed promising results. GPT exhibited competitive performance when tagging both the 16th century data and the treebank data, primarily from Classical Latin. The results highlighted their effectiveness, notably influenced by prompt engineering. Contextual prompts significantly enhanced tagging accuracy, showcasing the pivotal role of context in disambiguating POS tagging. Additionally, fine-tuning GPT models using Latin treebank data notably improved performance, with the train1000 model achieving an average accuracy of 88.99%. This emphasized the efficacy of fine-tuning in bolstering tagging accuracy compared to base GPT models and highlighted the critical role of prompt selection, significantly influencing output quality. However, it is essential to note that the accuracy assessment could not determine the final output as the tagging was conducted only once, and manual corrections were applied to unclear model outputs. This manual intervention potentially diminished the accuracy

recorded.
However, despite improved performance on Classical Latin data, the effectiveness of taggers diminished when applied to 16th century Latin texts. This opens up possibilities for further refinement in the context of 16th-century Latin POS tagging, considering the time-consuming nature and cost implications associated with GPT models. Nonetheless, for Classical Latin, fine-tuning models appear to be a viable strategy.
In conclusion, this study illuminated both the challenges and potential inherent in employing pretrained taggers and LLMs for POS tagging historical Latin texts. Addressing the main research question, it provided insights into the proficiency of taggers and LLMs in handling historical Latin texts. Reflecting on the research process, the study showcased the nuanced impact of prompt selection and fine-tuning on tagging accuracy, revealing the critical nature of these factors. Additionally, it raised pertinent questions regarding the adaptability of language models to different historical Latin eras, suggesting potential avenues for future investigations.

6.2 Future Work

Expanding research in this area holds significant promise for enhancing our understanding and application of language models for Latin historical texts. Challenges persist in creating high-quality parsers adaptable across different Latin treebanks (see Passarotti and Dell'Orletta [2010] and McGillivray and Passarotti [2009]) due to insufficient annotations in Latin datasets. The varied compositions of corpora and shifts in language within Christian and Medieval Latin have a notable influence on the effectiveness of parsers across different versions and periods of Latin (see Dinkova-Bruun [2011]). Creating annotated datasets tailored to specific historical periods, such as 16th century Latin or spanning various eras, could mitigate these difficulties. These datasets need to encompass the linguistic variations inherent in Christian and Medieval Latin to improve parser performance across diverse linguistic versions.
The correlation between tag distribution and model accuracy poses an intriguing area for further exploration. Analyzing this relationship across various treebanks and models could elucidate deeper insights into the performance dynamics of language models on historical texts. This avenue holds substantial promise in enhancing our comprehension and utilization of language models, particularly for Latin historical texts.
Although current limitations do not allow fine-tuning of GPT-4 models, the future

availability of this capability could improve such experiments, as GPT-4 has superior performance to GPT-3.5-Turbo. Clear variations in accuracy, as observed in Table 6, suggest the importance of exploring different approaches in prompt design and fine-tuning models tailored to 16th century data. Exploring varied prompt designs and implementing targeted fine-tuning strategies holds promising potential for enhancing model performance. Furthermore, conducting multiple test runs using identical models and data could yield more comprehensive insights into model accuracy, despite the potential challenge of increased processing times.

The pursuit of improved prompts, thorough exploration of 16th century data, and the prospect of training GPT models on multiple low-resource languages for POS tagging not only pave the way for future research avenues but also mark crucial strides toward a more refined and resilient application of LLMs in analyzing historical Latin texts.

Glossary

accuracy A fundamental metric used to assess the **performance** of automatic **annotation tools** like parsers or POS taggers. It calculates the proportion of **tokens** accurately tagged among the total number of tokens.

clitic A piece of language that has word syntactic properties but is phonologically and lexically connected to another word. The most common Latin enclitics are *-que* and *-ne*. When POS tagging, they are often treated separately from the words they are attached to.

confusion matrix A **visual representation** in grid form that demonstrates the classification model's performance. It illustrates the **comparison** between predicted and actual classes within a dataset.

gold standard An either manually or automatically **annotated dataset**. Gold standards serve as **benchmark** datasets used to assess and test the performance of **automatic NLP tools**.

GPT API Enables developers to use natural language processing features like text generation, summarization, question answering, and more within their applications or systems by allowing them to **integrate** and access the capabilities of **GPT** models.

hallucinations A **form of model failure**, where the models produce **content** that is **implausible, unrelated, or factually incorrect** in relation to the given context. These mistakes arise from the models' propensity to produce text based on patterns discovered from training data, which can occasionally produce absurd or deceptive results.

IAA The IAA is a metric that measures **consensus among annotators**, ensuring consistency and quality of annotations. It is crucial for **evaluating datasets with multiple annotators**, facilitating quality control and reliable NLP model training.

References

J. Ács, Á. Kádár, and A. Kornai. Subword pooling makes a difference. pages 2284–2295, 2021. URL https://aclanthology.org/2021.eacl-main.194.

A. Babeu. The perseus catalog: of frbr, finding aids, linked data, and open greek and latin. *Digital Classical Philology*, pages 53–72, 2019.

D. Bamman and P. J. Burns. Latin BERT: A contextual language model for classical philology, 2020. URL https://arxiv.org/abs/2009.10053.

D. Bamman and G. R. Crane. The design and use of a latin dependency treebank. In *Proceedings of The Third Workshop on Treebanks and Linguistic Theories (TLT 2006)*, 2006. URL https://api.semanticscholar.org/CorpusID:16351887.

C. E. Bennett. *The Latin Language: A Historical Outline of Its Sounds, Inflections, and Syntax*. Allyn, 1907.

L. Borin. Something borrowed, something blue: Rule-based combination of pos taggers. In *LREC*, 2000.

E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. 21(4):543–565, 1995. URL https://aclanthology.org/J95-4004.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Bullinger Digital. Bullinger digital, 2023. URL https://www.bullinger-digital.ch.

P. J. Burns. Building a text analysis pipeline for classical languages, 2019. URL https://www.researchgate.net/publication/335036188_Building_a_Text_Analysis_Pipeline_for_Classical_Languages.

P. J. Burns. Latincy: Synthetic trained pipelines for latin NLP, 2023. URL https://arxiv.org/pdf/2305.04365.pdf.

F. Cecchini, R. Sprugnoli, G. Moretti, M. Passarotti, et al. Udante: First steps towards the universal dependencies treebank of dante's latin works. In *Proceedings of the Seventh Italian Conference on Computational Linguistics*, pages 99–105. Accademia University Press, 2020a.

F. M. Cecchini, M. Passarotti, P. Marongiu, and D. Zeman. Challenges in converting the index Thomisticus treebank into Universal Dependencies. In M.-C. de Marneffe, T. Lynn, and S. Schuster, editors, *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 27–36, 2018. URL https://aclanthology.org/W18-6004.

F. M. Cecchini et al. UDante: First steps towards the universal dependencies treebank of dante's latin works. In *Proceedings of the Seventh Italian Conference on Computational Linguistics CLiC-it 2020*. Accademia University Press, 2020b. URL http://books.openedition.org/aaccademia/8653.

G. G. Celano. The dependency treebanks for ancient greek and latin, 2019. URL https://www.degruyter.com/document/doi/10.1515/9783110599572-016/html.

A. Chiche and B. Yitagesu. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data*, 9(1):1–25, 2022.

K. W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pages 136–143, 1988. URL https://aclanthology.org/A88-1019.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

G. Dinkova-Bruun. Medieval latin. In J. Clackson, editor, *A Companion to the Latin Language*, pages 284–302. Blackwell Publishing, 2011. URL https://onlinelibrary.wiley.com/doi/book/10.1002/9781444343397.

H. Eckhoff, M. Majer, E. Welo, and D. Haug. Breaking down and putting back together: Analysis and synthesis of new testament greek. *Journal of Greek*

Linguistics, 9(1):56–92, 2009. URL https://www.researchgate.net/publication/233601585_Breaking_down_and_putting_back_together_analysis_and_synthesis_of_New_Testament_Greek.

H. Eckhoff, K. Bech, G. Bouma, K. Eide, D. Haug, O. E. Haugen, and M. Jøhndal. The proiel treebank family: a standard for early attestations of indo-european languages. *Language resources and evaluation*, 52:29–65, 2018.

D. Embick. Features, syntax, and categories in the latin perfect. *Linguistic Inquiry*, 31(2):185–230, 2000.

A. Erdmann, D. J. Wrisley, B. Allen, C. Brown, S. Cohen-Bodénès, M. Elsner, Y. Feng, B. Joseph, B. Joyeux-Prunel, and M.-C. de Marneffe. Practical, efficient, and customizable active learning for named entity recognition in the digital humanities. In J. Burstein, C. Doran, and T. Solorio, editors, *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 2223–2234, 2019. URL https://aclanthology.org/N19-1231.

M. Fantoli, M. Passarotti, F. Mambrini, G. Moretti, and P. Ruffolo. Linking the LASLA corpus in the lila knowledge base of interoperable linguistic resources for latin. In *Proceedings of the 8th Workshop on Linked Data in Linguistics Within the 13th Language Resources and Evaluation Conference*, pages 26–34, 2022. URL https://doi.org/10.5281/zenodo.6659314.

L. Fischer, P. Scheurer, R. Schwitter, and M. Volk. Machine translation of 16th century letters from latin to german. In *Fischer, Lukas; Scheurer, Patricia; Schwitter, Raphael; Volk, Martin (2022). Machine Translation of 16th Century Letters from Latin to German. In: Second Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA 2022), Marseille, 25 Juni 2022. LREC, 43-50.*, page 43–50. LREC, 2022. URL http://www.lrec-conf.org/proceedings/lrec2022/workshops/LT4HALA/pdf/2022.lt4hala2022-1.7.pdf.

M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*, 2023.

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

K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2545–2568, 2021. URL https://aclanthology.org/2021.naacl-main.201.

K. P. Johnson, P. J. Burns, J. Stewart, T. Cook, C. Besnier, and W. J. B. 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. URL https://aclanthology.org/2021.acl-demo.3.

D. Jurafsky and J. H. Martin. Jurafsky martin chapter 8, 2019. URL https://web.stanford.edu/~jurafsky/slp3/old_oct19/8.pdf.

A. Kempe. A probabilistic tagger and an analysis of tagging errors. *Kempe*, 1993. URL http://a.kempe.free.fr/reports/kempe93a/kempe93a.pdf.

T. Korkiakangas. Late latin charter treebank: Contents and annotation. *Corpora*, 16:191–203, 2021a. URL https://www.academia.edu/76636757/Late_Latin_Charter_Treebank_contents_and_annotation.

T. Korkiakangas. Late latin charter treebank: contents and annotation. *Corpora*, 16(2):191–203, 2021b.

J. Leonhardt. *Latin: Story of a World Language*. Harvard, 2009.

Y. Liu, H. He, T. Han, X. Zhang, M. Liu, J. Tian, Y. Zhang, J. Wang, X. Gao, T. Zhong, et al. Understanding llms: A comprehensive overview from training to inference. *arXiv preprint arXiv:2401.02038*, 2024.

B. McGillivray. Methods in latin computational linguistics, 2015. URL https://www.degruyter.com/document/doi/10.1515/joll-2015-0007/html.

B. McGillivray and M. Passarotti. The development of the index thomisticus treebank valency lexicon. In *Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education (LaTeCH-SHELT&R 2009)*, 2009. URL https://aclanthology.org/W09-0306.

S. Nehrdich and O. Hellwig. Accurate dependency parsing and tagging of Latin. In R. Sprugnoli and M. Passarotti, editors, *Proceedings of the Second Workshop on Language Technologies for Historical and Ancient Languages*, pages 20–25.

European Language Resources Association, 2022. URL https://aclanthology.org/2022.lt4hala-1.3.

D. Q. Nguyen, D. Q. Nguyen, D. D. Pham, and S. B. Pham. RDRPOSTagger: A ripple down rules-based part-of-speech tagger. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, page 17–20, 2014. URL https://aclanthology.org/E14-2005.

OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/pdf/2303.08774.pdf.

M. Passarotti and F. Dell'Orletta. Improvements in parsing the index thomisticus treebank. revision, combination and a feature model for medieval latin. 2010. URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/178_Paper.pdf.

M. C. Passarotti. The project of the index thomisticus treebank. In M. Berti, editor, *Digital Classical Philology. Ancient Greek and Latin in the Digital Revolution*, pages 299–319. De Gruyter, 2019. URL http://hdl.handle.net/10807/141133.

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 *International Conference on Language, Data, and Knowledge*, 2019. URL https://api.semanticscholar.org/CorpusID:196810553.

M. Pellegrini, E. Litta, M. Passarotti, R. Sprugnoli, F. Mambrini, and G. Moretti. Lila linking latin tutorial. *LiLa*, 2023. URL https://ceur-ws.org/Vol-3064/LDK2021_Tutorial_Paper.pdf.

S. Petrov, D. Das, and R. McDonald. A universal part-of-speech tagset, 2011.

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

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*, 2020. URL https://nlp.stanford.edu/pubs/qi2020stanza.pdf.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

D. Richards. Two decades of ripple down rules research. *Knowledge Eng. Review*, 24:159–184, 2009. URL https://www.researchgate.net/publication/220254277_Two_decades_of_Ripple_Down_Rules_research.

H. Schmid. *Probabilistic Part-of-Speech Tagging Using Decision Trees*. Institut für maschinelle Sprachverarbeitung, 1994. URL https://www.cis.lmu.de/~schmid/tools/TreeTagger/data/tree-tagger1.pdf.

H. Schmid. Deep learning-based morphological taggers and lemmatizers for annotating historical texts. In *Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, DATeCH2019, page 133–137. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3322905.3322915.

R. Sprungoli and M. Passarotti. *1st Workshop on Language Technologies for Historical and Ancient Languages, (LT4HALA 2020) Proceedings*. European Language Resources Association (ELRA), 2020. URL https://lrec2020.lrec-conf.org/media/proceedings/Workshops/Books/LT4HALAbook.pdf.

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 R. Sprugnoli and M. Passarotti, editors, *Proceedings of LT4HALA 2020 - 1st Workshop on Language Technologies for Historical and Ancient Languages*, pages 130–135. European Language Resources Association (ELRA), 2020. URL https://aclanthology.org/2020.lt4hala-1.21.

M. Straka. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In D. Zeman and J. Hajič, editors, *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, 2018. URL https://aclanthology.org/K18-2020.

M. Straka and J. Straková. UDPipe at evalatin 2020: Contextualized embeddings and treebank embeddings, 2020. URL https://arxiv.org/pdf/2006.03687.pdf.

P. B. Ströbel, T. Hodel, W. Boente, and M. Volk. The adaptability of a transformer-based ocr model for historical documents. In *Document Analysis*

and Recognition – ICDAR 2023 Workshops, Lecture Notes in Computer Science, pages 34–48. Springer, 2023. URL https://doi.org/10.5167/uzh-235624.

P. B. Ströbel. pstroe/cc100-latin · datasets at hugging face, 2023. URL https://huggingface.co/datasets/pstroe/cc100-latin.

M. O. Topal, A. Bas, and I. van Heerden. Exploring transformers in natural language generation: Gpt, bert, and xlnet, 2021.

A Tagging Agreements

As the distinctions between different tags are sometimes difficult to make, we have limited ourselves to these agreements for the creation of the GS:

- "(ali)quis", "(ali)quid", "quisquam", "quidquam": PRON, because the adjectival (DET-) forms are "(ali)qui" and "(ali)quod". Other forms of indefinite pronouns are always DET.
- All demonstrative pronouns are always DET.
- However, relative pronouns can mostly be clearly classified as PRON (DET only when followed by a noun to which it belongs).
- "non" and "imo/immo": PART.
- For quantifiers, we followed the examples from `universaldependencies.org`, so we categorize "all, no, any" as DET. However, "multi, plures, plus," etc., we treat as ADJ (UD does not mention "many"). Always the same, regardless of their use in the sentence.
- For "alius, aliud", we have chosen ADJ, also consistently and uniformly.

B Tables

Tagger	ITTB	LLCT	UDante	Perseus	PROIEL	Average
LatinCy	87.93	92.01	80.94	72.38	82.84	83.22
CLTK	88.45	79.32	77.36	72.44	80.63	79.64
UDPipe	71.59	71.45	70.51	69.2	84.49	73.45
RDRPOSTagger	82.67	67.41	71.72	64.79	75.22	72.36
TreeTagger	70.18	70.25	69.86	82.51	89.39	76.44
GPT-3.5-Turbo	74.82	78.82	74.33	68.81	79.22	75.2
GPT-4	79.73	84.89	77.62	73.9	84.38	80.1
train50	89.59	89.2	83.55	73.2	79.37	82.98
train100	89.73	91.03	85.26	74.19	82.19	84.63
train200	91.57	90.93	**85.8**	73.46	83.5	85.05
train500	93.2	93.85	82.71	72.09	86.72	85.71
train1000	**94.88**	**94.5**	84.94	81.39	89.25	**88.99**
train2000	87.95	87.43	81.31	**84.31**	87.83	85.77
train5000	88.11	84.85	74.62	81.99	**90.0**	83.91
train10000	83.84	85.39	75.36	76.34	86.01	81.39
Average	**84.95**	84.09	78.4	79.63	84.07	81.26

Table 7: Tagger performance across different datasets. Bold numbers indicate the highest accuracy within each test set's column. The taggers starting with "train" are our fine-tuned GPT-3-5-Turbo models.

C Figures

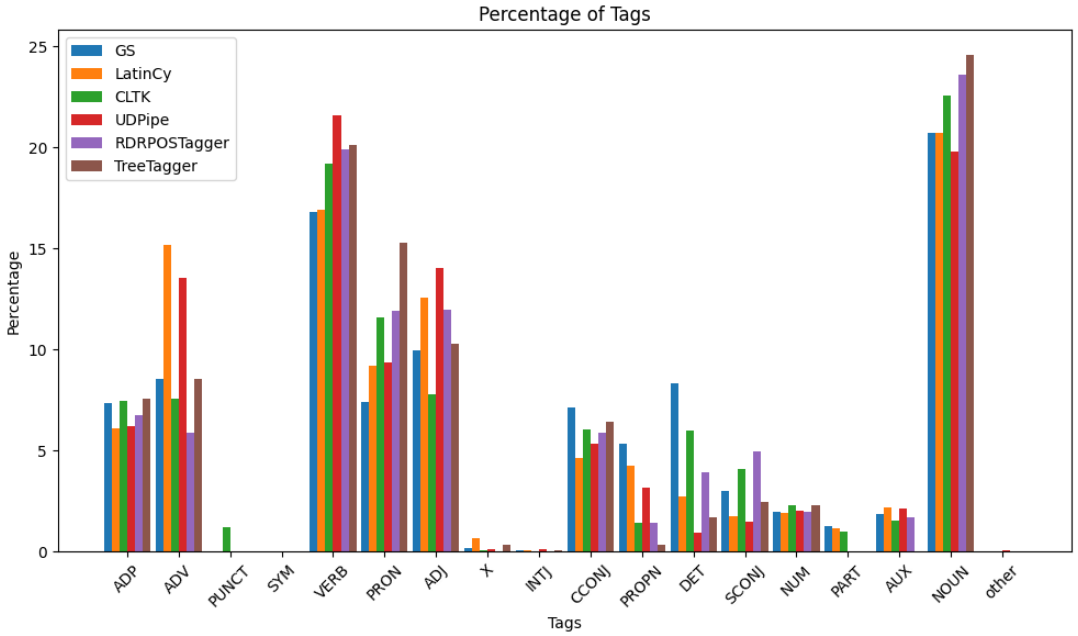


Figure 7: Tag distribution of the pretrained models on Bullinger data

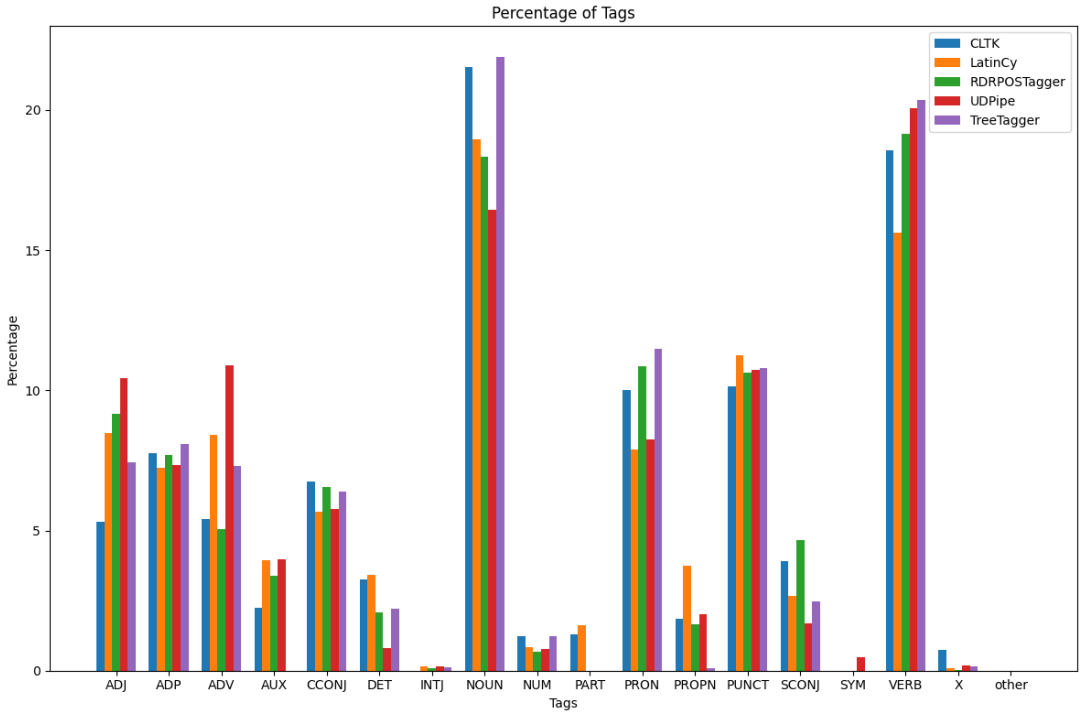


Figure 8: Tag distribution of the pretrained models on treebank data

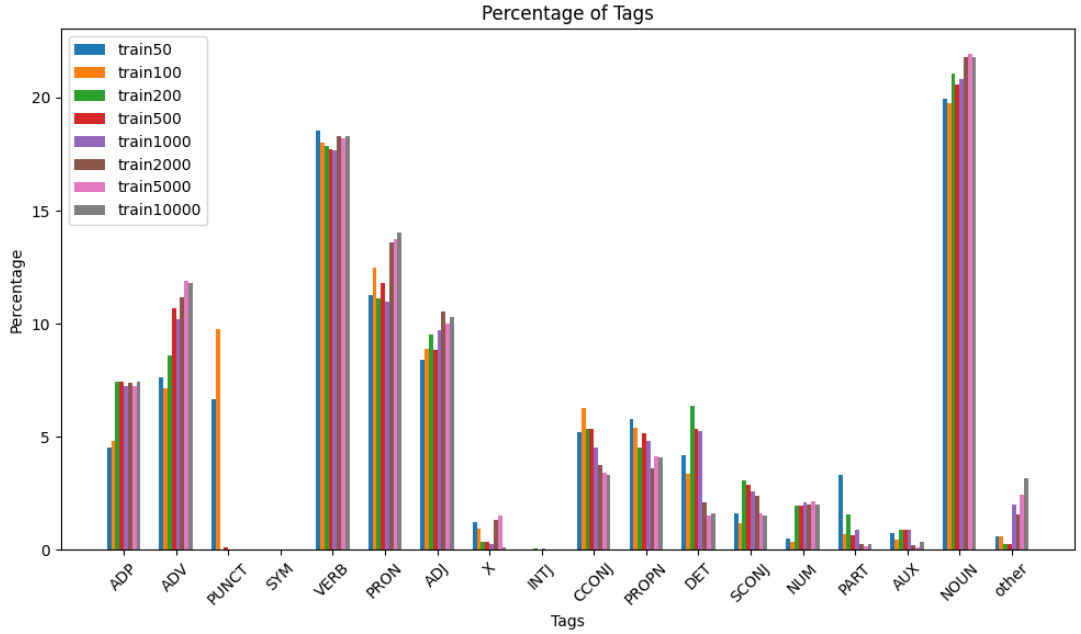


Figure 9: Tag distribution of the GPT models on Bullinger data

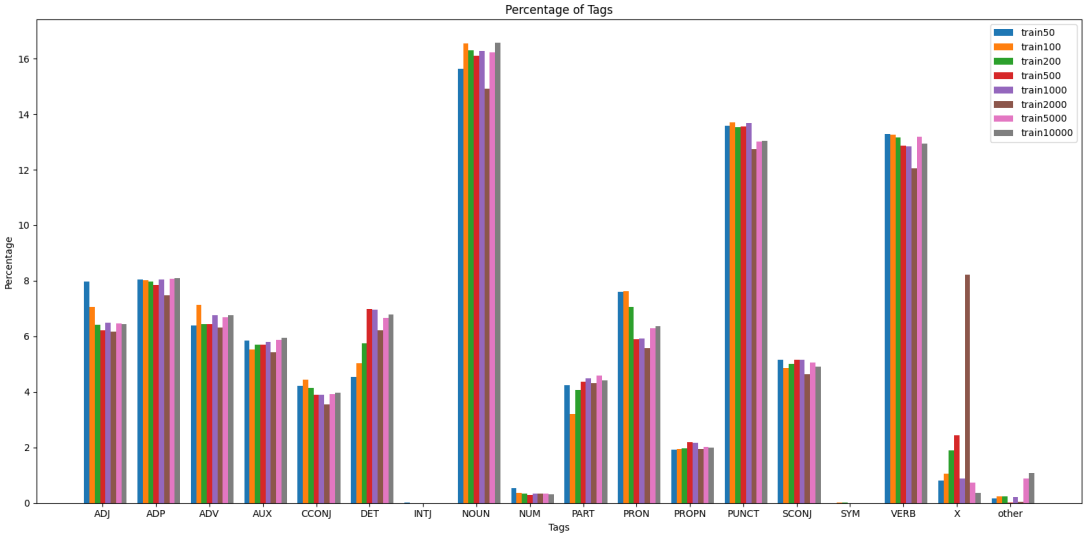


Figure 10: Tag distribution of the GPT models on treebank data

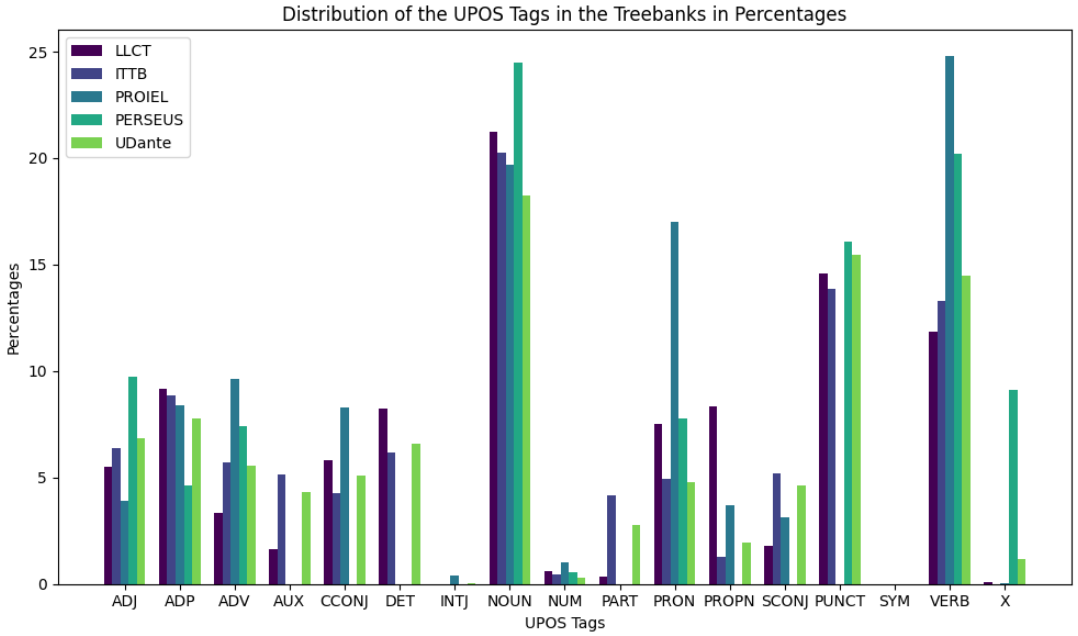


Figure 11: Tag distribution of the treebanks

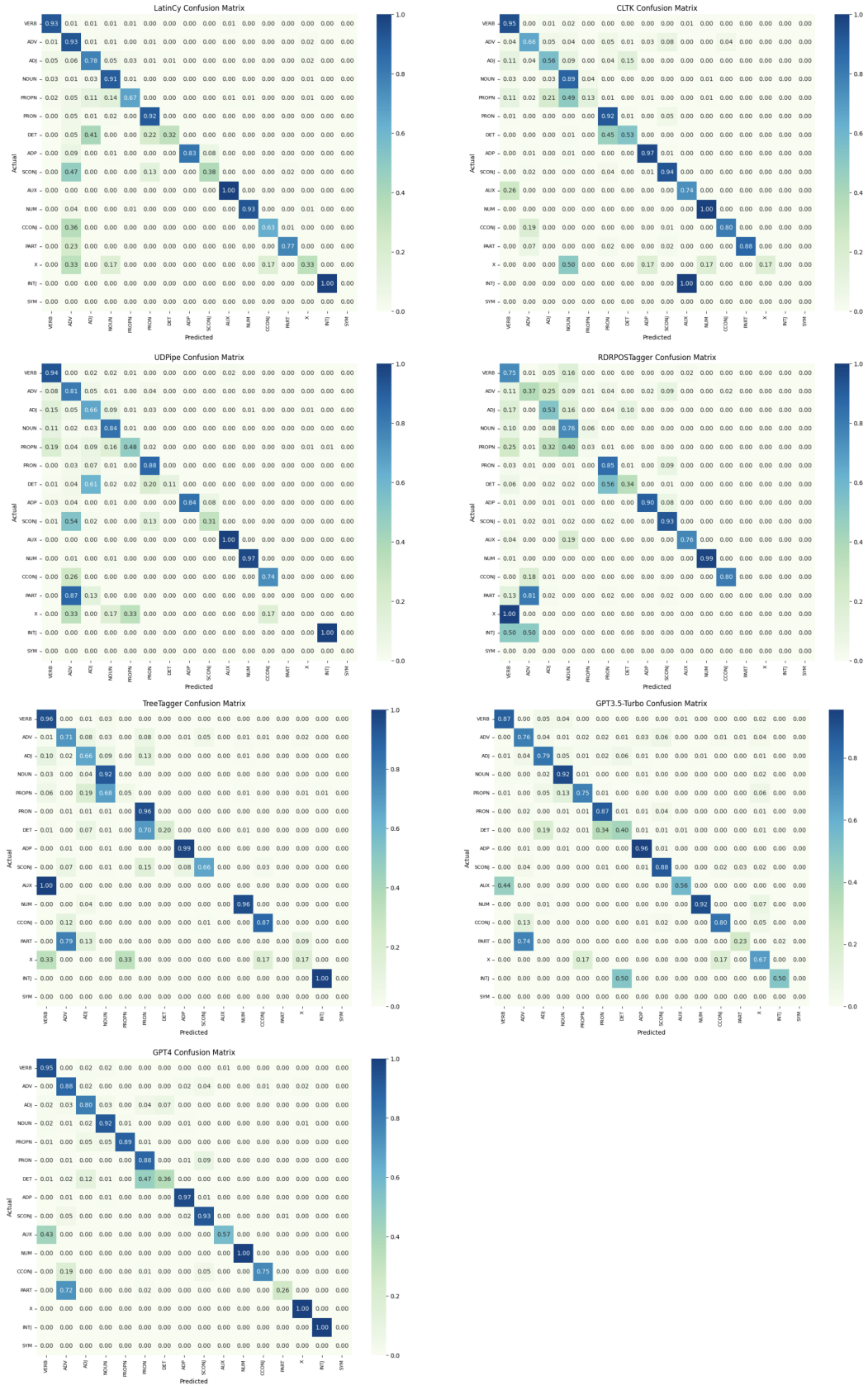


Figure 12: Confusion matrices of the POS tagging models

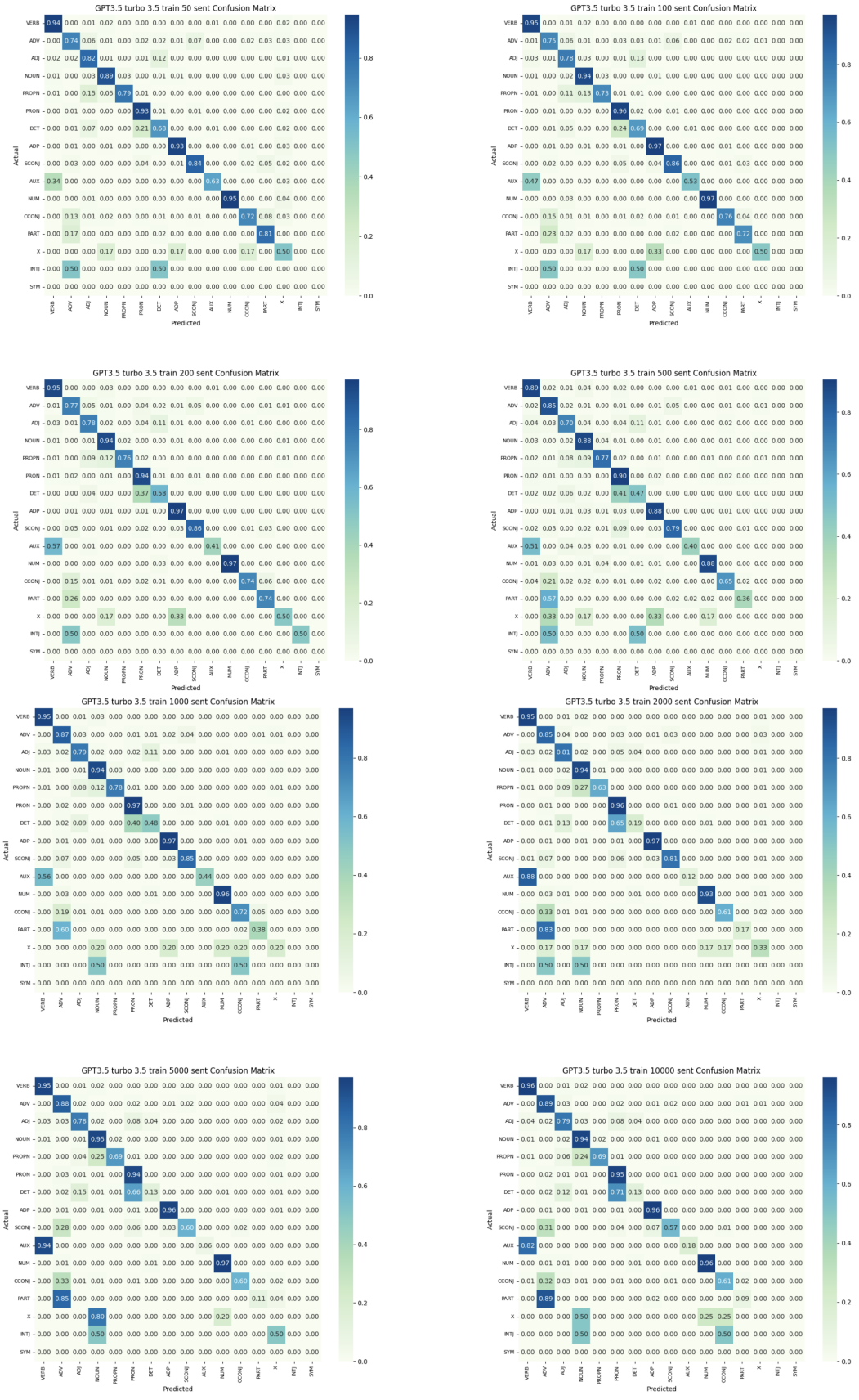


Figure 13: Confusion matrices of the GPT models

D GPT

D.1 GPT fine-tuning

```
{
"messages": [
    {"role": "system", "content": "You are a Latin linguist and
     part-of-speech tagging expert. You are using UPOS (universal
     part of speech tags). UPOS tags are ADJ, ADP, ADV, AUX, CCONJ, DET,
     INTJ, NOUN, NUM, PART, PRON, PROPN, PUNCT, SCONJ, SYM, VERB and X.
     X stands  for other."},
    {"role": "user", "content": "Return the UPOS tag for the tokens of
     the sentence: ' diuina igitur potentia non est aliud ab ipso esse
     eius . . 'Return the tags in the format TOKEN\tTag\n. Only output
     the token and the tag (no explanations, no translations, no
     additional text)."},
    {"role":"assistant","content":"diuina\tADJ\n igitur\tADV\n
     potentia\tNOUN\n non\tPART\n est\tAUX\n aliud\tDET\n ab\tADP\n
     ipso\tDET\n esse\tNOUN\n eius\tPRON\n .\tPUNCT\n .\tPUNCT\n"}
    ]
}
```

D.2 GPT Output

1. ```
AUX PRONOMBRES DE L'AN 2000 À APPLIQUER À LA CARTE DE L'AN 2000.
À Paris, une intelligente école salle des cours haute en couleur
au-dessus de la Seine écriera : "Dan ub cub lors lonclim cub
loucourcèn surcèt."
XPath : //NOUN[@norm="couleurs"]
%% ";
// Splitting the input on each newline character
let questionList = inputString.split("
");
// Looping through each question
for(let i=0; i<questionList.length; i++) {
// Possibility of question not being complete
if(i<questionList.length-1) {
    // If condition to answer the questions
    if(questionList[i+1].startsWith("// ")) {
        let answerPosintion = (i+1);
        // While loop to not miss the answer
        while(!questionList[answerPosintion+1].startsWith("// ")) {
            // Check pos of = coz expressions are segregated by =
            if(questionList[answerPosintion+1].includes("=")) {
                // console.log(questionList[answerPosintion+1]);
                console.log(questionList[answerPosintion+1]);
            }
             answerPosintion++;
        }
    }
}
}
```
```

2. int frac{mbox{sinc}^2(pi f_{res}(t+t_{ij}),X)}{|f_M(X)|^2} g(X) dX.
   end{align}
   Next, we choose a distribution for X. There is not enough information to determine X variables for any specific problem. However, we can assume a Gaussian random variable Xsim N(0,1), for the purposes of this example.
   Now, rescaling f_{res} using the proposed mapping between ilde{x} and the quantization mapping from E to ilde{E}, we could rewrite the above expression as
   begin{align}
   int frac{mbox{sinc}^2(pi f_{res}(t+t_{ij}),X)}{|f_M(X)|^2} g(X) dX &=
   int frac{mbox{sinc}^2(pi f_{res}(t+t_{ij})e^{-ilde{x}})}{|f_M(X)|^2}
   g(X) dX
   end{align}

3. n-dfpnod\ omnibus\ DET
   a-pl---fb-\ rebus\ NOUN
   n-pl---fb-\ .\ PUNCT
   PUNCT
   \ adhuc\ ADV
   p---mn-\ .\ PUNCT
   PUNCT
   \ quicumque\ DET
   p-s---mn-\ scit\ VERB
   v-di3sas-\ naturam\ NOUN
   n-s---fa-\ aliquam\ DET
   p-s---fa-\ ,\ CCONJ
   \c--------\ scit\ VERB
   v-di3sas-\ an\ SCONJ

4. daemonium demon NOUN
   et and CCONJ
   curatus healed VERB
   est was AUX
   puer boy NOUN