UNIVERSITY OF TWENTE.

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE

HUMAN MEDIA INTERACTION

MASTER'S THESIS

Towards a Standard Fine-Grained Part-of-Speech Tagging for Northern Kurdish

---

Student:
Peshmerge Morad

Committee members:
Dr. Nicola Strisciuglio (Committee chair)
Dr. Lorenzo Gatti (1st supervisor)
Dr. Sina Ahmadi (2nd supervisor)

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Data Science

November 9, 2023

Study program:
Data Science & Technology, M.Sc.

Acknowledgements

I would first like to thank my thesis supervisors, Lorenzo Gatti at the University of Twente and Sina Ahmadi at George Mason University. During my thesis, they were always around whenever I ran into trouble or had questions about my research. They consistently allowed this thesis to be my own work but steered me in the right direction.

Finally, I must express my very profound gratitude to my wife, Berfrîn, my parents, and our family and friends for providing me with unconditional love, unfailing support, and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you, spas!

Dedicated to the children of the fire and the sun ....
Ji bo zarokên agir û rojê ....

Abstract

In the growing domain of Natural Language Processing (NLP), low-resourced languages like Northern Kurdish remain largely unexplored due to the lack of resources needed to be part of this growth. In particular, the tasks of Part-of-Speech (POS) tagging and tokenization for Northern Kurdish are still insufficiently addressed by the research community. In this study, we aim to bridge this gap by evaluating a range of statistical and neural network-based POS models specifically tailored for Northern Kurdish. Leveraging limited but valuable datasets, including the revisited Universal Dependency Kurmanji treebank and a novel manually annotated and tokenized gold-standard dataset consisting of 136 sentences ($2,937$ tokens). In this research, we set out to establish both a baseline and a state-of-the-art POS tagger for Northern Kurdish. Challenges such as data scarcity, absence of dedicated research, correct representation of linguistic features, and tokenization methods are carefully addressed through a multifaceted approach involving data refinement, manual annotation, and experimentation with multiple tokenization methods. We propose a POS tagging pipeline for Northern Kurdish where various training and test datasets, POS tagging models, and tokenization methods can be integrated, used, and evaluated. We evaluate our proposed POS tagging models on the novel gold-standard dataset; our transformer-based model outperforms traditional statistical models, achieving an accuracy of 0.87 and a macro-averaged F1 score of 0.77. However, among the traditional statistical models, the CRF model achieves competitive results, 0.84 and 0.74 for accuracy and macro-averaged F1, respectively. This study offers crucial insights into the linguistic peculiarities of Northern Kurdish that affect the performance of tokenization and POS tagging methods and lays down a road map for future work, including dataset expansion and adaptability tests for other Kurdish dialects.

Keywords: Part-of-Speech, Northern Kurdish, Natural language processing, Morphosyntactic analysis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Thanks to the increased access to the Internet worldwide and the rise of social networks, mobile and web applications, and (micro) blogging platforms, tremendous amounts of visual and textual data are generated and shared on the Internet. This rapid increase in textual data has been the leading force in enabling more research in natural language processing (NLP), a field that has multiple tasks, including natural language generation (NLG), sentiment analysis, named entity recognition (NER), and part-of-speech tagging (POS tagging).

NLP is a field that lies at the intersections of linguistics, computer science, and artificial intelligence (AI), and it aims to allow computers to understand human languages. NLP-based technologies power a vast amount of applications we interact with in our daily lives. Virtual assistant agents like Amazon Alexa[1] or Apple Siri[2], typing assistants, smart spelling and grammar checkers like Grammarly[3], and online AI assistants like ChatGPT[4] are all examples of applications powered by NLP and advancements in AI.

However, most of the aforementioned services and technologies are available for a limited group of languages. The advancement in NLP research and technologies for any language is coupled with the availability of resources of that language, be it online textual content, annotated corpora, or any digitally available dictionaries. Therefore, a language can be either classified as a *high-resource* language (HRL), a medium-resource, or a *low-resource* language (LRL), based on the availability of the resources in the digital space. Figure 1.1 depicts the hierarchy of languages with regard to digitally available resources.

On the one hand, English, German, Spanish, Japanese, and French are languages with an abundant amount of (un)annotated data and therefore considered HRLs (P. Joshi et al. 2020; Ruder 2020). On the other hand, languages like Kurdish, Zulu, Maltese, Uralic, Sami, Tigrinya, and many other languages are resource-scarce, less computerized, and lack data and resources needed for developing NLP applications and therefore considered LRLs (Ruder, Søgaard, and

[1] https://alexa.amazon.com/
[2] https://www.apple.com/siri/
[3] https://grammarly.com/
[4] https://chat.openai.com/

Vulić 2019; Magueresse, Carles, and Heetderks 2020; Singh 2008; Cieri et al. 2016; Tsvetkov 2017; Ahmadi 2020a).

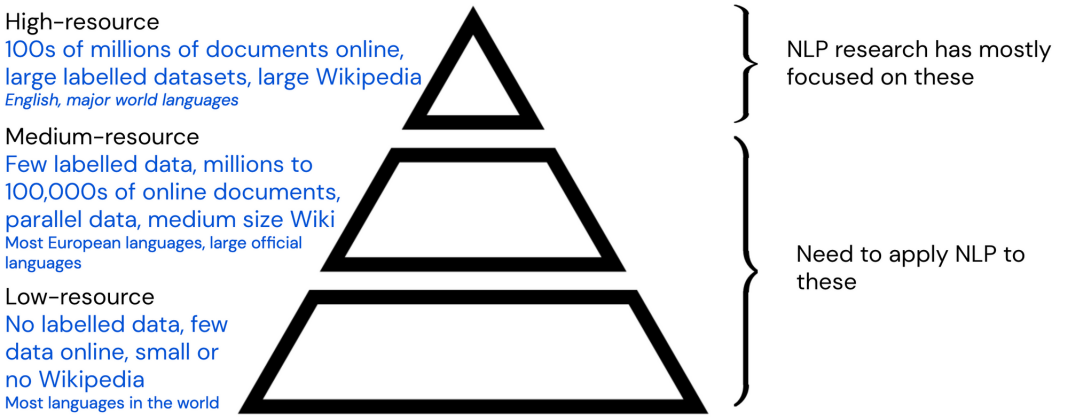


FIGURE 1.1: Conceptual view on the hierarchy of languages with regard to the availability of resources (Ruder 2019)

Traditionally, NLP applications have a pipeline architecture, which means that for a specific NLP task like POS tagging, other NLP tasks are required to be executed first, like normalization and tokenization. In the same way, the task of POS tagging is a building block for many other NLP tasks, like machine translation (MT), question answering (QA), NER, text summarization, and text-to-speech (Gimpel et al. 2010; Schmid 1994; Kanakaraddi and Nandyal 2018; Mitkov 2022; Jurafsky and J. H. Martin 2023).

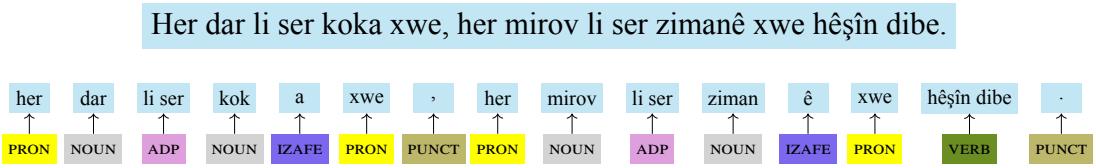


FIGURE 1.2: Manually tokenized and POS-tagged proverb in Northern Kurdish *'As every tree stands over its roots, so does every human blossom with their mother tongue.'*

POS tagging is the process of assigning POS tags to words in any given text and language. POS tags are essential parts of understanding a sentence's structure and underlying meaning, and they are also known as word classes, lexical tags, or morphological classes (Jurafsky and J. H. Martin 2023). The collection of all possible POS tags for a language is called *tagset*. Figure 1.2 shows an example of a manually tagged sentence (proverb) in Northern Kurdish in both raw (untokenized) and tokenized formats[5].

For HRLs, the tasks of tokenization and POS tagging are well-established tasks that have received a great amount of attention from the research community throughout the years. Therefore, there exist state-of-the-art techniques for addressing both tasks effectively. More generally, rule-based, hidden Markov models, conditional random fields, bidirectional long short-term memory, and transformer-based methods have been widely employed for performing POS tagging (Chiche and Yitagesu 2022; Jurafsky and J. H. Martin 2023).

[5]The output of all our POS tagging models for this sentence is provided in figure A.7

The Kurdish language is a good example of an LRL since the resources for building NLP applications are very scarce, and as a result, there is very modest progress in the field of Kurdish-NLP. Kurdish is a morphological-rich language with four main dialects; it belongs to the North-western Iranic branch within the Indo-European languages family, and it is spoken by more than 30 million people (Ahmadi 2020a; Anonby 2022; Ashti Afrasyaw and Kayhan 2021). In this study, we focus on Northern Kurdish, also known as Kurmanji, which is the most widely spoken dialect of Kurdish. We aim to investigate the tasks of POS tagging and, to a lesser extent, tokenization for Northern Kurdish. Further, we discuss the language's various linguistic features and demonstrate the impact of those features on the performance of both tasks if correctly addressed in the training and testing datasets for the tasks. Research related to both tasks is comparatively limited (Jacksi, Ali, et al. 2023; Ashti Afrasyaw and Kayhan 2021).

With the exception of the work of (Walther, Sagot, and Fort 2010) dedicated to the task of POS tagging for Northern Kurdish and the work of (Ahmadi 2020a) that addresses multiple NLP tasks for the Kurdish language, our literature study has shown that there has not been any effective and open source research dedicated to Northern Kurdish. In the studies we surveyed, Northern Kurdish primarily served as a test subject, relying on the only annotated available dataset, the Universal Dependencies (UD) Kurmanji treebank (Gökırmak and Tyers 2017).

1.2 Research questions

In order to clearly delineate the scope and objectives of our study, we have formulated three research questions. These questions guide our investigative process, shaping both the methodology and the analysis of results.

R1 What statistical and neural machine learning models can be effectively used to build a baseline POS tagger for Northern Kurdish?

R2 To what extent can we use the power of transformer-based multilingual language models to create a state-of-the-art POS tagger for Northern Kurdish?

R3 What are the linguistic features of Northern Kurdish that can affect part-of-speech tagging?

1.3 Contributions

Guided by our research questions and the gaps identified in the existing literature, we demonstrate the main contributions of this study. These contributions not only shed light on the complexities of our specific area of investigation but also introduce innovative approaches and methodologies that advance the broader field. Our contributions are as follows:

1. Revisiting and enhancing the UD Kurmanji treebank by reannotating tokens that belong to specific word classes and introducing fine-grained linguistic features of Northern Kurdish.
2. Creating a manually tokenized and POS-annotated gold-standard dataset for Northern Kurdish with a total of 136 sentences and 2,937 tokens.
3. Examining the effect of training data quality with regard to fine-grained linguistic features on the task of POS tagging for Northern Kurdish.
4. Demonstrating the effect of different tokenization methods on testing data for POS tagging with regard to the fine-grained linguistic features.
5. Implementing different POS tagging models and introducing a state-of-the-art transformer-based POS tagger for Northern Kurdish.

1.4 Thesis outline

This introductory chapter is followed by chapter 2, where we provide a detailed discussion about the Kurdish language in general, its dialects, and the linguistic features of Northern Kurdish. In addition, we describe the task of tokenization and POS tagging along with their methods and evaluation techniques. Chapter 3 presents a comprehensive inventory of related work and state-of-the-art studies on the task of POS tagging for LRLs in general and for Northern Kurdish in particular. In addition, we present the currently available datasets for the Kurdish language. In chapter 4, we detail our methodology for the task of POS tagging for Northern Kurdish with regard to the training data, the data enhancements we do, the creation of our own gold-standard data for testing, and the employed tokenization and POS tagging models.
The technical details and implementation of the different POS tagging models are outlined in chapter 5. In chapter 6, we present the evaluation results of the tokenization and the POS tagging models for Northern Kurdish. In addition, we provide a brief discussion on the improvements and shortcomings of the best-performing POS tagging model with regard to the different tokenization methods and the granularity of annotation for training and testing data. Finally, we present the conclusions and future work suggestions in chapter 7.

1.5 Target audience

This thesis is mainly tailored for programmers and computational linguists, presenting an in-depth overview of the topic with sufficient detail for replication or further development. Nonetheless, given its limited focus on the detailed linguistic features of the Kurdish language, linguistic researchers might not find it entirely comprehensive.

Chapter 2

Background

In this chapter, we lay the foundations for different yet related topics for our study. In section 2.1, we give a comprehensive overview of the Kurdish language in general and the Northern Kurdish dialect in particular, delving into linguistic features that are important for the task of POS tagging. In section 2.2 and section 2.3, we detail the task of tokenization and various tokenization methods we use in our study. Consequently, we explain the task of POS tagging along with its approaches in section 2.4 and section 2.5. Finally, we discuss the evaluation metrics for tokenization and POS tagging in section 2.6.

2.1 Kurdish language

The Kurdish language belongs to the Northwestern Iranic branch within the Indo-European languages family, spoken by more than 30 million people. People who speak the Kurdish language are called Kurds or Kurdish people (Ahmadi, Hassani, and McCrae 2019; Ahmadi 2020a; Thackston 2006a; Ashti Afrasyaw and Kayhan 2021). The Kurds are the world's largest ethnic group without a state of their own; they live in homogeneous regions across four countries in the Middle East: Syria, Turkey, Iraq, and Iran (Eppel 2016; Phillips 2017). Kurdish people refer to those regions altogether as Kurdistan.

The Kurdish people and their language have long suffered from targeted state violence, discrimination, and oppression in the aforementioned countries. This ongoing issue has hindered the development and standardization of the Kurdish language for over a century. Unlike other Middle Eastern languages, such as Arabic, Turkish, Hebrew, and Persian, which have thrived more freely, the Kurdish language has faced widespread suppression.
The impact of this oppression has been significant, and the Kurdish community continues to feel the consequences of this discrimination (Allsopp and Wilgenburg 2019; Bozarslan, Gunes, and Yadirgi 2021). As a result, the Kurdish language lacks behind in terms of resources, and it is considered an LRL since there is very modest progress in the field of Kurdish-NLP (Ahmadi, Hassani, and McCrae 2019; Ahmadi 2020a). However, the situation of the Kurds in Iraq has changed drastically since the fall of Saddam Hussein in 2003 and the formation of the federal government of Iraq. The Kurdish region in northern Iraq is officially and internationally recognized as the Kurdistan region (KRG), where Kurdish is considered an official language.

Northern and Central Kurdish are the two major Kurdish dialects with the most speakers among Kurds worldwide (Ahmadi, Hassani, and McCrae 2019; Ahmadi 2020a). The Kurdish language[1] is divided into the following four dialect groups (with corresponding ISO 639-3 languages codes):

- Northern Kurdish or Kurmanji (**kmr**)
- Central Kurdish or Sorani (**ckb**)
- Southern Kurdish (**sdh**)
- Laki (**lki**)

On the one hand, the Northern Kurdish dialect is widely spoken in northern areas of Kurdistan (Syria and Turkey), the northern part of the KRG, and Armenia. It is written using a Latin-based (Hawar alphabet) script introduced by Jeladet Ali Bedirkhan in 1932. Northern Kurdish Latin-based script has 31 letters, eight vowels, and 23 consonants. 26 letters are from the ISO basic Latin alphabet; the remaining five letters come with a diacritic.

On the other hand, the Central Kurdish dialect is spoken in the southern and southeastern regions of Kurdistan (northern Iraq and western and northwestern Iran) and is generally written in a modified version of the Arabic-Persian script (read from right to left). Central Kurdish Arabic-based script has 27 consonants and seven vowels (Thackston 2006a; Ahmadi, Hassani, and McCrae 2019; Ahmadi 2020a; Matras 2019; Anonby 2022). Both alphabets are shown in table 2.1 and table 2.2.

A a	B b	C c	Ç ç	D d	E e	Ê ê	F f	G g	H h	I i	Î î	J j	K k	L l	
M m	N n	O o	P p	Q q	R r	S s	Ş ş	T t	U u	Û û	V v	W w	X x	Y y	Z z

Table 2.1: Northern Kurdish Latin-based script (Hawar) with 23 consonants and 8 vowels.

غ	ع	ش	س	ژ	ز	ڕ	ر	د	خ	ح	چ	ج	ت	پ	ب	ا
ئ	ێ	ی	وو	ۆ	و	ه	ھ	ن	م	ڵ	ل	گ	ک	ق	ف	ڤ

Table 2.2: Central Kurdish Arabic-based script with 27 consonants and 7 vowels.

The following is an example of the same sentence in Northern and Central Kurdish, and English[2]:
Parêzgarê Duhokê daxwaz ji encûmena wezîran dike Akrê bibe paytexta Newrozê.
پارێزگاری دهۆک داوا لە ئەنجوومەنی وەزیران دەکات ئاکرێ بکرێتە پایتەختی نەورۆز.
'The governor of Duhok requests the council of ministers to make Akre the capital of Newroz.'

While Northern and Central Kurdish have the same word order, subject-object-verb (S-O-V), the differences between Northern Kurdish and Central Kurdish are not limited to their

[1] The Kurdish language has ISO 639-3 **kur** language code

[2] The sentence in both dialects is the title of a news item taken from the Kurdish news site Kurdistan24 https://www.kurdistan24.net/

orthography. The differences extend to phonological and morphological levels as reported in (Ahmadi 2021; Thackston 2006b; Thackston 2006a; Esmaili and Salavati 2013).
In contrast to Central Kurdish, Northern Kurdish has the notion of gender (feminine and masculine) and case opposition (absolute and oblique) for nouns and pronouns. In addition, Northern Kurdish has the full ergative alignment in the past tense with transitive verbs, while Central Kurdish does not. Central Kurdish uses pronominal enclitics to achieve the same goal. Moreover, the passive voice (conjugated in all persons, moods, and tenses) in Northern Kurdish is constructed using the verb *hatin* 'to come' and *dan* 'to give' plus the infinitive, while in Central Kurdish, it is created using the verb morphology. Table 2.3 shows the linguistic properties of Northern and Central Kurdish.

	Word order	Passive voice	Gender	Case	Alignment
Northern Kurdish	S-O-V	constructed with *hatin* (to come)	feminine and masculine	nominative, oblique, Izafe, vocative	nominative-accusative, only in past transitive ergative–absolutive
Central Kurdish	S-O-V	morphological	No gender	nominative, oblique, limited Izafe, vocative	nominative-accusative, only in past transitive ergative–absolutive

TABLE 2.3: Linguistic properties of both Northern Kurdish and Central Kurdish.

Before we progress further into our study's theoretical and technical aspects, we delve more into some of Kurdish's finer nuances and linguistic features. We provide a comprehensive analysis of those features using example sentences in Northern Kurdish in the following subsections. However, first, we address an important and prevalent challenge of the Kurdish language in subsection 2.1.1.

2.1.1 Orthographic inconsistencies

Orthographic inconsistencies and standardization in the Kurdish language are particularly pronounced in the context of Northern Kurdish. The primary issue is the absence of a universally accepted, centralized orthographic system for all dialects (Matras, Haig, and Öpengin 2022; Ahmadi 2020b; Gündoğdu et al. 2019). Northern Kurdish lacks a standardized authority for its written form, making it more susceptible to regional variations and irregularities in spelling. As a result, there exists significant divergence in how words are written in different regions where the dialect is spoken.

However, efforts have been made to standardize Northern Kurdish and writing guidelines have been published throughout the years. The most recent books are *Rêbera Rastnivîsînê* (Aydoğan 2012; Weqfa 2019) and *Rêzimana Kurmancî* (Tan 2015). In this work, the grammars of Northern Kurdish discussed and the examples provided are derived from those books, unless stated otherwise.

2.1.2 Noun in Northern Kurdish

Noun in Northern Kurdish in its absolute state and without any suffixes represents the generic and definite senses of the noun. Therefore, we do not use any article in Northern Kurdish to represent the definite state of the noun. However, to make the noun indefinite, we use indefinite markers or suffixes *-ek* for singular and *-in* for plural indefinite nouns. Both markers may be preceded by *-y* and become *-yek* and *-yin* in case the noun ends in a vowel. Table 2.4 shows the indefinite markers in Northern Kurdish. In addition, example 2.1.1 demonstrates the definite and indefinite states along with the indefinite markers for the noun *pirtûk* 'book'.

	Noun ends with a consonant	Noun ends with a vowel
Singular	*-ek*	*-yek*
Plural	*-in*	*-yin*

Table 2.4: Indefinite noun markers based on the number and ending of the noun in Northern Kurdish.

In all examples in this study, we use a hyphen - before a marker to denote that the marker is affixed to a noun. Conversely, we use a dot . before a marker to indicate that the entire token conforms to the case represented by the marker. All symbols used in the examples are listed in table A.1.

Example 2.1.1. *INDF* indicates the indefinite noun marker in Northern Kurdish.

(1)	*pirtûk*	(2)	*pirtûkek*	(3)	*pirtûkin*
	pirtûk		pirtûk-ek		pirtûk-in
	book.F.SG		book-INDF.F.SG		book-INDF.PL
	'the book', 'book' or 'the books'		'a book'		'some books'

In Northern Kurdish, the nouns in the singular form are either feminine or masculine. The gender of each noun must be learned and memorized because of its importance in cases different than the nominative case. However, some nouns may be used as masculine or feminine, depending on the context. From our previous example, the noun *pirtûk* 'book' is feminine, while the noun *heval* 'friend/comrade' could be used for both genders. There is no gender differentiation of the plural nouns in Northern Kurdish. Furthermore, nouns can be inflected in four cases: nominative, vocative و oblique, and construct. The noun's gender and ending determine how it is inflected and what suffixes will be added. Noun's inflection will become clearer in the next subsections as we will introduce the vocative, oblique, and construct cases.

2.1.3 The vocative case

Nouns in the vocative case in Northern Kurdish get suffixes attached to them based on the number and gender. Table 2.5 demonstrates the vocative case markers in Northern Kurdish.

Singular feminine	Singular masculine	Plural
-o	*-ê*	*-no/-ino*

TABLE 2.5: The vocative case markers in Northern Kurdish

Example 2.1.2. Examples of the vocative case markers *VOC* in Northern Kurdish.

(1) *hevalo! were.*

hevalo-o were .
friend-VOC.M.SG come .

'friend, come.'

(2) *keçê! were.*

keç-ê were .
girl-VOC.F.SG come .

'girl, come.'

(3) *hevalino! werin.*

heval-ino werin .
friend-VOC.PL come .

'friends, come.'

(4) *hevalno! werin.*

heval-no werin .
friend-VOC.PL come .

'friends, come.'

2.1.4 The oblique case

Nouns, proper nouns, personal pronouns, and demonstrative adjectives in Northern Kurdish in the oblique case (OBL) undergo a form change. They are either completely altered, or oblique markers are added to the end of the noun. Those markers, shown in table 2.6, are unstressed markers that reveal the gender and number of nouns in the oblique case. The nouns and proper nouns in Northern Kurdish appear in the oblique case in the following situations:

1. The noun is a direct object in the present and future tenses.
2. The noun is a subject of transitive verbs in the past tense.
3. The noun is a second member of the construct case. The construct case is explained in subsection 2.1.5.
4. The noun is a complement of prepositions.
5. The noun comes after demonstrative adjectives in the oblique case.

Furthermore, it is of significance to highlight the case where we have a sequence of nouns or proper nouns connected through the conjunction *û* 'and' in the oblique case; only the last noun gets the oblique case marker, while other nouns will remain in the absolute case. Example 2.1.3 provides a complete list of instances for the nouns in the oblique case.

Demonstrative adjectives in Northern Kurdish appear in both nominative and oblique cases. In the nominative case, the nouns after them appear in the absolute state. On the other hand, in

	Definite	Indefinite
Singular feminine noun	*-ê*	*-ekê*
Singular masculine noun	*-î*	*-ekî*
Plural nouns	*-an*	*-inan*

TABLE 2.6: Oblique case markers based on the number, gender, and definiteness of the noun in Northern Kurdish. If the noun ends with a vowel, the markers will be preceded by a *-y*.

the oblique case, the demonstrative adjectives and the nouns they modify are altered. Table 2.7 summarizes the nominative and oblique case for the demonstrative adjectives.

	Nominative case		Oblique case	
Singular feminine noun	*Ev* 'This'	*Ew* 'That'	*vê* 'this'	*wê* 'that'
Singular masculine noun	*Ev* 'This'	*Ew* 'That'	*vî* 'this'	*wî* 'that'
Plural nouns	*Ev* 'This'	*Ew* 'That'	*van* 'these'	*van* 'those'

TABLE 2.7: Demonstrative adjectives in Northern Kurdish in both nominative and oblique case based on the gender of the noun that appears after them.

The personal pronouns in Kurdish can appear in nominative and oblique cases. Additionally, the possessive pronouns in Northern Kurdish are identical to the personal pronouns in the oblique case. Table 2.8 summarizes the personal pronouns in nominative and oblique cases.

Nominative case	Oblique case	Possessive pronoun
ez 'I'	*min* 'me'	*min* 'my'
tu 'you'	*te* 'you'	*te* 'your'
ew 'she'	*wê* 'her'	*wê* 'her'
ew 'he'	*wî* 'him'	*wî* 'his'
em 'we'	*me* 'us'	*me* 'our'
hûn 'you'	*we* 'you'	*we* 'your'
ew 'they'	*wan* 'them'	*wan* 'their'

TABLE 2.8: Personal and possessive pronouns in nominative and oblique case in Northern Kurdish

Example 2.1.3. Examples of nouns, proper nouns, and demonstrative adjectives in the oblique case in Northern Kurdish. *OBL* and *EZF* indicate the oblique and construct case markers.

(1) *ez hespî dibînim.*

ez hesp-î dibînim .
I horse-OBL.M.SG see .

'I see the horse.'

(2) *ez sêvekê dixwim.*

ez sêvek-ê dixwim .
I apple-OBL.F.SG eat .

'I eat an apple.'

(3) *zarokan ez dîtim.*

zarok-an ez dîtim .
child-OBL.PL I saw .

'the children saw me.'

(4) *Li bin maseyê.*

Li bin mase-yê .
at under table-OBL.F.SG .

'under the table.'

(5) *bala xwe bide van xalan.*

bal-a xwe bide van xal-an .
attention-EZ.F.SG self give these.OBL.PL point-OBL.PL .

'pay attention to these points'

(6) *wêneyê zarokan.*

wêne-yê zarok-an .
picture-EZF.M.SG child-OBL.PL .

'children's picture.'

(7) *enstîtuyên Kurdî yên Stenbol û Amedê.*

enstîtu-yên Kurdî yên Stenbol û Amed-ê .
institute-EZ.PL Kurdish EZ.PL Istanbul and Amed-OBL.F .

'The Kurdish institution of Istanbul and Amed.'

In addition to the oblique case marker for singular and definite masculine nouns or proper nouns in Northern Kurdish, the oblique case can be expressed using the *ablaut* system (Haig 2007). This phenomenon entails vowel alternation to represent morphological distinction or grammatical case change. However, within the context of Northern Kurdish, the *ablaut* is only limited to words that contain either the vowel *a* or *e*. Therefore, the noun in oblique case *hespî* 'horse' in the first sentence within our previous example 2.1.3 could be written as *hêsp*. We give two more examples of this special case in Example 2.1.4.

Example 2.1.4. Examples of definite, singular, and masculine nouns and proper nouns that contain vowels *a* and *e* in the oblique case.

(1) *ez Osmanî dibînim.*

ez Osman-î dibînim .
I Osman-OBL.M see .

'I see Osman.'

(2) *ez Osmên dibînim.*

ez Osmên dibînim .
I Osmên.OBL.M see .

'I see Osman.'

(3) *ez xanîyî ava dikim*

ez xanî-yî ava dikim .
I house-OBL.M.SG built do .
'I build the house'

(4) *ez xênî ava dikim*

ez xênî ava dikim .
I house.OBL.SG built do .
'I build the house'

2.1.5 The construct case (Izafe)

The construct case, often referred to as the *construct state*, represents a grammatical phenomenon in some languages that occurs mostly in a noun phrase where a noun or proper noun undergoes a morphological change to denote ownership or the state of the noun. This altered or inflicted word form is paired with a subsequent word, a noun, or an adjective that specifies the possessor or the sate. This grammatical phenomenon can be found in many languages like Persian, Kurdish, Arabic, and Hebrew (Comrie 2017; Samvelian 2007; Thackston 2006a; Karimi 2007). The English equivalent of construct case in a possessive relationship is the preposition *of* (Gutman 2016; Anonby 2022).

The construct case phenomenon in Northern Kurdish is called *ravekirin* 'to clarify'. The Kurdish term for the construct case phenomenon serves the purpose eloquently since the function of construct case is to give more information and clarify the state, number, and gender of someone or something we are talking/writing about. However, we will be using the term *Izafe*[3] instead, since it is the most common and widespread term used in the literature when handling the construct case in Kurdish.

	Definite	Indefinite
Singular feminine noun	*-a*	*-eke/-eka*
Singular masculine noun	*-ê*	*-ekî*
Plural nouns	*-ên*	*-ine*

Table 2.9: Izafe markers based on the number, gender, and definiteness of the noun in Northern Kurdish. If the noun ends in a vowel, the Izafe markers will be preceded by a *-y*.

The Izafe case in Northern Kurdish is formed with the help of Izafe markers/particles. They are important unstressed suffixes added to the end of the nouns or proper nouns. In its simplest form, the function of the Izafe particle in a noun phrase is to link two words together when there is a relationship between those words. This relationship could be a possessive (between a possessive noun and its possessed noun) or an attributive adjective (between an adjective and the noun it describes) relationship. Table 2.9 shows the different Izafe markers, taking into account the number, the gender, and the state of the noun. It is worth noting that with indefinite nouns, the Izafe markers are added after the indefinite markers, as depicted in the second column of the table.

[3] I/Ezafe/ is derived from the Arabic word إضافة 'addition'.

The first word (*nomen regens* 'governing noun') in the Izafe case could be either a noun or a proper noun. The subsequent word (*nomen rectum* 'governed noun') must be either a noun, a proper noun, a personal pronoun (in the oblique case), an adjective, or a (multi-word) adverb (Thackston 2006a; Tan 2015). In summary, the Izafe case in Northern Kurdish can be found in the following possible situations:

1. A noun or proper noun is immediately followed by:
   (a) another noun or proper noun.
   (b) an adjective.
   (c) the reflexive pronoun *xwe*.
   (d) possessive pronouns.
   (e) the relative pronoun *ku* 'that/who/which'
2. A noun followed by an adverb.

Additionally, the Izafe case is not limited to only two nouns. We can have multiple successive nouns where each noun gets the Izafe marker according to its gender and number. However, the last noun or proper noun will get the oblique case marker as explained in subsection 2.1.4. Furthermore, in a sequence of nouns connected through the conjunction *û* 'and', only the last noun gets the Izafe marker. Other nouns will be in the absolute case.

Moreover, the Izafe markers in Northern Kurdish are not strictly bound to a single position. They may appear both as a suffix and as separate particles that create a new Izafe case by linking a previous Izafe case with a noun, proper noun, adjective, or personal pronoun. In this case, Izafe markers for definite nouns are used; this phenomenon is referred to as *construct extender* because it allows us to extend the Izafe case by adding more adjectives or nouns to the first Izafe case (Thackston 2006a). In example 2.1.5, we provide a set of example sentences covering the entire spectrum of Izafe cases to guarantee the inclusion of all potential variations.

Example 2.1.5. Izafe cases in Northern Kurdish. *EZ* and *OBL* indicate the Izafe and oblique case markers, respectively. Examples marked with * are taken from our annotated gold-standard dataset, which is introduced in section 4.4.

(1) dara guzê.

dar-a guz-ê .
tree-EZ.F.SG walnut-OBL.F.SG .

'the walnut tree.'

(2) parêzgarê Duhokê. *

parêzgar-ê Duhok-ê .
governer-EZ.M.SG Duhok-OBL.F .

'the governer of Duhok'

(3) bajarên mezin.

bajar-ên mezin .
city-EZ.PL big .

‘the big cities.’

(4) destê xwe bişo.

dest-ê xwe bişo .
hand-EZ.M.SG self wash .

‘wash your hand.’

(5) hevalekî min.

heval-ekî min .
friend-INDF.EZ.M.SG my.OBL .

‘a (male) friend of mine.’

(6) keça li jor.

keç-a li jor .
girl-EZ.F.SG upstairs .

‘the girl upstairs.’

(7) kilîla deriyê mala apê min.

kilîl-a deriy-ê mal-a ap-ê min .
key-EZ.F.SG door-EZ.M.SG house-EZ.F.SG uncle-EZ.M.SG my.OBL .

‘the door’s key of my uncle’s house.’

(8) keça ku stranan dibêje, dotmama min e.

keç-a ku stran-an dibêje, dotmam-a min e .
girl-EZ.F.SG who song-OBL.PL say, niece-EZ.F.SG my.OBL is .

‘the girl who sings, is my niece.’

(9) Hevalê birayê min ê bejinbilind ji bajêr hat.

Heval-ê biray-ê min ê bejinbilind ji bajêr
friend-EZ.M.SG brother-EZ.M.SG my.OBL EZ.M tall from city.OBL.M.SG
hat .
came .

‘my brother’s tall friend came from the city.’

(10) Keç û xortên Êzidî di meha Nîsanê de zewacê nakin.*

Keç û xort-ên Êzidî di meh-a nîsan-ê de
girl.F.SG and guy-EZ.PL Yezidi in month-EZ.F.SG April-OBL.F ADP
zewac-ê nakin .
marriage-OBL.F.SG do not do .

‘Yezidi girls and guys do not marry in the month of April.’

2.1.6 Contracted prepositions with third-person singular complements

Kurmanji exhibits four prepositions that contract with third-person singular complements (pronouns in the oblique case ”*wî* ‘him’, *wê* ‘her’), akin to the Celtic languages’ preposition–pronoun combinations and the Spanish term *contigo* ‘with you’ or French *au*= *à le* ‘to the’. Table 2.10 summarizes the contracted prepositions in Northern Kurdish.

Preposition	Third-person complement	Contracted form
li ‘at/in’	*wî* ‘him’ / *wê* ‘her’	*lê*
ji ‘from’	*wî* ‘him’ / *wê* ‘her’	*jê*
bi ‘with’	*wî* ‘him’ / *wê* ‘her’	*pê*
di at/in	*wî* ‘him’ / *wê* ‘her’	*tê*

TABLE 2.10: Contract prepositions with third-person singular complements in Northern Kurdish

Example 2.1.6. Variations of prepositions (contracted) with the third-person singular pronouns.

(1) *ew bi wê re çû bajêr*

ew bi wê re çû bajêr
he with her ADP went city.OBL.SG

‘he went with her to the city’

(2) *ew pê re çû bajêr.*

ew pê re çû bajêr .
he with her ADP went city.OBL.SG .

‘he went with her to the city.’

(3) *min ji wî re got.*

min ji wî re got .
my.OBL from him ADP said .

‘I told him.’

(4) *min jê re got.*

min jê re got .
my.OBL from him ADP said .

‘I told him.’

2.1.7 Contracted prepositions with adverbs and nouns

Northern Kurdish has another contracted preposition case where the preposition is removed, and the preceding adverb or noun gets the particle *î* as a suffix. It is worth mentioning that this case and the previous one are not mandatory in Kurdish. Regular and contracted forms of contracted prepositions can be observed in co-occurrence.

Example 2.1.7. Variations of prepositions (contracted) with adverbs and nouns.

(1) *nêzîk li min.*

nêzîk li min .
close to my.OBL .

‘close to me.’

(2) *nêzîkî min.*

nêzîk-î min .
close-PART my.OBL .

‘close to me.’

(3) *dûr ji mala min.*

dûr ji mal-a min .
far from house-EZ.F.SG my.OBL .
'far from my house.'

(4) *dûrî mala min.*

dûr-î mal-a min .
far-PART house-EZ.F.SG my.OBL .
'far from my house.'

2.2 Tokenization

Before exploring part-of-speech tags and the associated tagging process, we must first address the foundational task of tokenization. Just as part-of-speech tagging serves as a precursor for tasks like syntactic parsing, tokenization is a crucial task in NLP, and it is a prerequisite for POS tagging. Tokenization is the process of segmenting the input text into smaller, distinct units termed *tokens*. These tokens can encompass compound words, single words, sub-words, symbols, or other significant elements. At its most fundamental level, tokenization separates tokens using whitespace as a delimiter (Mitkov 2022, p. 549).

We have seen in the previous section how suffixes play a vital role in the various Northern Kurdish grammatical cases, such as vocative, Izafe, and oblique cases. The case markers are important because they help reveal the grammatical relationships between the different parts of the sentences and the real meaning of the sentence. Additionally, those markers are not just adding meaning to the words in isolation, but they help determine how the words they are attached to fit into the larger grammatical structure of the sentence. Therefore, it is a crucial and non-trivial task to detect word and sub-word boundaries when dealing with texts in Northern Kurdish.

In NLP, tokenization plays an instrumental role in determining the quality and quantity of data to be processed in subsequent tasks. The outcomes produced by various tokenization methods can exhibit significant disparities. Such variations arise not only due to intrinsic properties and design principles of the tokenization method in question but also because of the linguistic structures and idiosyncrasies inherent to different world languages. The implications of these disparities are particularly evident in tasks further down the NLP pipeline. For instance, differences in tokenization outputs can directly influence the data available for POS tagging and the performance of the task as well (Jurafsky and J. H. Martin 2023).

Historically, tokenization followed a relatively straightforward approach, particularly in segmented languages like English or other European languages, where whitespaces served as a dependable indicator of word boundaries. Initial computational models utilized basic rules, making use of spaces, punctuation marks, and other typographical symbols to define tokens. Nonetheless, complexities emerged when dealing with languages lacking clear word boundary indicators, like Chinese, or languages with rich morphological compositions, such as Kurdish, Turkish, or Finnish. In the case of the latter languages, we need more advanced techniques to address the tokenization task in an effective way, ensuring the preservation of the underlying

syntactic relationships among words (Jurafsky and J. H. Martin 2023; Mitkov 2022). Example 2.2.1 shows the output of the widely used NLTK *word_tokenize* tokenizer. We see how the clitic *'m*, the genitive marker *'s* (along with the apostrophes), and the period *(.)* are considered separate tokens despite not having whitespaces between them and the words they attached to.

Example 2.2.1. The output of the popular NLTK library *word_tokenize*[4] tokenizer for the given input sentence.

Input: **I'm writing my master's thesis.**
Output: **[I, 'm, writing, my, master, 's, thesis, .]**

The recent surge of machine learning, and particularly the dominance of deep learning in NLP, has profoundly transformed the field of tokenization. Traditional rule-based tokenization approaches have faced competition from data-driven methods, harnessing extensive datasets to infer token boundaries. A notable shift in this paradigm has been the emergence of subword tokenization algorithms such as WordPiece (Schuster and Nakajima 2012), Byte-Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2016), Unigram language model (Kudo 2018), and SentencePiece (Kudo and Richardson 2018). Subword tokenization algorithms aim to tokenize text at a level that strikes a balance between individual characters and complete words, resulting in a more consistent representation of morphemes or semantically meaningful units. In addition, they played a vital role in the success of the state-of-the-art large language models (LLMs). For instance, Google's BERT model (Kenton and Toutanova 2019) uses the WordPiece tokenization, XLM-roBERTa (Conneau, Khandelwal, et al. 2020) uses SentencePiece, and OpenAI GPT-2 (Solaiman et al. 2019) uses BPE.

2.3 Tokenization methods

In the previous section, we have defined the task of tokenization. In this section, we demonstrate the multiple tokenization methods we are using in this study.

2.3.1 Manual tokenization

The manual tokenization, as the name suggests, is the process of manually tokenizing any given text. This method is mostly performed in pairs with the task of manually annotating tokens with the corresponding POS tags. Despite being very time-consuming, it is considered to have the best outcome because it is done by humans with good linguistic knowledge of the language. Therefore, the manually tokenized text can be considered the ground truth or the gold tokens that can be used for evaluating other automatic tokenization methods.

[4] https://www.nltk.org/api/nltk.tokenize.html

2.3.2 Kurdish Language Processing Toolkit tokenizer

The Kurdish Language Processing Toolkit (KLPT) is a Python-based NLP solution tailored for the Kurdish language (Ahmadi 2020a). It has multiple modules for different NLP tasks like preprocessing, stemming, and transliteration. A pivotal component of KLPT is its tokenization module, referred to henceforth as the KLPT tokenizer. This module, rooted in the research of (Ahmadi 2020b), is distinguished by its design, which encompasses nearly all the linguistic features of the Kurdish language detailed in section 2.1.

2.3.3 Natural Language Toolkit tokenizer

The Natural Language Toolkit (NLTK) is a comprehensive Python-based NLP library. Initially released in 2001, NLTK offers a suite of tools for linguistic data manipulation, including functionalities for classification, tokenization, stemming, tagging, parsing, and semantic reasoning (Bird, Klein, and Loper 2009). Within NLTK, the word_tokenize function has become, through the years, the de-facto tokenization method in NLP for HRLs like English. The NLTK word_tokenize tokenizer leverages the Punkt tokenizer models (Kiss and Strunk 2006), which uses unsupervised machine learning techniques to tokenize sentences across many languages.

2.3.4 Unigram tokenizer

In contrast to the previous tokenization method, the Unigram model is a data-driven subword segmentation technique introduced in the context of neural machine translation (Kudo 2018). The Unigram tokenization model works by iteratively attempting to maximize the likelihood of the training data being segmented into subwords. The model starts with a large vocabulary of potential subwords and, during training, probabilistically removes less likely subwords. The result is a flexible and adaptable subword segmentation that can effectively handle morphological-rich languages and out-of-vocabulary terms.

2.3.5 Byte-Pair Encoding tokenizer

Byte Pair Encoding (BPE) is a subword tokenization method originally introduced for data compression. However, the authors of (Sennrich, Haddow, and Birch 2016) adapted it for neural machine translation to address the issue of out-of-vocabulary words. BPE merges frequent pairs of characters iteratively in the training data, resulting in a fixed-size vocabulary of variable-length symbols that can represent any word in the dataset. This technique has been especially useful in languages with rich morphology or for tasks where a large vocabulary is impractical.

2.3.6 WordPiece tokenizer

Similar to the two previous tokenizers, the WordPiece tokenization model is a subword segmentation method originally developed to address challenges in machine translation (Schuster and

Nakajima 2012). The model operates by iteratively splitting words into smaller units based on frequency counts. Starting from individual characters, the model gradually combines pairs of units, prioritizing those that appear most frequently in the dataset. The process continues until a predefined vocabulary size is reached. This method allows for efficient handling of out-of-vocabulary words and is particularly adept at managing morphologically complex languages, providing a foundation for several subsequent state-of-the-art models in NLP.

2.4 Part-of-Speech tagging

In the previous section, we explained the task of tokenization and highlighted its importance in setting the stage for the task of POS tagging. In this section, we explain POS tagging and the approaches for performing the task.

A POS tag is a word class (also known as a lexical tag or morphological class) that provides significant information about the word and its neighboring words in a sentence. POS tags capture morpho-syntactic behavior, and they are only discrete approximations for the behavior of words in sentences. They are useful cues for the sentence's structure and meaning. POS tags can be categorized into two main categories: closed-class words and open-class words. In the case of English, prepositions and function words like *of, it, and, or* belong to the closed-class POS tags because we rarely add new prepositions to a language. On the other hand, open-class POS tags contain nouns, verbs, adjectives, and adverbs. Those tags are more dynamic, and as a language evolves, more words of this class could be added (Jurafsky and J. H. Martin 2023).

The collection of all possible POS tags for any specific language used in a corpus is called a POS *tagset*. Thus, tagsets can be different depending on the language they belong to. Examples of those tagsets are the Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993), the Claws tagset C7 (Rayson and Garside 1998), and the UD (De Marneffe et al. 2021) tagset. Over the course of years, many attempts were made to establish a consistent, universal, and cross-lingual treebank/tagset annotation for many languages. (Petrov, Das, and McDonald 2012) introduced the first universal POS tagset (UPOS) consisting of twelve common POS tags for 22 languages. Based on the same work, (De Marneffe et al. 2021) introduced the UD for English, where they combined multiple English treebanks with extended POS tags (XPOS). The UD for English consists of seventeen UPOS tags, shown in table 2.11.

POS tagging is the process of assigning POS tags to each word/token in a given text in any language. POS tagging is a disambiguation task because words naturally are ambiguous and can have more than one correct tag depending on the context and their position in the sentence. The task is linguistically very important because it helps identify the grammatical structure of a sentence and the relationships between its constituent parts. This information can be used to analyze the language and gain insights into its structure. In addition, it enables studying the evolution of a language because it helps identify the changes in a language over time, such as word class distribution and changes in word usage.

Automatic POS tagging (hereafter referred to as POS tagging) is a task in NLP performed

	POS tag	Description
Open Class	NOUN	Words to describe persons, places, things, etc.
	PROPN	Proper noun: name of a person, organization, place, etc
	VERB	Words to describe actions and processes
	ADJ	Adjective: noun modifiers describing properties
	ADV	Adverb: verb modifiers of time, place, manner
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.
Closed class	DET	Determiner: marks noun phrase properties
	PRON	Pronoun: a shorthand for referring to an entity or event
	NUM	Numeral
	CCONJ	Coordinating Conjunction: joins two phrases/clauses
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc
	ADP	Adposition (Preposition/Postposition): marks a noun's spatial, temporal, or other relation
	PART	Particle: a function word that must be associated with another word
Other	PUNCT	Punctuation
	SYM	Symbols like $ or emoji
	X	Other

TABLE 2.11: UDT English POS tagset defined by (De Marneffe et al. 2021; Petrov, Das, and McDonald 2012)

by a computer system called *POS tagger*, which is trained (with or without supervision) to automatically label all tokens in any given text with the corresponding POS tag. POS tagging serves many purposes in NLP applications, and it is traditionally considered a building block for other tasks such as NER, information extraction, spelling correction, text classification, NLG, and MT. However, nowadays, within end-to-end neural NLP schemes, unlike traditional pipeline architectures, those NLP tasks are oftentimes performed using one model without being dependent on POS tagging (Gimpel et al. 2010; Schmid 1994; Kanakaraddi and Nandyal 2018; Mitkov 2022; Jurafsky and J. H. Martin 2023).

POS tagging is a type of sequence labeling (SL) task in which each word w_i in any given input sequence W with a corresponding label or tag t_i from output sequence T (Jurafsky and J. H. Martin 2023; Akbik, Blythe, and Vollgraf 2018; Shen, Satta, and A. Joshi 2007). The output sequence is a set of all possible tags given the defined SL task. From a different point of view, an SL task can be decomposed into a set of multi-class classification tasks because the objective of a sequence labeler is to find the most accurate label or class for every token in the input sequence. The classification of the current token is influenced by the preceding and succeeding tokens. Figure 2.1 depicts an SL task for a sentence in Northern Kurdish.

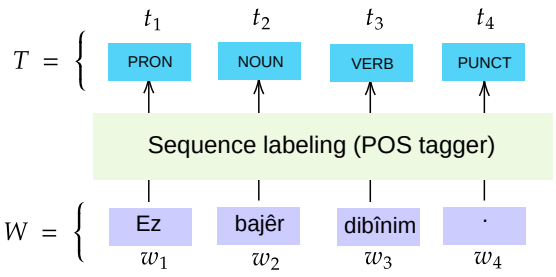


FIGURE 2.1: The task of sequence labeling demonstrated as a POS tagger mapping each word of the input sequence *Ez bajêr dibînim.* 'I see the city.' to corresponding POS tags.

2.5 POS tagging methods

The task of POS tagging can be seen as a multi-class classification task where a model is trained on annotated data to enable it to classify each token in any given sequence of tokens. There are multiple approaches to tackle the task of POS tagging. Generally, those approaches can be grouped into three categories: rule-based, statistical, and neural-based (Jurafsky and J. H. Martin 2023; Kanakaraddi and Nandyal 2018). In the following subsections, we explain those approaches in more detail.

2.5.1 Rule-Based

Rule-based methods are historically the first approaches for POS tagging. They are based on a handwritten and predefined set of rules depending on the linguistic features, such as lexical, syntactical, and morphological information of the specific language the tagger is created for. Linguistic experts mostly create those rules (Kanakaraddi and Nandyal 2018; Mitkov 2022; Jurafsky and J. H. Martin 2023; Chiche and Yitagesu 2022). Examples of rule-based methods are constraint grammar tagging (Karlsson 1990) and transformation-based tagging. Brill's tagger (Brill 1992) is the most commonly used transformation-based tagger that utilizes a hybrid approach of data-driven and rule-based. However, rule-based methods are limited and can not handle unknown or ambiguous words since all rules are predefined. Statistical and neural-based POS tagging methods overcome this limitation by relying on contextual clues.

2.5.2 Hidden Markov Models

A Hidden Markov Model (HMM) is the best-known type of probabilistic/statistical generative model that is used to solve the problem of finding the most likely sequence of states in a finite-state system, given a sequence of observable events or observations (referred to as observables), such as a sequence of input words. The input words are considered *observed events*, while the POS tags are considered *hidden events*. HMM is based on the Markov chain, a model that informs us about the probabilities of sequences of random variables or states whose values can be taken from some set. Those sets are finite, like words, tags, or symbols.

The HMM is a probabilistic finite-state formalism characterized by a tuple $\lambda_T(\pi, A, B)$. T is a finite set of POS tags from a given tagset. π is the initial tag's probability given the first occurring word in sequence. A is the transition matrix, while B is the emission matrix.
Matrix A contains transition probabilities, and matrix B contains the observation likelihoods. The values of A represent tag transition probabilities $P(t_i|t_{i-1})$ of a tag occurring given the previous tag. The maximum likelihood estimate of this transition can be computed as follows:

$$\textbf{Transitions probability: } P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})} \tag{2.1}$$

The values of matrix B represent the emission probabilities $P(w_i|t_i)$. It is the probability of a POS tag t associated with a given word w.

$$\textbf{Emission probability: } P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)} \tag{2.2}$$

A first-order HMM within the context of POS tagging instantiates two assumptions: (1) the Markov assumption and (2) the output independence assumption. Within the Markov assumption, the probability of any given POS tag only depends on the tag before. Formally put:

$$\textbf{Markov assumption: } P(t_i|t_1...t_{i-1}) = P(t_i|t_{i-1}) \tag{2.3}$$

While the Markov assumption tells us that only the previous tag matters when predicting the current tag. The second assumption ensures that the probability of a word w_i depends only on the POS tag t_i that produced the word; formally:

$$\textbf{Output independence assumption: } P(w_i|t_1, ..., t_i, t_T, w_1, ..., w_i, ...w_T) = P(w_i|t_i) \tag{2.4}$$

The training of an HMM for POS tagging depends on our resources according to two types of training: *supervised* or *unsupervised*. For supervised training, the HMM model is trained using an annotated dataset where the words and their corresponding POS tags are known. This allows for a direct estimation of transition probabilities between the tags and the emission probabilities of the words given the tags. Conversely, for unsupervised training, the HMM model is trained using an unannotated dataset, relying only on the words. Since there are no POS tags available in the dataset, the model treats them as hidden variables where their probabilities are estimated using the expectation-maximization algorithm. However, since the unsupervised HMM training is less accurate, we use supervised HMM training in this study.

2.5.3 Conditional Random Field model

One of the prevalent challenges in NLP, in general, and within POS tagging, in particular, is the *unknown words* or more technically *Out-of-Vocabulary* words. New words are oftentimes added to languages in the open-class word category, and in the case of LRLs, the training data may not contain all words. This means that already-built models like HMM cannot recognize

those newly added or unseen words. In addition, generative models, like HMM, cannot capture global knowledge (the entire sequence of words and their associated probabilities), because the transitions and the emissions are all local knowledge (the immediate context in which a word appears). We can solve this problem by using higher-order HMMs; however, this raises the computation cost needed for performing fast POS tagging. Therefore, the conditional random field (CRF) (Lafferty, McCallum, and Pereira 2001) model is introduced to overcome this problem.

CRF is a discriminative model that considers more realistically the inherent lack of complete datasets required to build a robust and wide-coverage statistical model. Given an input sequence (words) $W = w_1...w_n$, the goal of a discriminative POS tagger is to find the most likely sequence of POS tags $T = t_1...t_n$. We want to find the sequence $\hat{T}$ among all possible tags τ.

$$\hat{T} = \arg\max_{T \in \tau} P(T|W) \tag{2.5}$$

CRF uses a set of features to represent each word in the sentence, such as the word itself, its previous and next words, its position in the sentence, and other linguistic features. Then it applies a set of weights to each feature to determine the probability of assigning a particular tag to the word. In addition, CRF utilizes the notion of feature function F, which maps an entire input sequence W (words) and an entire output sequence T (POS tags) to a feature vector. The feature functions allow us to encode any dependency in the data. For example, we can have a function that outputs a one every time it encounters a noun following an adverb; otherwise, it is a zero. Equation 2.5 describes the linear chain CRF, the most common CRF model used for NLP tasks. Linear chain CRF restricts the model to use only the current observed POS tag t_i, the previous one t_{i-1}, and the entire input sequence W at a specific position in the input sequence. This restriction makes the CRF more efficient in the task of POS tagging.

2.5.4 Decision Trees model

A decision tree (DT) is a non-parametric supervised learning method that embodies a graphical representation of the process of decision-making or mapping observations to conclusions. Its structure resembles that of a tree, with internal nodes signifying decisions or tests on specific attributes, branches indicating outcomes of these tests, and leaf nodes representing class labels (for classification tasks) or predicted values (for regression tasks). A DT shines particularly in intricate decision-making scenarios that encompass numerous attributes or features. For the task of POS tagging, a DT is used as a classification tree because the target variable is a discrete set (a finite set of POS tags in the human language) of values (Schmid 1994).

Since the DTs are supervised learning methods, it is required to define a set of features on which the DT will base its decision-making process. For this purpose, there are two important concepts when working with decision trees, Entropy and Information Gain. Entropy is the measure of impurity or disorder in the training data. It is used to quantify the uncertainty in the distribution of class labels across all POS tags. Despite being easy to use and very effective

in addressing classification and regression problems, DTs could suffer from known problems like overfitting, bias, instability, and high variance. Therefore, Random Forests and ExtraTrees (extremely randomized trees) are introduced to overcome those problems.

ExtraTrees is a variation of Random Forest trees, and its goal is to improve the latter by introducing more randomness during the construction of the DTs. ExtraTrees are characterized by random feature selection where at each split node, a random subset of features is considered for finding the best split rather than using all features. In addition, a random threshold is used during each split instead of finding the best threshold. This enhances the model's robustness and increases diversity, automatically leading to more computational efficiency and less overfitting.

2.5.5 N-gram model

Compared to the previously mentioned algorithms for POS tagging, N-gram models offer computational efficiency as they primarily rely on counting word and tag occurrences in the training data, making them faster in both training and prediction compared to other methods. Additionally, due to their probabilistic nature, they can easily handle previously unseen words by falling back on lower-order grams, which offers a form of built-in generalization.

An N-gram model is a probabilistic language model that is used to capture statistical relationships between N consecutive words in any given text. The relationship is addressed by modeling the likelihood of a particular word or sequence of words occurring, given the occurrence of previous words in a sentence or a whole text. The N-gram is based on the assumption that the probability of a word depends only on the preceding $N-1$ words in the sequence, making the probability distribution of the current word conditionally independent of words that are further back in the sequence. This is called the Markov assumption, which is shown in equation 2.3 (Jurafsky and J. H. Martin 2023).

The N in the N-gram model represents the number of consecutive words to be considered when training the model. Unigram (1-gram), bigram (2-gram), and trigram (3-gram) are the most common N-gram models in the field of NLP. The Unigram model is a straightforward approach that assigns tags to tokens solely based on their most frequent POS tags seen during the training phase, independent of their neighbors in the sequence. While this often results in reduced tagging accuracy, it compensates with computational efficiency and simplicity of implementation. These characteristics make it an ideal baseline method for POS tagging.

The training of an N-gram model entails estimating the probabilities of all words or sequences of words based on the frequencies of their occurrences in a corpus. It is worth noting that within n-gram models, we are not calculating the full history of each word; we try to approximate the history by using the preceding n-words. The most straightforward way of estimating the probabilities is using the Maximum Likelihood Estimation (MLE). The MLE is calculated by getting the count of $w_1, w_2, w_3, ..., w_{n-1}, w_n$. Then we normalize the counts by dividing them by the total count of the sequence $w_1, w_2, w_3, ..., w_{n-1}$, which is necessary because we

want to keep the value of MLE between 0 and 1. More formally:

$$P(w_n|w_1, w_2, w_3, ..., w_{n-1}) = \frac{count(w_1, w_2, w_3, ..., w_{n-1}, w_n)}{count(w_1, w_2, w_3, ..., w_{n-1})} \tag{2.6}$$

The numerator and the denominator in equation 2.6 represent the observed frequency, and the ratio is called the relative frequency (Jurafsky and J. H. Martin 2023).

However, in cases where the training data is small, there is a chance that a certain N-gram might not be found in the training data, resulting in a zero value in the numerator in equation 2.6. As a result, the probability will be zero. Therefore, Laplace smoothing is introduced to avoid such cases and to provide more robust and reasonable probability estimates for unseen N-grams. Laplace smoothing is the assumption that each n-gram in any given corpus occurs exactly one more time than it actually does. It involves adding a constant $K = 1$ to the numerator and $|V|$, which is the total number of unique words in the corpus to the denominator in equation 2.6.

2.5.6 Artificial Neural Networks

In addition to the rule-based methods, discriminative (CRF), generative (HMM), decision trees, and N-gram models for POS tagging, artificial neural networks (ANNs) are a common approach to tackle the task of POS tagging and other NLP tasks nowadays. ANNs can extract features automatically from the input text. Thus, there is no need for any predefined set of rules as in rule-based methods or any feature engineering as in CRFs and decision trees models. In addition, ANNs help capture long-term contextual dependencies in input data, leading to better results for POS tagging.

2.5.6.1 The Perceptron algorithm

The concept of neural networks can be traced back to 1943 when Warren McCulloch and Walter Pitts (McCulloch and Pitts 1943) laid the foundation for understanding how simple artificial neurons could mimic certain aspects of the brain's functioning. They introduced a network that could solve a binary problem without active learning. Building on their work, in 1957, Frank Rosenblatt introduced the concept of the Perceptron, a single-layer neural network capable of learning linearly separable patterns by adapting its own weights (Rosenblatt 1958) figure 2.2.

A perceptron is the basic constituent unit of most ANNs and can be viewed as a simplified model of biological neurons. A perceptron has a fixed number of inputs and a single output. Each input has its corresponding weight indicating the influence that input has while computing the output. The output of a perceptron is produced by constructing a linear combination of the input values and their corresponding weights and then applying a nonlinear activation function (step function) such as ReLU, Sigmoid, Tahn, or Logistic.

Formally, let $X = x_1, x_2, ..., x_n$ be the vector of input values and $W = w_1, w_2, ..., w_n$ the vector of corresponding weights. To compute the output, first, we need to calculate the weighted

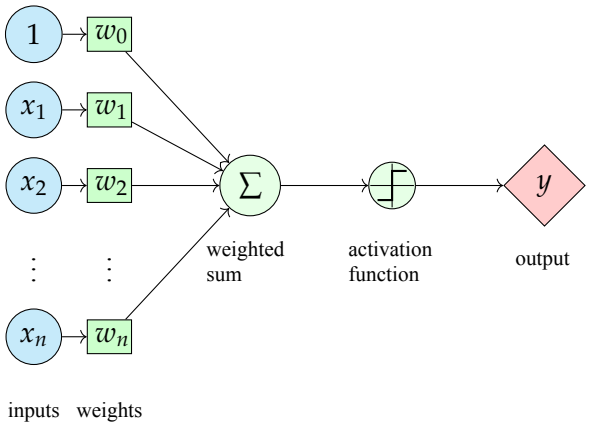


FIGURE 2.2: A perceptron

sum of $\xi = WX$, often referred to as pre-activation or inner potential of the neuron.

$$\xi = \sum_{i=1}^{n} w_i x_i + b = w^T x + b \tag{2.7}$$

Second, we pass the result of ξ to an activation function; the result of this activation function is a binary value as described in equation 2.8.
The term b in equation 2.7 is the bias, and it is replaced by the equivalent input $x_0 = 1$ and its corresponding weight $w_0 = b$.

$$y = \begin{cases} 1 & if \xi >= 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

The weights of the perceptron are changed based on a defined rule where the desired output of the perceptron y is constantly compared with the actual (estimated) output of the perceptron $\hat{y}$. Formally, given training pair (X, y) where is $X = x_1, x_2, ..., x_n$, the weights are adapted using the following formula:

$$w_i = w_i - \eta.(\hat{y} - y).x_i \tag{2.9}$$

The η is the learning rate that plays a vital role in finding the weights that lead to minimizing the difference between the desired and estimated output of the perceptron.

A very effective variation of the perceptron algorithm is the averaged perceptron (Collins 2002). In the context of POS tagging, it serves as a middle ground between the simplicity of classical methods like HMM and CRF and the computational complexity of neural approaches like Bi-directional LSTM. The goal of the averaged perceptron is to find the optimal hyperplane that separates different POS tags in a high-dimensional feature space. Its strength lies in its ability to rapidly converge during training, producing decent results with a relatively quick training cycle. By averaging the weights over the iterations, it mitigates the issue of overfitting, making it more robust on unseen data. It is often used for its efficiency and ease of implementation, especially in low-resource settings. However, in comparison with other methods like

the Bi-directional LSTM, it may lack the capacity to capture long-range dependencies between words.

2.5.6.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) (Rumelhart et al. 1986) are a class of feed-forward neural networks (NN) inspired by the cyclical connectivity of neurons in the human brain for handling ordered sequential data, something feed-forward neural networks like Convolutional neural networks (CNNs) fail to deal with. An RNN takes as input a sequence of items like time series, speech, text, audio, video weather, or financial data and produces a fixed- or variable-size output. RNNs have the concept of *memory* that helps them preserve the (hidden) states or information of previous inputs, which makes them very well suitable for the task of POS tagging since they always consider the data received previously for making any predictions. The memory concept is accomplished because the RNNs have loops; in other words, RNNs can be seen as multiple copies of the same network where each network passes information to the next network.

Formally, given a sequence $W = (w_1, w_2, ..., w_n)$, the RNN updates its recurrent hidden states h_i at time i by:

$$h_i = \begin{cases} 0 & i = 0 \\ \phi(h_{i-1}, w_i), & \text{otherwise} \end{cases} \tag{2.10}$$

ϕ is any nonlinear function such as a logistic sigmoid.
The recurrent hidden state h_i from 2.10 can be updated according to the following:

$$h_i = g(\Omega_w w_i + U_h h_{i-1} + b_h) \tag{2.11}$$

The input sequence W is parameterized by the weights matrix Ω_w, while the hidden states are parameterized by the weights matrix U_h. The Ω_t are the weights matrix for the hidden units to the output units. b_h and b_y are the biases associated with the recurrent and feedforward layers.

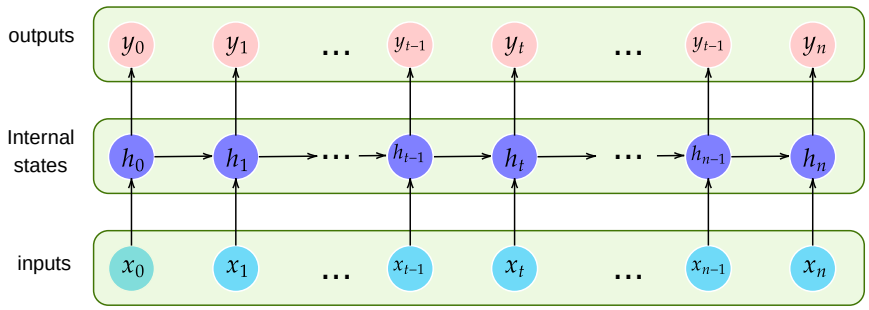


FIGURE 2.3: The architecture of a Recurrent Neural Network with one hidden layer in different time steps t.

The output t at time i can be calculated as follows:

$$t_i = g(\Omega_t \cdot h_i + b_y) \tag{2.12}$$

Another important property of RNNs is that they share parameters across each network layer. While feed-forward networks have different weights across each node, RNNs share the same weight parameter within each network layer. Backpropagation and gradient descent are utilized to ensure the weights are adjusted accordingly. RNNs can use the *backpropagation through time* (BPTT) algorithm to determine the gradients. BPTT is different from traditional backpropagation because of its ability to work with sequential data. It sums the errors at each time step across each layer.

Despite the ability of RNNs to handle sequential data and variable-sized inputs, they can not take into account future input for decision-making as they are not bidirectional. In addition, they can not store information for a long time. They have difficulties with capturing long-term dependencies (Bengio, Simard, and Frasconi 1994). They suffer from the two common problems in deep networks, vanishing or exploding gradient problems where the gradients used to compute the weights updates may get very close to zero or grow exponentially, preventing the network from learning (Goodfellow, Bengio, and Courville 2016). Thus, new variants of RNNs have been proposed to overcome those problems: bidirectional RNNs and Gated RNNs.

2.5.6.3 Long Short-Term Memory

Long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) is another efficient and effective variant of gated RNNs that extends the concept of memory to be able to remember new information, hold previous states or forget information in order to achieve effective processing of sequential data and long-term information access. The LSTM architecture consists of a set of recurrently connected memory blocks or memory cells. Each block contains one or multiple self-connected memory cells and three multiplicative units. The input, output, and forget units (gates) provide continuous analogous of write, read, and reset operations of each LSTM cell. Those gates control access to the memory cells. At each input state, a gate is responsible for deciding how much new input should be saved to the memory and how much current information should be forgotten. The architecture of LSTM enables it to learn long-term dependencies, which is essential in problems like sequence labeling (POS tagging). An LSTM with a forget gate can be formally defined by the following:

$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned} \tag{2.13}$$

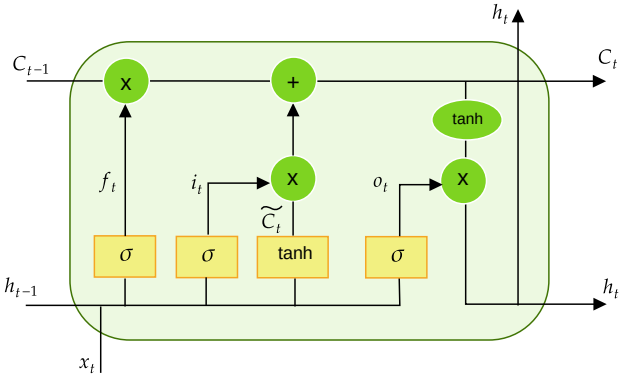


FIGURE 2.4: The architecture of an LSTM cell, adapted from (Chevalier 2018)

f_t, i_i, and o_i respectively refer to the forget, input, and output gates activation vectors at time step t.h_i and c_i are the hidden state (output vector of LSTM) and cell state vectors at time step t. The sigmoid and hyperbolic tangent functions are the σ_g σ_c. The $\odot$ represents the Hadamard product (element-wise product).

2.5.6.4 Bidirectional LSTM

While LSTM models can capture long-term dependencies, they only process the data in one direction, and the data flows only forward. This means that potential combined contextual information from the past and the future will be missed. Therefore, bidirectional LSTM (BiLSTM) models (Graves and Schmidhuber 2005) are introduced to mitigate this shortcoming. The BiLSTMs have two LSTM layers that process information in both directions, enabling them to capture contextual information surrounding a specific data item in a sequence, for example, a word in a given sentence. This property makes the BiLSTM very suitable for sequence labeling tasks such as POS tagging, where context from the previous and following words are crucial for determining the current context.

In contrast to approaches like decision trees where manual feature engineering is required, the BiLSTM models use word, sub-word, or character embeddings such as word2vec (Mikolov et al. 2013), Polyglot (Al-Rfou, Perozzi, and Skiena 2013), or fastText (Bojanowski et al. 2017). Word or sub-word embeddings provide a nuanced input representation capturing both semantic and syntactic elements.

2.5.6.5 Transformer

The Transformer (Vaswani et al. 2017) was introduced in 2017 as the first sequence transduction model wholly based on self-attention. Transformers can be seen as a replacement for recurrent layers used in encoder-decoder architectures. They are the most robust approach for solving sequence-to-sequence tasks while handling long-range dependencies. In addition, the utilization of parallelization makes them faster and more effective than traditional RNNs. The transformer's architecture stacks self-attention and point-wise, fully connected layers for both

the encoder and decoder. The encoder component comprises six stacked identical layers, each with two sub-layers. The first is a multi-head self-attention, whereas the second is a simple, position-wise, fully connected feed-forward network. A residual connection followed by a normalization layer is added to each sub-layer. The architecture's sub-layers and embedding layers produce an output of 512 dimensions.

Like the encoder, the decoder comprises six identical layers. The difference is that the decoder has a third multi-head attention sub-layer that operates over the encoder stack's output. In addition, the self-attention sub-layer is modified to ensure that the predictions for position i can depend only on the known outputs at positions before i. Figure 2.5 demonstrates the encode-decoder structure of the transformer architecture.

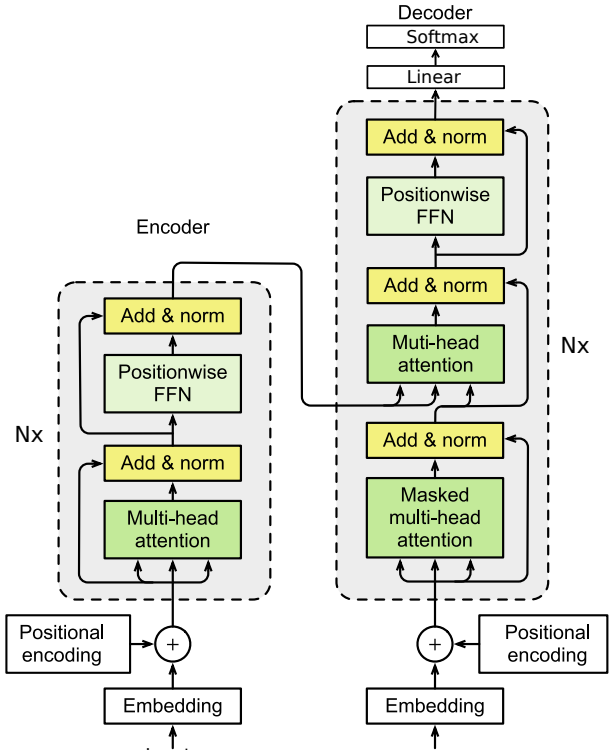


FIGURE 2.5: The encoder-decoder structure of the Transformer architecture, adapted from (Vaswani et al. 2017)

Attention is the key to the transformers' success with sequence-to-sequence tasks. Within the current architecture, the attention is a function that maps a query Q, and a set of key-value pairs (K, V), all vectors, to an output computed as a weighted sum of all values. A compatibility function of the query with the corresponding key computes the weight assigned to each value. The transformers utilize multi-head attention where the queries, keys, and values are linearly projected h times, and then scaled dot-product attention is applied h times in parallel. Afterward, h times outputs are concatenated and projected again.

The advent of transformer-based architectures served as a catalyst for the emergence of large language models (LLMs), revolutionizing the landscape of AI across multiple domains like computer vision, robotics, and NLP. GPT2 (Radford et al. 2019), BERT (Kenton and Toutanova

2019), RoBERTa (Liu et al. 2019), XLM-RoBERTa (Conneau, Khandelwal, et al. 2020), UDify (Kondratyuk and Straka 2019) and most recently Falcon LLM (Penedo et al. 2023) are all examples of recent LLMs with very powerful capabilities. However, despite the fact that it has become a standard to train all those LLMs on multiple human languages, giving them the character of being multilingual or cross-lingual models. They do not always necessarily achieve state-of-the-art results when performing specific downstream tasks like POS tagging compared to monolingual models, especially within LRLs. Nevertheless, they are very adaptive, and they can deliver better results when fine-tuned on monolingual data.

The XLM-RoBERTa (Conneau, Khandelwal, et al. 2020) is a transformer-based LLM that is trained on textual data from 100 languages, including Northern Kurdish. The model aims to offer improved cross-lingual performance and generalization capabilities across a wide variety of languages, making it highly versatile for multilingual tasks. The model has two variants, *base* and *large*. The base variant has a smaller architecture, fewer parameters, and fewer transformer layers, and it is computationally less intensive than the large variant.

2.6 Evaluation

Regardless of the employed tokenization and POS tagging methods, the evaluation of those methods can be expressed statistically using predefined evaluation metrics. The notion of evaluation in the context of POS tagging means evaluating the performance of a POS tagger by comparing its output to a ground truth or a gold standard. Both the output and gold standard are lists of tokens paired with POS tags. While the tags in the output are predicted by the tagger, in the gold standard, those tags are either assigned by a POS tagger or a human annotator.
In the following two subsections, we explain the evaluation of tokenization and POS tagging and what evaluation metrics we use in this study.

2.6.1 Tokenization evaluation metrics

In NLP pipelines, errors in the tokenization stage have a great impact on the task of POS tagging as it relies on accurate token boundaries to classify each token into its corresponding grammatical category (POS tag). Thus, the integrity of tokenization is critical to ensure optimal performance throughout our NLP pipeline. Therefore, evaluating the output of the tokenization methods helps us understand the mistakes the tokenizers make and enables us to reduce error propagation by detecting errors in the very first stages of the pipeline.

The success of any automated tokenization method is measured by the number of tokens it produces in comparison with the expected tokens number in the ground truth. Thus, we distinguish three three performance states of any tokenization method. Consider a tokenization method T that takes in an input text S, O will be a list of output tokens upon applying T to S, and N is a predefined number of expected tokens. The tokenization task will be considered perfect if $|O| = N$, under-tokenized (tokens omission) if $|O| < N$, or over-tokenized (tokens

addition) if $|O| > N$. More formally:

$$T(S) = O = \begin{cases} \text{under-tokenization} & \text{if } |O| < N \\ \text{perfect-tokenization} & \text{if } |O| = N \\ \text{over-tokenization} & \text{if } |O| > N \end{cases} \tag{2.14}$$

Moreover, for evaluation purposes, we distinguish between two types of tokenization evaluation: 1) intrinsic evaluation and 2) extrinsic evaluation. Within the intrinsic evaluation, we want to evaluate the quality of the tokenization system in isolation from the later stages, POS tagging in our case. The intrinsic evaluation directly measures the tokenization system's capabilities by comparing it to similar systems. However, there are no specific evaluation metrics made for tokenization evaluation. Nonetheless, treating the tokenization task as a *machine translation* or a *natural language generation* task gives us the space to evaluate the tokenization performance using the overlap of n-grams between the generated and the gold standard tokens. We follow the same approach of (Ahmadi 2020b) by performing tokenization evaluation using the Bilingual Evaluation Understudy Score (BLEU).

BLEU (Papineni et al. 2002) is an n-gram precision-based measure for measuring the quality of generated texts. It is computed by multiplying the test corpus's precision score's geometric mean by an exponential brevity penalty factor, ρ. In the context of machine translation, the brevity penalty factor is used to penalize the system for generating short yet accurate translations. Given a gold standard tokenized text with length r and a tokenization system output with length c, the brevity penalty and the BLEU score are calculated as follows:

$$\rho = \begin{cases} 1 & \text{if } c > r \\ \exp\left(\frac{1-r}{c}\right) & \text{if } c <= r \end{cases} \tag{2.15}$$

$$\text{BLEU} = \rho \times \exp\left(\sum_{i=1}^{N} w_i \log_{p_i}\right) \tag{2.16}$$

We use $N = 4$ because we want to calculate the metric for 4-grams. The w_n represents the weight for each n-gram; since we are using $N = 4$, the weights will be $w = \{0.25, 0.25, 0.25, 0.25\}$. The p_n represents the precision for each n-gram. BLEU is an objective measure of the system performance; it quantifies the overlap between the n-grams tokens generated by the tokenizer and those present in the gold standard. However, while being fast and simple to use, BLEU does not consider sentence meaning or structure. In addition, it must not be the only utilized evaluation metric as reported by (Reiter 2018). Therefore, we verify our obtained BLEU score by performing an extrinsic evaluation.

Within the extrinsic evaluation, we evaluate the tokenization system by measuring its impact on our whole NLP pipeline. In our case, the tokenization system's quality greatly affects the POS tagger's performance. Therefore, the tokenization correctness can also be determined by examining the F1 and accuracy scores of the POS tagger.

2.6.2 POS tagging evaluation metrics

Several evaluation metrics exist for the POS tagging task, such as accuracy, precision, recall, and F1 score. Accuracy is the percentage of correctly tagged words in the sample. Precision is the proportion of correctly tagged words out of all words that the tagger labeled with a specific POS tag. The recall is the ratio of correctly tagged words out of all words that actually have that POS tag. F1 score is an algebraic mix or the harmonic mean of precision and recall. It reflects the system's robustness by highlighting the high-precision, low-recall, or low-precision high-recall trade-off. The precision, recall, accuracy, and F1 score can be computed at a document, sentence, or tag level. Formally, the metrics can be defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.17}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.18}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.19}$$

$$\text{F1 score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.20}$$

TP, TN, FP, and FN represent true positive, true negative, false positive, and false negative samples, respectively.

In multi-class classification scenarios, when we are dealing with training data with class imbalance, it is highly recommended to use macro-averaged metrics over micro. Thus, we mitigate the effect of some classes being over-represented while others are under-represented. The macro-averaged metrics entail calculating the metrics for every class independently and then averaging the results across all classes. For example. in any given textual dataset, the percentage of word tokens with NOUN tag will be very high, while some other tags such as ADJ might be very low. Therefore, we report the macro-averaged metrics because we want to make sure that each POS tag (class) is treated with equal importance. In this work, we report only the macro-averaged F1 score and accuracy. Taking the equation 2.19 and equation 2.20, the averaged metrics for n classes are defined as follows:

$$\text{Accuracy}_{macro} = \frac{1}{n} \sum_{i=1}^{n} \text{Accuracy}_i \tag{2.21}$$

$$\text{F1 score}_{macro} = \frac{1}{n} \sum_{i=1}^{n} \text{F1 score}_i \tag{2.22}$$

In addition to the aforementioned metrics, the confusion matrix can offer a comprehensive view of a model's performance across all classes. It helps reveal correct and incorrect classifications and where exactly they occur. The confusion matrix provides a deep insight into the tagger performance by revealing how well the tagger handles under-represented classes in the dataset.

Chapter 3

Related Work

In chapter 2, we delved into the linguistic characteristics of Northern Kurdish and explored methods and metrics for tasks of tokenization and POS tagging. With that foundation set, in this chapter, we survey prior research on the task of POS tagging, focusing broadly on its general methodologies and, more specifically, on approaches tailored for LRLs. We categorize the related work according to the learning paradigm. On the one hand, section 3.1 provides an overview of previous research where the statistical or neural POS models are trained from scratch on training data. On the other hand, section 3.2 provides an overview of previous research where fine-tuning an existing language model on new training data has been performed for creating a new POS tagging model. In addition, we provide a list of all available open-source datasets for the Kurdish language in Section 3.3. Finally, we identify the gap in the current research in section 3.4.

3.1 Traditional model training from scratch

(Walther, Sagot, and Fort 2010) has constructed a morphological lexicon (KurLex) and a POS tagger for Northern Kurdish using raw corpora, lexical information, and non-formalized reference grammar. The KurLex contained 22,327 entries that generated 412,320 inflected form entries from multiple online sources like Kurdish Wiktionary, Kurdish Wikipedia, the Open Office spell-checker lexicon, and the Kurdish glossary developed by *Institut Kurde de Paris* and Northern Kurdish-English vocabulary from (Thackston 2006a). The authors designed a tagset for Northern Kurdish with 36 POS tags. The developed POS tagger was built using the MElt tagger (Denis and Sagot 2009), and it can be used as a pre-annotation tool for developing a POS annotated corpus. The tagger achieved 85.7% precision on small evaluation and manually annotated sub-corpus (13 sentences).

Similar to the previous work, however, with a broader focus on the matter of NLP as a whole for the Kurdish language, including both dialects, Northern Kurdish and Central Kurdish, (Ahmadi 2020a) developed a complete NLP toolkit for the Kurdish language (KLPT). It has multiple components to address various NLP tasks, such as preprocessing, tokenization, and transliteration. The tokenizer provided by KLPT stands out as the sole tokenizer proficient in discerning the morphological structure of Northern Kurdish lexemes. It effectively tokenizes

while incorporating intricate linguistic characteristics such as the Izafe, vocative, and oblique case markers. In addition, it has a morphological analyzer that can predict the POS tag for tokens in both Northern and Central Kurdish.

The authors of (Plank, Søgaard, and Goldberg 2016) introduced a unique BiLSTM model with auxiliary loss that effectively performs POS tagging task across 22 languages and achieves state-of-the-art performance, thanks to combining subtoken representations and character embeddings with word embeddings. The proposed method proved to be effective at enhancing the accuracy of infrequent words, leading to notable improvements, especially for languages with rich morphology. The authors used treebanks from the UD project (version 1.2) (Zeman et al. 2022) with their canonical splits for training and testing purposes, and for a treebank to be included, it must have had at least 60K tokens. In addition, The authors found out that BiLSTM models are less sensitive to training data representations, target languages, dataset size, and small label noise.

In a different work, the authors of the Uppsala system (Smith et al. 2018) introduced a pipeline consisting of three components: a joint word and sentence segmentation component, a POS and morphological features predictor, and a dependency trees predictor based on words and tags. They included 82 treebanks from the UD project (Zeman et al. 2022), including Northern Kurdish. The authors used a BiRNN-CRF for word boundary tag prediction for the first component, complemented by an attention-based LSTM model. The LSTM was used for transducing non-segmental multiword tokens. For the second component, they utilized a character BiLSTM that combined the full context of the given sentence with a Meta-BiLSTM. The authors used a BiLSTM for tokens' representation learning and thereafter trained with a multilayer perceptron for the third component. For the task of POS tagging for the Northern Kurdish, the authors reported an accuracy of 61.33% evaluating the UD Kurmanji treebank test split (Gökırmak and Tyers 2017).

In a parallel work to the work of (Smith et al. 2018) with regard to multilingual training using UD treebanks, the authors of (Qi, Zhang, et al. 2020) introduced an open-source language-agnostic neural pipeline for various NLP tasks, named Stanza. It supports 66 languages, including Northern Kurdish, and it can perform multiple NLP tasks like tokenization, multi-word token expansion, lemmatization, POS tagging, and dependency parsing. The author trained the pipeline on 112 datasets, including UD treebanks (UD Kurmanji treebank was also included). The introduced pipeline uses architecture proposed by (Qi, Dozat, et al. 2018), where the trained a sequence-to-sequence model by utilizing a BiLSTM encoder with an attention mechanism in the form of a multilayer perceptron. In addition, they used word2vec (Mikolov et al. 2013) or fastText (Bojanowski et al. 2017) embeddings. The authors of Stanza evaluated the pipeline for various NLP tasks on the UD treebanks. They achieved a macro-averaged F1 score of 57.17% for POS tagging for Northern Kurdish.

In contrast to the previous research, the authors of (Eskander, Muresan, and Collins 2020) utilize an unsupervised cross-lingual transfer approach for the task of POS tagging for LRLs using the texts of the Bible as parallel data. The authors' goal is to create a POS tagger for a target language without direct supervision only by relying on parallel translations between one

or more source languages. The proposed POS tagger is a BiLSTM model that expects a labeled sentence. The sentences are a concatenation of word and sub-word representations based on two types of word embeddings. The first type is pre-trained contextualized embeddings generated by the XLM-RoBERTa (Conneau, Khandelwal, et al. 2020) model. The second is randomly initialized embeddings. Treebanks' test splits from the UD project (De Marneffe et al. 2021) were used for evaluating the proposed POS tagger.

Similar to the previous work regarding the use of parallel corpora, the authors of (Imani-Googhari et al. 2022) introduced a novel approach for tackling unsupervised POS tagging for LRLs called Graph Label Propagation. The labels from multiple source HRLs to target LRLs are simultaneously transferred. The authors used translations of a sentence in multiple languages from the Parallel Bible Corpus to create a graph with words as nodes and alignment links as edges by aligning words for all language pairs. Then, they utilized a Graph Neural Network to propagate POS labels associated with source language nodes to target language nodes. The authors evaluated their method on multiple treebanks from the UD project (De Marneffe et al. 2021).

The authors of (Fang and Cohn 2017) demonstrate the effectiveness of a bilingual dictionary for performing POS tagging for LRLs without the need for parallel corpora. The method requires a bilingual dictionary for knowledge transfer, modest monolingual corpora for both the HRL and the LRL, and a small annotated corpus of around 1000 tokens for the LRL. They introduce a joint neural model for multi-task learning, distant cross-lingual supervision, and ground truth supervision. The first component of the model is a BiLSTM POS tagger based on cross-lingual word embeddings trained on the monolingual HRL corpus and the bilingual dictionary. Then, the POS tagger annotates the unannotated text in the monolingual LRL corpus, resulting in data called *distantly supervised data*.
The second component is used to model the manually annotated data (1000 tokens) of the LRL. It utilized the same model architecture from the first component in addition to an extra hidden perceptron output layer with *tanh* activation function and a softmax output transformation. Combining both components ensures the errors in the first component are accounted for and corrected in the second one. The introduced method (joint multi-task learning) performs better than supervised and distant cross-lingual learning methods with English as a source and ten European languages as a target.

Regarding unsupervised POS tagging, the authors of (Cardenas et al. 2019) developed an unsupervised, language-agnostic POS tagging strategy that does not require human-labeled data or parallel corpora. They consider the task of POS tagging a clustering problem where they utilize a two-step pipeline to find output POS tags T for any given input word sequence W via cluster sequence C. This method requires having an HRL as a parent language with labeled POS training data and an LRL as a target language.
The probability of output POS tags is approximated by the tag distribution from the parent language. Then, the probability of the estimated cluster given output POS tags can be calculated using the noisy-channel, expectation-maximization algorithm.
For training and testing, the authors used the UD (De Marneffe et al. 2021). The method

achieves SOTA tagging performance for both Sinhalese and Kinyarwanda languages.

In a different and recent work by (Pradhan and Yajnik 2023), the authors experimented with HMM, CRF, and LSTM models for the task of POS for Nepali. It is considered an LRL, but compared to Northern Kurdish, it has more data and is more established considering the socioeconomic and political situation of Nepal. The authors used a tagged corpus called Nepali Monolingual Text Corpus ILCI-II, which has a total of 424716 tokens paired with corresponding POS tags. From the experiments conducted and subsequent results, it was observed that the Bi-LSTM models surpassed the performance of the CRF and HMM models by a significant margin, exceeding 10%.

3.2 Fine-Tuning pretrained language models

In the study published by (Vries, Wieling, and Nissim 2022), the researchers investigated the efficiency of cross-lingual transfer learning using large multilingual pre-trained models, particularly for low-resource languages without labeled training data. The researchers argued that for many LRLs, as the name suggests, there is no available data for model fine-tuning for specific tasks in those LRLs. They use part-of-speech (POS) tagging data as a primary evaluation metric due to its availability in a variety of languages, including low-resource ones. Factors such as language families, writing systems, and pre-training are evaluated for their impacts on cross-lingual training. They fine-tuned the pre-trained multilingual XLM-RoBERTa base model (Conneau, Khandelwal, et al. 2020) for the task of POS tagging using 56 source languages and tested it on 105 target languages. Northern Kurdish was one of the target languages in their experimental setup. The researchers concluded that a simple fine-tuning for any multilingual LLM on English does not harness the full cross-lingual potential of the model. In addition, source and target languages that are members of the same family tend to be helpful for achieving better results for the task of POS tagging.

In recent research based on the work of (Vries, Wieling, and Nissim 2022), the authors of (Mollanorozy, Tanti, and Nissim 2023) verified the findings from the previous research by focusing the tasks of POS tagging and NER in two ways. First, using Persian as a source language and other closely related languages as a target. Second, using those target languages as a source while using Persian as a target language. The authors experimented with two transformer-based models, ParsBERT (Farahani et al. 2021) and XLM-RoBERTa (Conneau, Khandelwal, et al. 2020). They have found that Persian seems to be a reasonable source language for Northern Kurdish for effective POS tagging. One more interesting finding from their research was the fact that the monolingual ParsBERT model outperformed the pre-trained multilingual XLM-RoBERTa model on the selected tasks. This research verified the findings of an earlier study by (Vries, Bartelds, et al. 2021).

Similar to the work of (Fang and Cohn 2017), however, without relying on bilingual dictionaries, the authors of (Bao et al. 2019) propose a model with multi-head self-attention (Vaswani et al. 2017) mechanism called Multilingual Language Model with deep semantic Alignment

(MLMA). The proposed approach differs from previous works by using contextualized representations instead of word-level alignments and without using any bilingual resources. The authors train the MLMA on monolingual corpora from each language and align its internal states across different languages. Then, the MLMA generates a language-independent representation to bridge the gap between HRLs and LRLs. After training the MLMA, its parameters are fixed, and the hidden states are extracted as cross-lingual contextualized representations (CLCRs). Afterward, a sequence labeling model is built upon those CLCRs using a joint character level LSTM and linear-chain CRF model. The approach is evaluated on multiple European languages and distant language pairs like English-Chinese. The results are competitive with previous methods, which use large-scale parallel corpora.

In other work similar to (Bao et al. 2019) regarding the utilization of the transformer architecture, researchers (Kondratyuk and Straka 2019) introduce UDify. This semi-supervised multi-task self-attention model automatically produces UD annotations in any supported UD language. The authors use the multilingual pre-trained BERT model (mBERT) (Kenton and Toutanova 2019), trained on Wikipedia data from 104 languages. By concatenating all available training sets, they train and fine-tune the BERT model on the 75 UD languages (including Northern Kurdish). Sentences from those languages are fed into the model to produce contextual embeddings. They introduce task-specific layer-wise attention and decode each UD task simultaneously using softmax classification. In addition, they apply a heavy amount of regularization, such as input masking, increased dropout, weight freezing, and discriminative fine-tuning. The authors conclude that fine-tuning BERT on all UD treebanks (multilingual learning) delivers superior results than fine-tuning monolingually, especially for LRLs, even ones without training data at all, via zero-shot learning. A second round of fine-tuning on monolingual treebanks can improve the model's performance by using BERT weights saved from UDify's multilingual training. In addition, the authors report an accuracy of 53.36% for the UPOS tagging for Northern Kurdish.

In other work, the authors of (Doostmohammadi, Nassajian, and Rahimi 2020) have used the *Izafe* (discussed in detail in subsection 2.1.5), the unstressed morpheme that can appear at the end of nouns and proper nouns in Persian and Kurdish, as a means to improve the POS tagging for Persian. They consider both tasks of *Izafe* recognition and POS tagging as sequence labeling problems. The class space for the former is either 0 for words without or 1 for words with *Izafe*, while for POS tagging, the class space is 14 coarse-grained POS tags. The authors used the Persian Bijankhan corpus, which contains 10 million words. The authors experimented with CRFs, RNN-based (BiLSTM), and transformer-based models. The *Izafe* information provided for POS tagging helped the first two models to achieve good results, while it was not beneficial for transformers.

In a different study by (Muller et al. 2021), the authors focus on 15 low-resource and typologically diverse languages (Uralic languages and Indo-European) and members of the Bantu, Semitic, and Turkic families). Those languages were unseen by mBERT (Kenton and Toutanova 2019) and XML-R (Conneau, Khandelwal, et al. 2020). The authors demonstrate the diversity of behavior depending on the script, the amount of available data, and the relation

to pretraining languages. They specifically focus on the critical role of the script in the transfer abilities of multilingual language models and how transliterating to a script used by a related language seen during the pretraining significantly improves downstream tasks. For some languages, the authors used deduplicated datasets from the OSCAR project (Suárez, Sagot, and Romary 2023); for others, they used Wikipedia dumps.
The researchers compare the performance of their model with the non-contextual models like LSTM-based UDPipe future system(Straka 2018) for POS tagging and Stanza(Qi, Zhang, et al. 2020) for NER. They categorize the chosen languages into three categories: easy, intermediate, and hard. Easy languages are those where the mBERT-base model outperforms the baseline; intermediate ones need Mask-Language-Model tuning on the mBERT. And finally, the hard languages are the ones where the mBERT model underperforms the baselines. One of the chosen languages in this research is Central Kurdish. The authors use transliteration to convert the data from the Central Kurdish (Persian-Arabic) script to the Latin script. Northern Kurdish was used by the authors as the target language for the transliteration of Central Kurdish. The transliteration helped boost the model's performance for the NER task for Central Kurdish from 75.6% to 82.7%.

The authors of (L. Martin et al. 2020) investigate the feasibility of training a monolingual (French) Transform-based language model for multiple downstream tasks like POS tagging, dependency parsing, NER, and natural language inference. They introduced CamemBERT, a model based on BERT that uses the same architecture as RoBERTa (Liu et al. 2019), and it is trained using the French part of the open-source corpora from OSCAR (Suárez, Sagot, and Romary 2023). The authors compare the newly introduced CamemBERT with multiple multilingual language models like multilingual mBERT (Kenton and Toutanova 2019), cross-lingual language models (XLMs) $XLM_{MLM-TLM}$ (Conneau and Lample 2019), UDify (Bao et al. 2019), UDPipe Future (Straka 2018), and UDPipe Future + mBERT + Flair (Straka, Straková, and Hajič 2019). In addition, they used the official 2019 French Wikipedia dumps for pretraining the model. CamemBERT achieves SOTA results on the aforementioned four downstream tasks. In addition, the authors succeed in proving that using web-crawled data with high variability is preferable to using Wikipedia-based data. Besides, they prove that transformer-based monolingual models could reach surprisingly high performance with as low as 4GB of pretraining data. The latest finding questions the need for large-scale pretraining corpora, and it paves a new way for training monolingual contextual language models for LRL.

In a similar work to the CamemBERT, the authors of (Straka, Náplava, et al. 2021) introduced RobeCzech, a monolingual contextualized language representation model based on the RoBERTa (Liu et al. 2019), and trained solely on Czech data. Their introduced model outperformed different multilingual language models for Czech, like multilingual BERT(Kenton and Toutanova 2019), XLM-RoBERTa(Conneau, Khandelwal, et al. 2020) and SlavicBERT (Arkhipov et al. 2019). RobeCzech has a total of 125 million parameters and was trained on publicly available Czech data with more than 500 Million tokens. The authors evaluated the model on five NLP tasks, including POS tagging and NER on publicly available treebanks. For all the tasks, RobeCzech achieved a state-of-the-art performance and outperformed the

previously mentioned language models, except for POS tagging, where the reported accuracy, 99.36%, was equal to the accuracy of the XLM-RoBERTa model. However, RobeCzech has approximately 78% fewer parameters than XLM-RoBERTa.

In an earlier study by the authors of (Vries, Wieling, and Nissim 2022), (Vries, Bartelds, et al. 2021), they investigated the adaptability of monolingual language models for LRLs using zero-shot transfer learning. They have adapted three monolingual BERT-based English, Dutch, and German models (source languages) and mBERT to two LRLs, Gronings and West-Frisian. Their strategy was retraining the lexical layers of the BERT-based models using data from the two target languages. The base models were fine-tuned for the task of POS tagging. The authors found that mBERT may not facilitate transfer to LRLs, while monolingual BERT models showed a great performance even with very little data, especially if the source and target languages are relatively similar. The authors have used the lexical-phonetic LDND measure for calculating the language similarity. For the task of POS tagging, the proposed models for both Gronings and West Frisian scored the highest when the monolingual Dutch Bert model BERTje was used as a base model.

Parallel to the study described in (Qi, Zhang, et al. 2020), the authors of (Nguyen et al. 2021) introduced a light-weight **Tran**sformer-based tool**kit** named Trankit. It is a multilingual NLP trainable pipeline pre-trained on 90 UD treebanks (Nivre et al. 2020) for 56 languages, including Northern Kurdish. The authors fine-tuned both variants (base and large) of multilingual pre-trained transformer-based XLM-RoBERTa (Conneau, Khandelwal, et al. 2020) on all treebanks together. The procedure of simultaneous fine-tuning on all treebanks was possible thanks to the utilization of Adapters (Pfeiffer, Rücklé, et al. 2020; Pfeiffer, Vulić, et al. 2020). They are small networks injected in the last layers of the pre-trained transformer-based models; they are specialized to capture language- and task-specific nuances.
For the validation of Trankit, the authors selected the same treebanks that had served as the basis for fine-tuning. Their evaluation extended to various downstream tasks, such as POS tagging and NER. Specifically for the POS tagging task in the context of Northern Kurdish, they disclosed macro-averaged F1 scores of 74.33% and 75.07% for the base and large configurations of XLM-RoBERTa, respectively.

In other work specifically performed for the Maltese language as an LRL, the authors of (Micallef et al. 2022) utilized the power of mBERT to perform morphosyntactic and semantic classification tasks such as dependency parsing, POS tagging, NER, and sentiment analysis, respectively. In addition, they introduce a new unlabeled corpus for Maltese (Korpus Malti v40) with almost 500M tokens. The data in the corpus is collected from specific online and offline sources. Further, the authors introduce two new language models pre-trained for Maltese: monolingual (BERTu) and multilingual (mBERTu). The authors pre-train the same models on Maltese Wikipedia data as baselines to examine the training data quality on the language models' performance for LRLs. The authors concluded that BERTu and mBERTu trained on their newly created corpus perform better than the same models trained on Wikipedia data. In addition, the monolingual BERTu outperforms the multilingual mBERTu in general, and they achieve a 98.5% accuracy score for the POS tagging task.

In more recent research similar to (L. Martin et al. 2020) with regards to the utilization of a monolingual setup for BERT with relatively small training data, researchers (Gessler and Zeldes 2022) introduced MicroBERT. It is based on BERT but has 1.19% of a BERT base model's size, characterized by a small parameter count, and it produces monolingual embeddings that can outperform comparable multilingual approaches. The authors hypothesized about the benefit the monolingual models can gain from multitask pretraining with auxiliary tasks incorporating labeled data. In addition, they demonstrate that monolingual BERTs for LRL are extremely overparametrized in the sense that many components within the architecture can be either removed or compressed. MicroBERT's performance was evaluated on a diverse group of seven LRLs chosen carefully based on several criteria. One criterion was that the languages must have UD treebanks with a train, dev, and test split, and they must have between 500K and 1M unlabeled tokens.

3.3 Available datasets

In the previous two sections, we provided an extensive overview of related work in the field of POS tagging for LRLs including Northern Kurdish. In this section, we provide an overview of the four available open-source datasets available for the Kurdish language and its Northern and Central Kurdish dialects.

3.3.1 Open Super-large Crawled Aggregated coRpus

Open Super-large Crawled Aggregated coRpus (OSCAR) dataset (Suárez, Sagot, and Romary 2019) is a multilingual dataset extracted from Common Crawl snapshots aiming to provide web-based multilingual resources and datasets for a wide variety of NLP tasks. The latest version of OSCAR contains corpora for 152 languages, including unannotated Northern Kurdish and Central Kurdish. Information about Kurdish within OSCAR is found in table 3.1

Language	Language code	# Documents	# Tokens
Central Kurdish	ckb	182,508	61,334,746
Northern Kurdish	kmr	80,338	25,921,607

TABLE 3.1: Statistic of the Kurdish language within the multilingual corpus OSCAR (Suárez, Sagot, and Romary 2023)

3.3.2 Universal Dependencies Kurmanji treebank

Universal Dependencies (UD) (De Marneffe et al. 2021) is an open-source project that seeks to develop cross-linguistically consistent treebank annotation of morphology and syntax for multiple languages. In 2015, the first version of the dataset was introduced, and it consisted of 10 treebanks for 10 languages. The data presented in UD is annotated data and consists of UPOS

(universal POS tags), XPOS (language-specific POS tags), Feats (universal morphological features), lemmas, dependency heads, and universal dependency labels. The latest version of UD was released in 2022, and it contains 243 treebanks in over 138 languages (Zeman et al. 2022).

The UD project has a Northern Kurdish (UD Kurmanji) treebank that contains morpho-syntactic information (POS tags and some morphological features). The data in the treebank is drawn from fiction and encyclopedic data in roughly equal measure. It consists of the Kurdish translation of *The Adventure of the Speckled Band* story and sentences from the Northern Kurdish Wikipedia. UD Kurmanji has been annotated in accordance with the UD annotation scheme (Gökırmak and Tyers 2017). Table 3.2 summarizes the properties of the UD Kurmanji treebank.

Language	Language code	# Sentences	# Tokens	Tagset
Northern Kurdish	kmr	754	10,260	18 POS tags

Table 3.2: Statistic of UD Kurmanji (Gökırmak and Tyers 2017)

3.3.3 Multilingual Open Text

Multilingual Open Text (MOT) (Palen-Michel, J. Kim, and Lignos 2022) is an unlabeled multilingual corpus that contains text in 44 languages, many of which are considered LRL. The corpus contains over 2.8 million news articles and an additional one million short snippets (photo and video) from the Voice Of America (VoA)[1] published in the period of 2001-2022. VoA has websites in 47 languages, and it started with publishing news in Kurdish in 1992. MOT contains news articles in both Kurdish dialects, Northern Kurdish and Central Kurdish. Information about the size and the amount of news articles is shown in table 3.3

Language	Language code	# Articles	# Tokens
Central Kurdish	ckb	55,062	-
Northern Kurdish	kmr	40,100	-

Table 3.3: Statistic of Kurdish language within the multilingual corpus MOT (Palen-Michel, J. Kim, and Lignos 2022)

3.3.4 Pewan-a Kurdish corpus and test collection

The Pewan corpus (Esmaili, Eliassi, et al. 2013; Esmaili and Salavati 2013) was the first standard test collection for evaluating Central Kurdish information retrieval systems. The team Pewan has followed TREC's standard test collection construction methodology for building the corpus. The corpus contains Northern and Central Kurdish documents collected from two multilingual news websites, Peyamner[2] and VoA. However, considering this corpus uses documents from VoA, there is a high probability that there is an overlap in content between Pewan

[1] https://www.voanews.com/
[2] https://peyamner.net/

and MOT in 3.3.3. The latter corpus has more content because the time range extends to 2022.

Language	Language code	# Articles	# Distinct tokens	# Tokens
Central Kurdish	ckb	115,340	50,1054	18,110,723
Northern Kurdish	kmr	25,572	127,272	4,120,027

TABLE 3.4: Statistics of Pewan Corpus for Kurdish language (Esmaili, Eliassi, et al. 2013; Esmaili and Salavati 2013)

3.4 Bridging the research gap

In Section 1.2, we defined our research questions concerning POS tagging methods and linguistic features of Northern Kurdish that affect the task. For the first two research questions, we focus on machine learning techniques that can be used for creating a POS tagging model for Northern Kurdish taking into account relatively simple methods like Unigram and CRF and more advanced approaches like BiLSTM and transformer-based methods.

Our literature review reveals several insights into POS tagging for LRLs, and Northern Kurdish in particular. In section 3.1 and section 3.2, we have seen that the techniques vary from statistical to neural approaches for POS tagging, and the learning paradigms where creating POS tagger can be created by training models from scratch which happens to be the case for statistical (CRF) and neural (BiLSTM) approaches, or by fine-tuning existing large language models using Transformer architecture. The latter approach is the most widely used in recent years and delivers competitive results compared to the first approach.

We observe that the work of (Walther, Sagot, and Fort 2010) is the only study focusing on POS tagging for Northern Kurdish, where the authors test their proposed POS tagger on a set of 13 manually annotated sentences. In addition, we see that the UD Kurmanji treebank (Gökırmak and Tyers 2017), listed in subsection 3.3.2, is the only available annotated dataset for Northern Kurdish. Based on our literature review, we observe that this treebank has been employed in multilingual pipelines for training and predominantly for testing purposes.

Chapter 4

Methodology

In chapter 3, we provided an overview of state-of-the-art techniques in POS tagging, tracing their historical evolution to understand the advancements and innovations in the field. In addition, we provided a list of available datasets for the Kurdish language. In this chapter, we will delve into our methodology, addressing various sub-tasks that fall within the broader scope of this study. In section 4.1, we provided a more detailed overview of the UD Kurmanji treebank. Then, we explain why and how enhanced the UD Kurmanji treebank in section 4.2, and in section 4.3 showcase the resulted treebank. In section 4.4, we introduce our manually annotated gold-standard dataset and explain the annotation procedure. Finally, we present and motivate our POS tagging pipeline for this study in section 4.5.

4.1 The UD Kurmanji treebank

We already introduced the UD Kurmanji treebank (Gökırmak and Tyers 2017) in subsection 3.3.2. In this section, we provide more insights into the treebank by discussing its technical details and analyzing the distribution of POS tags. Then, in section 4.2, we discuss the treebank's limitations with regard to the task of POS tagging and present our approach to mitigating those. However, we only list the limitations of UD Kurmanji that we have addressed during this study.

The treebank uses the CoNLL-U format, which is a revised format of the CoNLL-X format (Buchholz and Marsi 2006). The annotations are encoded in plain text files with UTF-8 file encoding A sentence that is annotated in this format would have the following elements:

- Sentence-level comments that start with the hash symbol #. In the case of UD Kurmanji, we have two comments: the source of the sentence and the raw (untokenized) sentence in Northern Kurdish.
- Word lines where each line represents a single token in the sentence along with the token's syntactic information spread over at most ten tab-separated columns.
- A blank line to denote the sentence boundary.

Listing 4.1 shows a sentence in Northern Kurdish drawn from the UD Kurmanji. It provides a clear insight into how the treebank is annotated, providing detailed information about each

token, such as the token's UPOS, XPOS, lemma, gender, grammatical case, and relations with other tokens. This level of detail makes the treebank suitable for building a supervised POS tagger. The creators of the treebank have followed the guidelines of the UD guidelines[1] by splitting the treebank into two splits: 1) train split that contains only 20 sentences, and 2) test split that contains 734 sentences.

```
# sent_id = wiki:wikibank.vislcg.txt:371:10279
# text = Rêveberên mezin têne ber wî û jê aqil digirin.
1   Rêveberên   rêveber NOUN    n   Case=Con|Definite=Def|Gender=Masc
2   mezin   mezin   ADJ adj Degree=Pos  1   amod    _   _
3   têne    hatin   VERB    vblex   Mood=Ind
4   ber ber ADP pr  AdpType=Prep    5   case    _   _
5   wî  ew  PRON    prn Case=Acc|Gender=Masc
6   û   û   CCONJ   cnjcoo  _   10  cc  _   _
7-8 jê  _   _   _   _   _   _   _
7   jê  ji  ADP pr  AdpType=Prep    8   case    _   _
8   _   ew  PRON    prn Case=Acc|Gender=Fem,Masc
9   aqil    aqil    NOUN    n   Case=Acc|Definite=Def|Gender=Masc
10  digirin girtin  VERB    vblex   Mood=Ind
11  .   .   PUNCT   sent    _   3   punct   _   _
```

Listing 4.1: A sentence from the UD Kurmanji original treebank in CoNLL-U format. Some data columns are omitted for better readability.

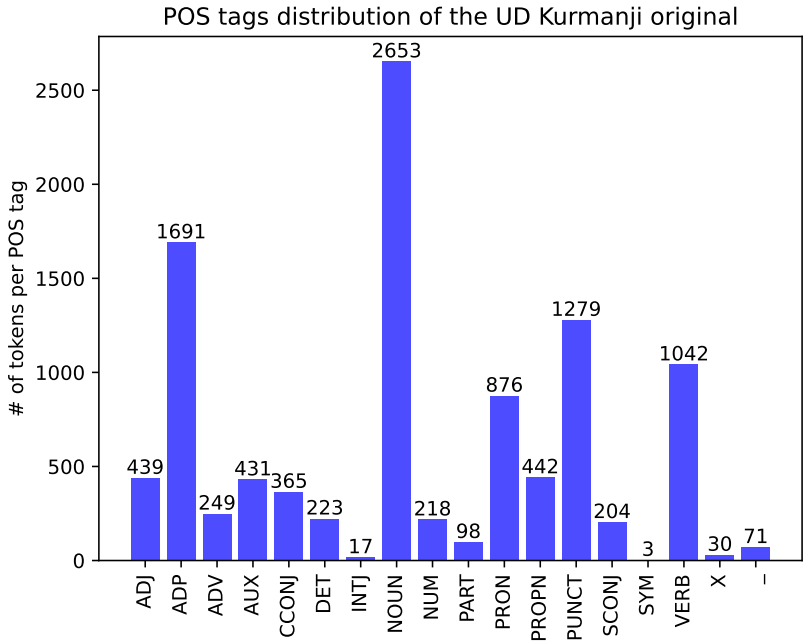


Figure 4.1: Number of tokens per POS tags present in the UD Kurmanji original (Gökırmak and Tyers 2017)

In this study, we merge both splits of the treebank and use all sentences for training purposes. We call this version of the treebank UD Kurmanji original because we use the data as is without performing any data alterations. We conduct a basic data assessment to understand the

[1] UD guidelines

distribution of POS tags in the treebank. Figure 4.1 shows the count of tokens in the treebank for each POS tag. Notably, 26.86% of the tokens are labeled as NOUN, which is not surprising given that nouns are typically predominant in many natural language corpora. This trend is also observed in other UD treebanks, like the UD English Web Treebank (Silveira et al. 2014). In addition, we see the presence of the - POS tag, which is explained in detail in subsection 4.2.1.

In the chapter 3, we have reviewed multiple studies that have used the UD Kurmanji test split for evaluating cross-lingual learning approaches for multiple downstream tasks, including POS tagging (Vries, Wieling, and Nissim 2022; Eskander, Muresan, and Collins 2020; Nguyen et al. 2021; Kondratyuk and Straka 2019; Straka 2018; Qi, Zhang, et al. 2020).

4.2 UD Kurmanji refinement

In this section, we outline our methodology for enhancing the UD Kurmanji treebank with regards to the Northern Kurdish's linguistic properties listed in section 2.1. We refer our enhanced version as UD Kurmanji revisited. Both the original and revisited treebanks are only used for training purposes.

In the case of Northern Kurdish, the indefinite, Izafe, and oblique case markers are important as they have a syntactic role, and they are very helpful in improving POS tagging itself and other tasks like syntactic parsing and alignment for machine translation. Therefore, they must be presented explicitly and correctly in the training and testing datasets for the task of POS tagging. One possible approach to present those markers in the dataset, as suggested by (Jurafsky and J. H. Martin 2023), is to combine the multiple tags for each syntactic word separated by a plus sign or a hyphen. For example, considering the the indefinite noun in the Izafe case given in example 2.1.5 *hevalekî* 'a friend', its POS tag would be either NOUN-DET-IZAFE or NOUN+DET+IZAFE, because the noun *heval* 'friend' has the indefinite marker *ek* followed by the Izafe marker for singular masculine nouns *î*.

However, in this study, we take another approach for presenting those markers in the dataset by splitting the markers from the tokens they are attached to. Thus, introducing new tokens. Taking the same example we showed before, we split *hevalekî* into three separate tokens, each with its corresponding POS tag: *heval* as NOUN, *ek* as DET, and finally *î* as IZAFE.
Our approach for enhancing the UD Kurmanji treebank bears a close resemblance to the research described by (A. R. M. M. D. Seddah 2023). The authors made significant steps in addressing tokenization issues to ensure consistency in the NArabizi[2] treebank annotations. For instance, they carefully segmented specific classes of words, such as determiners in noun phrases.

Having explained the reason behind our approach for enhancing the UD Kurmanji original. In the following sub-subsections, we showcase our approach for addressing those case markers and the outcome of each enhancing action in terms of POS tags count.

[2]The NArabizi treebank (D. Seddah et al. 2020) is created for the user-generated content variety of Arabic Algerian, which is known for its frequent usage of code-switching.

4.2.1 Contracted prepositions

The recent guidelines of the UD (Nivre et al. 2020) state that the basic units of annotations are syntactic words, not phonological or orthographic words. This means that contracted forms must be split off. For example, in French, *au* 'to the' must be presented as two separate tokens *à le*, or in German *zum* or *zur* must be tokenized into *zu dem* and *zu der*, respectively.

The creators of the UD Kurmanji treebank followed that specific UD guideline and applied it to the contracted preposition (*jê, lê, pê, tê*) in the treebank. We explained the contracted prepositions in Northern Kurdish in subsection 2.1.6). In the example given in listing 4.1, we see that the line with 7-8 has *jê*, which is contracted and its normal form could be either *ji wî* 'from him' or *ji wê* 'from her'. However, based on the context of the sentence in the corpus, it must be presented as *ji wî*. Nevertheless, the creators do not specify the third-person compliment and instead provide a hyphen - as a token instead, as shown in listing 4.2 on line 8.

```
1 7-8 jê  _   _   _   _   _   _   _   _
2 7   jê  ji  ADP pr  AdpType=Prep    8   case    _   _
3 8   _   ew  PRON    prn Case=Acc|Gender=Fem,Masc
```

Listing 4.2: Example of contracted preposition handling in UD Kurmanji original

Therefore, we remove the lines (7-8) and (8) and keep only the line (7). By looking at figure 4.1, we see that the treebank has 71 instances of such contracted prepositions annotation. Removing all instances from the treebank is a straightforward process that involves searching and removing the found lines that meet our removal rules.

4.2.2 Indefinite noun markers

As described in subsection 2.1.2 and demonstrated in table 2.4, the suffixes *-ek* and *-in* are attached to the end of the nouns in the nominative case in Northern Kurdish to denote indefiniteness. However, in both the oblique and the Izafe case, indefinite markers will be placed before the case markers. In the UD Kurmanji original treebank, indefinite markers are not separated from the nouns. Therefore, we separate those markers. In listing 4.1, the indefinite noun *kibrîtek* 'a matchstick' has the marker *-ek*, therefore it must be separated. A new line is introduced where the token is the marker *ek*, its lemma is *yek*, and its POS tag is DET. Within the scope of this enhancement task, we only separate indefinite markers in the nominative case. We obtain 53 markers from the treebank.

4.2.3 Izafe case markers

In subsection 2.1.5, we explained in a detailed manner the Izafe case, its markers, and cases. We explained that the Izafe markers could appear in a sentence in two separate forms, either: 1) as a suffix for nouns or proper nouns, or 2) as a separate marker known as *the construct extender*. Similar to the indefinite noun markers, the Izafe markers from the first form in the

UD Kurmanji original are not split. In addition, the Izafe markers from the second form are annotated in the treebank as ADP.

For both forms, we follow the list of all possible Izafe rules defined in subsection 2.1.5. For the first form, we preprocess the treebank and obtain 1,189 Izafe markers. It is worth noting that we apply strict rules for recognizing and splitting the Izafe markers. For example, the token *wesiyetname* 'testament' is a feminine noun that ends with a vowel; thus, in the Izafe case, the Izafe marker must be preceded by a *y* and therefore written as *wesiyetname-ya*. However, in the treebank, the word appears as *wesiyetnam-a*. Despite the fact that the last letter is the Izafe marker *-a*; we consider it as an invalid case of the Izafe marker and therefore ignore it.

Moreover, since the Izafe markers from the first case can be added to both definite and indefinite nouns, we split off the indefinite markers simultaneously. For example, considering the following Izafe case from the treebank *armanceke civakî* 'a social purpose', the noun *armanc-ek-e* contains an indefinite marker *-ek* followed by the Izafe marker *-e*. Therefore, the noun is split off into three tokens; the first is the noun itself *armanc*, the second represents the indefinite marker *ek* along with its POS tag DET, and the third represents the Izafe marker *e* with its POS tag IZAFE.

For the second form, revisiting the Izafe markers of the second form is relatively less complex than the first one. We preprocess the treebank according to the second form and obtain 99 Izafe markers. In total, we successfully introduce 1,288 Izafe tokens annotated as IZAFE.

4.2.4 Oblique case markers

In contrast to the Izafe markers, the oblique case markers only appear within the boundary of the word as a suffix. All possible cases of those markers are explained extensively in subsection 2.1.4, and based on those cases, we can find those markers with ease. In addition, we do not introduce any new POS tag to the UD tagset. We assign the POS tag DET to the oblique case markers.

However, similar to the Izafe case markers, we separate indefinite markers since the oblique case markers are affixed to the end of the nouns after the indefinite markers. Considering the following oblique case in the treebank: *biryarekê distîne* 'takes a decision.' The feminine noun *biryarekê* has both the indefinite marker *-ek* and the oblique case marker *-ê*. Therefore, the noun is split off into three tokens: the noun itself, the indefinite marker, and finally oblique case marker for feminine nouns. We annotated the second and the third tokens as DET.

4.3 The UD Kurmanji revisited treebank

We explained our approach to enhancing the UD Kurmanji original treebank in the previous subsections. In this subsection, we showcase the UD Kurmanji revisited, an enhanced version of the UD Kurmanji treebank. This version has a fine-grained annotation scheme where Northern Kurdish case markers are explicitly present as separate tokens.

```
# sent_id = wiki:wikibank.vislcg.txt:371:10279
# text = Rêveberên mezin têne ber wî û jê aqil digirin.
1   rêveber rêveber NOUN    n   Case=Con|Definite=Def|Gender=Masc
2   ên  ên  IZAFE   izafe
3   mezin   mezin   ADJ adj Degree=Pos  1   amod    _   _
4   têne    hatin   VERB    vblex   Mood=Ind
5   ber ber ADP pr  AdpType=Prep    6   case    _   _
6   wî  ew  PRON    prn Case=Acc|Gender=Masc
7   û   û   CCONJ   cnjcoo  _   10  cc  _   _
8   jê  ji  ADP pr  AdpType=Prep    9   case    _   _
9   aqil    aqil    NOUN    n   Case=Acc|Definite=Def|Gender=Masc
10  digirin girtin  VERB    vblex   Mood=Ind
11  .   .   PUNCT   sent    _   3   punct   _   _
```

Listing 4.3: A sentence from the UD Kurmanji revisited treebank in CoNLL-U format. Some data columns are omitted for better readability.

We revisit the example sentence from listing 4.1 and demonstrate the outcome of our approach on it in listing 4.3. We see how the line with ID 1 is altered, and the last Izafe marker *ên* is removed and added to the next line with ID 2 as a separate token. In addition, the contracted preposition *jê* on line 7 is now represented as a single token instead of three separate tokens in the UD Kurmanji original.

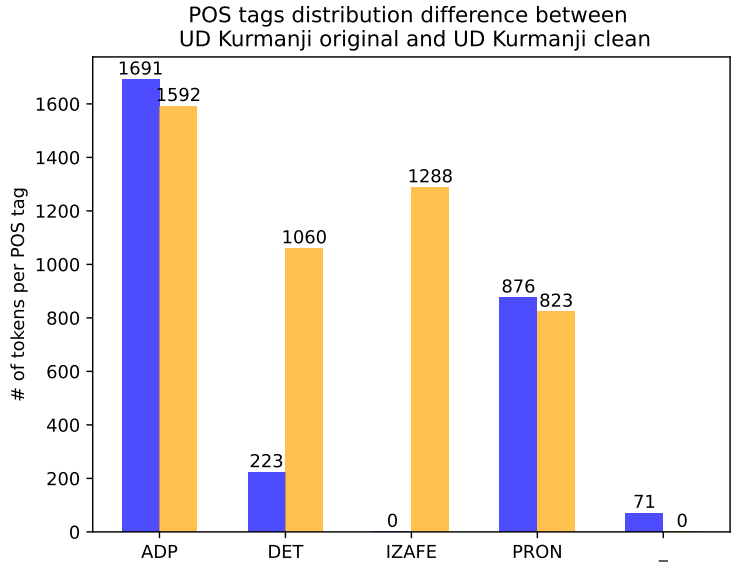


Figure 4.2: Number of tokens per POS tags in UD Kurmanji revisited (orange) in comparison to the original version (blue).

language	language code	# sentences	# tokens	tagset
Northern Kurdish	kmr	754	12,233	18 POS tags

Table 4.1: Statistic of UD Kurmanji revisited .

Table 4.1 demonstrates the statistics of the UD Kurmanji revisited. Our approach has led to the introduction of 1,972 more tokens (19% the number of the tokens in the UD Kurmanji original) along with corresponding POS tags, IZAFE is added, while _ is removed. Figure 4.2 shows the difference between the original and the revisited versions of the treebank. We showcase only the newly added IZAFE tag, the removed (_) tag, and other affected POS tags as a result of our enhancements.

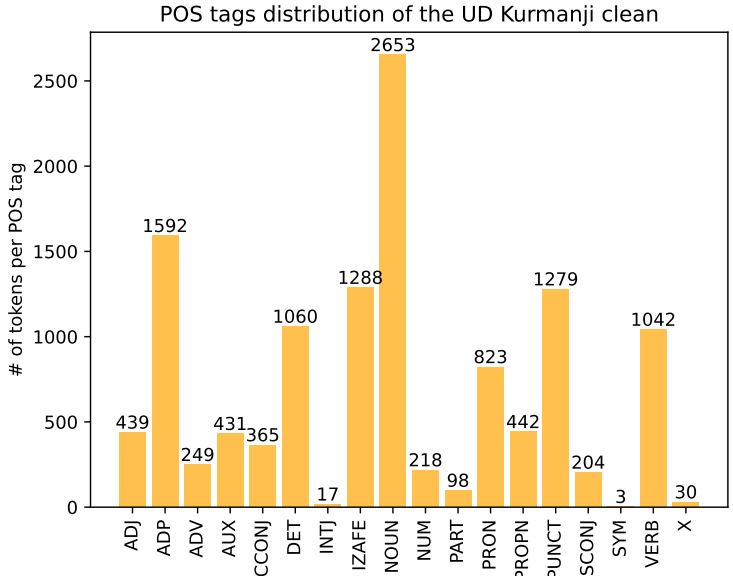


FIGURE 4.3: Number of tokens per POS tags in the UD Kurmanji revisited treebank.

Figure 4.3 depicts the number of tokens per POS tag in the enhanced version of the treebank, the UD Kurmanji revisited. Compared to the figure 4.1, we see the increase of tokens tagged as DET because of the addition of oblique case markers, the absence of the - tag, and the decrease of the ADP since we revisited the second form of Izafe markers.

4.4 Gold-standard dataset

In the previous section, we explained why and how we have enhanced the UD Kurmanji original treebank to overcome its limitations. However, despite some other limitations of the UD Kurmanji treebank, such as being drawn from an old translation in Northern Kurdish and containing many words that are nowadays less frequently used among Northern Kurdish speakers, it is still an invaluable source of data for training machine learning models for different downstream tasks, especially after the data enhancement we performed. Thus, we use the UD Kurmanji treebanks (revisited and original) for training our various POS tagging models.

However, considering the limitations of the UD Kurmanji and the amount of enhancements we had to apply to the treebank, we believe that creating a new annotated dataset where all the linguistic features of Northern Kurdish are carefully addressed in the annotation phase will have a greater impact on the NLP research for Northern Kurdish. In addition, it will encourage

more researchers to follow our path. We call this annotated dataset the gold-standard dataset, and we use it in this study as a test set to evaluate the performance of the various tokenization and POS methods we use in this study.

The gold-standard dataset was created in collaboration with the second supervisor Sina Ahmadi[3]. We collected and annotated 136 (2,937 syntactic words) sentences written in Northern Kurdish from multiple news websites. The first 100 sentences are taken from the unannotated Pewan corpus (Esmaili, Eliassi, et al. 2013), discussed in subsection 3.3.4. The remaining 36 sentences are taken from three Kurdish news websites, mainly Kurdistan24[4], Xwebûn[5], and Hawar News[6]. Table 4.2 demonstrates the statistics of this dataset.

language	language code	# sentences	# tokens	tagset
Northern Kurdish	kmr	136	2,937	15 POS tags

Table 4.2: Statistics of our gold-standard dataset.

We annotate the gold-standard dataset in a fine-grained annotation style, taking into account all case markers, indefinite noun markers, multi-word prepositions and adverbs, and compound verbs. For each given sentence in our gold-standard dataset, we provide: 1) the raw (untokenized) sentence where tokens are delimited by whitespaces and the case markers are not split-off, and 2) a list of tokens with corresponding POS tags where the case markers are explicitly represented as separate tokens. Listing 4.4 presents an example sentence drawn from the dataset. We see that the first line contains the sentence in its raw format (untokenized), followed by a blank line. Then, the tokens of the sentence with the corresponding POS tag (separated by a tab) appear on each line. Lines 9, 11, and 13 represent the Izafe and oblique case markers. The number of tokens does not necessarily correspond to the number of words presented in the sentence in the raw format since the tokens are not only orthographic words but rather syntactic words that can also be Izafe, oblique case, or vocative markers.

Moreover, the availability of the untokenized sentence, along with the list of the tokens, enables us to evaluate various tokenization methods. The untokenized sentence can be fed to any tokenizer, and its output can be compared against the list of tokens we already have, which we consider as gold tokens.

```
1  evroj bi xwe jî, dema dawîya dengdanê bû.
2
3  evroj   ADV
4  bi  ADP
5  xwe PRON
6  jî  ADV
7  ,   PUNCT
8  dem NOUN
9  a   IZAFE
10 dawî    NOUN
```

[3] https://sinaahmadi.github.io/
[4] https://www.kurdistan24.net/kmr
[5] https://xwebun1.org/
[6] https://hawarnews.com/kr/

```
ya   IZAFE
dengdan NOUN
ê    DET
bû   VERB
.    PUNCT
```

Listing 4.4: An example sentence in a raw and tokenized (along with the corresponding POS tags) forms drawn from our gold-standard dataset. 'today was the last day of voting.'

Figure 4.4 shows five examples from the gold-standard dataset in order to showcase how we annotated the dataset in accordance with the linguistic features of Northern Kurdish, ensuring that all markers are represented as syntactic words. Some of those examples are not complete sentences but rather phrases; we provide the untokenized form of the phrase, the tokenized form, and underneath the corresponding POS tags.

Figure 4.4: A collection of annotated sentences and phrases from the gold-standard dataset

Moreover, figure 4.5 illustrates the distribution of POS tags in the gold-standard dataset. This pattern echoes what is seen in both the original and revisited UD Kurmanji treebanks, where the POS tag NOUN has the highest frequency.

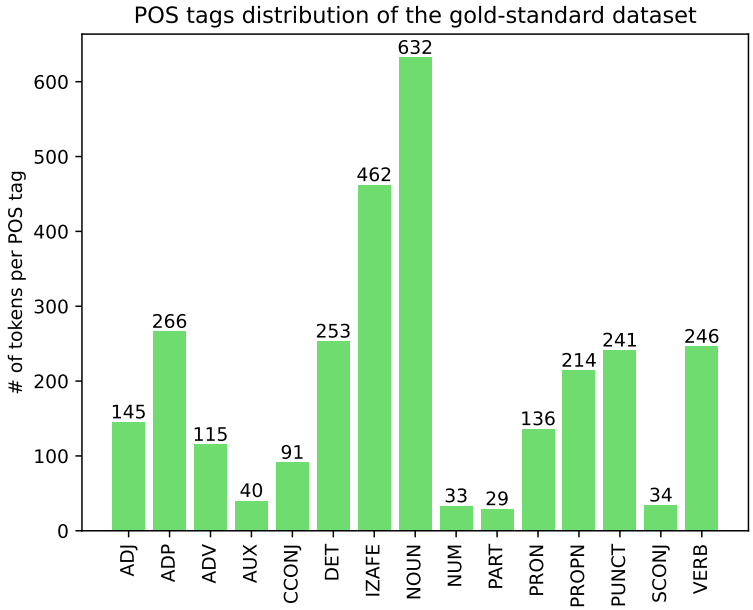


FIGURE 4.5: Number of tokens per POS tags in our gold-standard dataset

4.4.1 Differences between the gold-standard dataset and the UD Kurmanji

Aside from the difference between our gold-standard dataset and the UD Kurmanji with regard to the content and the type of data present, there are also differences in the annotation scheme and how the annotation is done in accordance with the linguistic features of Northern Kurdish. In comparison with the UD Kurmanji revisited tagset, which has 18 POS tags, our gold-standard dataset tagset has only 15 POS tags because the following three POS tags SYM, X, and INTJ are not present in our gold-standard dataset. The tokens belonging to those three POS classes in our dataset are either not present in our dataset or annotated differently.

For example, in the UD Kurmanji *gelek*, 'very' is annotated as X; however, in Northern Kurdish, it must be annotated either as ADJ or ADV, which is the case in our gold-standard dataset. In addition, X is assigned to nouns that are part of the compound verbs; in our case, we tag those nouns either as a NOUN or all together with the verbs they belong to as a multiword expression VERB. For example, in the UD Kurmanji, the compound verb *pêşkêş kirine* 'presented' is split into two tokens: *pêşkêş* and *kirine* and tagged X and VERB, respectively.

Multi-word expressions like compound verbs, compound prepositions, and compound adverbs are also handled differently. For example, the compound verb *dest pê kirin* 'start/begin' is separated into three tokens in the UD Kurmanji, *dest*, *pê*, and *kirin* and tagged as X, X and NOUN, respectively. In our gold-standard dataset, we tag it as VERB in case they appear after each other. In cases where other tokens appear between them, we separate them into three tokens and tag them as NOUN, ADP, and VERB, just like the following phrase *dest bi çinîna ceh û nîskan kirin* 'they started harvesting barely and lentil'; we tag *dest* as NOUN, *bi* as ADP, and *kirin* as VERB.

Regarding compound prepositions, we annotate the compound preposition *li ser* 'on/upon'

as ADP, while in UD Kurmanji, it is separated into two tokens *li* 'in/at' and *ser* 'onto' where both are tagged as ADP. In addition, compound adverbs such as *Bi tenê* 'only' are also separated into two tokens *bi* 'with' and *tenê* 'alone', both are annotated as ADP. However, in our gold-standard dataset, we treat it as one multi-word token, and we annotate it as ADV.

Moreover, the verb to be in Northern Kurdish *bûn* 'to be' is always annotated as AUX in the UD Kurmanji treebank, while we tag it as a VERB unless it appears in compound form. In addition, the particles *-ê* and *dê* are used for forming the future tense in Northern Kurdish and are tagged as AUX in UD Kurmanji. However, we tag those particles as PART because they are not auxiliary verbs. In addition, the tokens *jî* 'also/too' and *her* 'every' are annotated as PART and either as DET or ADV in the UD Kurmanji, respectively. We annotate the former as ADV and the latter as PRON.

4.4.2 Annotation procedure

In general, the manual annotation procedure of text is a challenging task since it is time-consuming and requires linguistic expertise and a clear understanding of the context where the words appear. In the case of the annotation procedure of our gold-standard dataset, there were extra challenges present. For example, despite the fact that Northern Kurdish is mainly written in the Latin alphabet, it still suffers from orthographic variations where a single word can be written differently due to dialectal and regional differences or historical spelling conventions.

For example, in Northern Kurdish, the nouns ending with the vowel *î* in the Izafe and oblique cases are affixed by a *y* + *Izafe or oblique case markers*. In some regions, the *î* is replaced by an *i*, while in other regions it remains untouched. Thus, we considered both forms to be correct; for example, both variants *şanogeriya kurdî* and *şanogerîya kurdî* 'Kurdish theatre' are considered correct.

In the case of annotation disagreement, regardless of the reasons, we either consulted linguistic books like (Tan 2015; Weqfa 2019), websites like Zimannas[7] or Wîkîferheng[8], or asked for help from the Kurdish linguistic community on X (formerly Twitter)[9].

4.5 Northern Kurdish POS tagging pipeline

In the previous sections, we have presented the UD Kurmanji original treebank and how we revisited it, leading to the UD Kurmanji revisited treebank. In addition, we introduced our gold-standard dataset and our annotation procedure. In this section, we explicitly delineate our approach for constructing and evaluating POS taggers for Northern Kurdish.
As depicted in figure 4.6, our proposed supervised POS tagging pipeline is composed of four key elements: training data, testing data, tokenization methods, and a set of seven POS tagging models. We highlight those elements in the following subsection.

[7] https://zimannas.wordpress.com/
[8] https://ku.m.wiktionary.org/
[9] https://twitter.com/

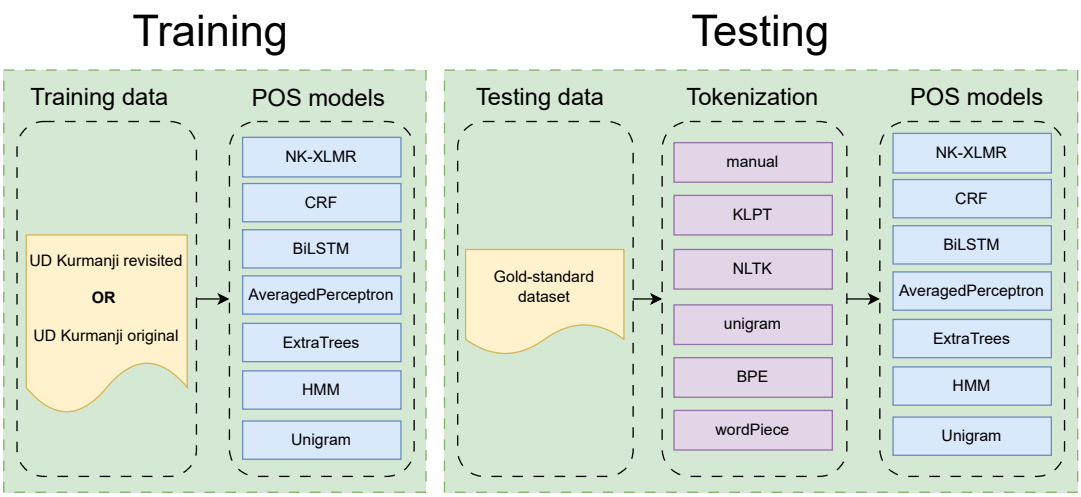


FIGURE 4.6: Our POS tagging pipeline for Northern Kurdish.

4.5.1 Training data

In this study, we train all POS tagging models independently, once on the UD Kurmanji original and once on the UD Kurmanji revisited. We take this approach because we want to assess the impact of the explicit presence of Northern Kurdish linguistic features (case markers) in training data on the performance of our pipeline.

4.5.2 Testing data

We use our fine-grained POS manually annotated **gold-standard dataset** outlined in section 4.4 as testing data to evaluate the performance of the tokenization methods and the POS tagging models in our pipeline.

4.5.3 Tokenization methods

In a POS tagging pipeline, tokenization is essential for preparing raw (untokenized) text for either training or inference purposes. Error in the tokenization stage can significantly impact the performance of the POS tagger. The authors of (Rust et al. 2021) conducted a comprehensive empirical investigation with regard to the effect of the tokenization method on the performance of monolingual and multilingual large language models. They concluded that there is a performance discrepancy between using language-specific tokenization methods instead of using general and language-agnostic tokenization methods. Their conclusion is confirmed by the results we obtain in this study.

While tokenization for segmented languages, such as English, is relatively well-established and straightforward, the task becomes considerably more challenging for agglutinative and morphologically-rich languages like Kurdish (Mitkov 2022). Just as with POS tagging and other NLP tasks for Northern Kurdish, tokenization is an understudied area. As of the writing

of this study, the work of (Ahmadi 2020b) represents the only dedicated work on tokenization for the Kurdish language. In addition to the KLPT tokenizer, the author provided multiple neural tokenization models trained (unsupervised) on Northern Kurdish raw corpora. We use three of those models: Unigram, BPE, and wordPiece. To summarize, in this study, we use six tokenization methods:

1. manual tokenization
2. KLPT tokenizer
3. NLTK tokenizer
4. unigram tokenizer
5. BPE tokenizer
6. wordPiece tokenizer

In order to demonstrate the performance difference of those models on untokenized text, we present in table 4.3 the outputs of various tokenization methods and their token counts for a sample sentence taken from our gold-standard dataset. We use the hyphen symbol - to indicate the places where the tokenizer is failing to produce tokens identical to the tokens in manual tokenization (ground truth). Although the KLPT tokenizer fails to produce perfectly tokenized sentence, its performance is very close to manual tokenization. This is due to the knowledge the tokenizer has of Northern Kurdish and its various linguistic features, such as the Izafe and the oblique case markers. On the other hand, the wordPiece tokenizer performs the worst, and it even removes tokens from the input text.

Sentence	Keç û xortên Êzidî di meha Nîsanê de zewacê nakin.															# tokens
manual	Keç	û	xort	ên	Êzidî	di	meh	a	Nîsan	ê	de	zewac	ê	nakin	.	15
KLPT	Keç	û	xort	ên	Êzidî	di	meh	a	Nîsanê	-	de	zewac	ê	nakin	.	14
NTLK	Keç	û	xortên	-	Êzidî	di	meha	-	Nîsanê	-	de	zewacê	-	nakin	.	11
unigram	Keç	û	xort	ên	Êzidî	di	meha	-	Nîsanê	-	de	zewacê	-	nakin	.	12
BPE	Keç	û	xortên	-	Êzidî	di	meha	-	Nîsanê	-	de	zewacê	-	nakin	.	11
wordPiece	-	û	xortên	-	-	di	meha	-	-	-	de	zewacê	-	nakin	.	8

TABLE 4.3: The output of the different tokenization methods for a sentence drawn from our gold-standard dataset.

The column # *tokens* in table 4.3 represents the tokens count of each method; the differences in the token count are a problem. In order to be able to perform evaluation for POS models, we must have the same number of POS tags in the ground truth and the predicted ones. Thus, we must have an identical token count even in cases where the tokenizer fails to produce correct tokens. Therefore, we introduce an intermediary preprocessing step between the tokenization and POS tagging stages within our pipeline. This step entails comparing the list of tokens (ground truth) from the manual tokenization against the list of tokens generated by other tokenization methods. When a token mismatch arises in case of under-tokenization, we insert a placeholder token denoted as **None**. Conversely, in cases of over-tokenization, we eliminate

the extra tokens, giving precedence to those that overlap with the gold tokens. This ensures that the token count remains consistent, fulfilling the prerequisites for the task of POS tagging. Table 4.4 shows our approach, the row *Processed* is the final output of the tokenization, which is then fed to the POS tagger.

Sentence	Keç û xortên Êzidî di meha Nîsanê de zewacê nakin.														
manual	Keç	û	xort	ên	Êzidî	di	meh	a	Nîsan	ê	de	zewac	ê	nakin	.
KLPT	Keç	û	xort	ên	Êzidî	di	meh	a	Nîsanê	-	de	zewac	ê	nakin	.
Processed	Keç	û	xort	ên	Êzidî	di	meh	a	None	None	de	zewac	ê	nakin	.
wordPiece	-	û	xortên	-	-	di	meha	-	-	-	de	zewacê	-	nakin	.
Processed	None	û	None	None	None	di	None	None	None	None	de	None	None	nakin	.

TABLE 4.4: The output of KLPT and WordPiece tokenizers before and after applying our approach.

To gain deeper insight into the performance of the tokenization methods on the gold-standard dataset, we detail the differences in figure 6.1, figure A.1 and figure A.2. The numbers in the figures denote the total number of sentences that are either over-, under-, or perfectly-tokenized upon applying tokenization methods.

4.5.4 POS tagging models

Based on our literature review in chapter 3, we propose a set of seven POS tagging models for our study:

1. Unigram model (baseline model)
2. HMM model
3. ExtraTrees model
4. AveragedPerceptron model
5. BiLSTM model
6. CRF model
7. Transformer-based (XLM-RoBERTa) model

In order to be able to compare the performance of proposed POS tagging models and indicate improvements throughout the study, it is very important to establish a baseline model. Therefore, we choose the Unigram model to be our baseline in this study. A Unigram model serves as an easily interpretable and computationally efficient starting point, offering a transparent lower bound for performance metrics. Its simplicity enables quick benchmarking and lends itself well to model comparisons, thereby ensuring that any advancements achieved by more sophisticated models are both meaningful and quantifiable. The experimental procedures employed for the baseline and POS tagging models are discussed in chapter 5.

Chapter 5

Experimental Setup

In the previous chapter, we demonstrated the training and the testing data and the differences between them. In addition, we showed the differences between the various tokenization methods we use and summarized our proposed POS tagging pipeline in figure 4.6. In this chapter, we explain the technical details of the various POS tagging models in our pipeline for Northern Kurdish.

As a general approach for all POS models, with the exception of the Unigram model, we perform hyperparameters optimization using the Optuna framework (Akiba et al. 2019). The values of the hyperparameters for each model are listed in appendix A.3. In addition, with the exception of the XLM-RoBERTa-based model, we perform a ten-fold cross-validation to gain a reliable estimate of the POS models' performance in the training stage.

5.1 Baseline model (Unigram model)

We utilize the Unigram model from the NLTK Python package (Bird, Klein, and Loper 2009) for POS tagging. This model assigns tags based on word frequency observed during training. It uses conditional frequency distributions to calculate the most likely tag for each given token. In linguistically resource-limited settings like ours, the model may encounter unfamiliar words (*out-of-vocabulary*). To address this limitation, we specify the default POS tag to be NOUN when it fails to determine a POS tag for a token. This is a common practice when establishing a baseline, and it is motivated by (Bird, Klein, and Loper 2009).

5.2 HMM model

Similar to the Unigram model, we employ an HMM model for POS tagging, leveraging the implementation available in the NLTK Python package, which is grounded in the research by (X. Huang et al. 2001).

5.3 ExtraTrees model

For this model, we use the **scikit-learn**[1] (Pedregosa et al. 2011) Python library for implementing an ExtraTrees model. In contrast to other models in this study, the model expects the training data to be vectorized. Therefore, the training tokens and tags must be converted into features. Consequently, those features must be converted to a sparse matrix. The set of features we extract from the input tokens and tags is listed in appendix A.3.3.

5.4 AveragedPerceptron model

Similar to the implementation of the ExtraTrees, the Averaged Perceptron POS tagging model has the notion of feature engineering. However, we do not define our own set of features. We use the standard features set defined by the NLTK Python package since we use their implementation[2].

5.5 BiLSTM model

In this study, we use the Flair Python package (Akbik, Bergmann, et al. 2019) to build our sequence tagging (POS tagging) model, which leverages a configurable BiLSTM architecture as originally proposed by (Z. Huang, Xu, and Yu 2015). In addition, we use pre-trained sub-word fastText embeddings (Grave et al. 2018) specifically pre-trained on Northern Kurdish data. The fastText embeddings are 300-dimensional trained on Common Crawl[3] and Wikipedia[4] data across 157 languages. Unlike conventional static word embeddings such as Word2Vec (Mikolov et al. 2013) or Polyglot (Al-Rfou, Perozzi, and Skiena 2013), fastText generates embeddings from character-level n-grams, thereby being better at capturing morphological nuances.
It is worth mentioning that Flair needs to have a development set alongside the primary training data. Therefore, we partition our training dataset, allocating 90% for training purposes and reserving the remaining 10% as a development set.

5.6 CRF model

Like the Unigram, HMM, and AveragedPerceptron models, we utilize the CRF implementation available in the NLTK Python package, which serves as a wrapper for the CRFsuite library (Okazaki 2007).

[1] https://scikit-learn.org/stable/index.html
[2] The NLTK implementation is based on Matthew Honnibal's implementation https://explosion.ai/blog/part-of-speech-pos-tagger-in-python
[3] http://commoncrawl.org/
[4] https://www.wikipedia.org/

5.7 Fine-tuned XLM-RoBERTa model (NK-XLMR)

In contrast to the previous models, where each model was trained from scratch for our task, we fine-tune the pretrained multilingual XLM-RoBERTa model (Conneau, Khandelwal, et al. 2020) on the UD Kurmanji treebanks. We utilize the 'base' version of XLM-RoBERTa because of its lower computational requirements, making it more amenable to fine-tuning.
This procedure is done using Trankit, a Python package introduced by (Nguyen et al. 2021), known for its pre-training on 90 UD treebanks, including UD Kurmanji. Trankit offers a relatively fast and straightforward approach for fine-tuning LLMs like XLM-RoBERTa, thanks to the utilization of Adapters (Pfeiffer, Rücklé, et al. 2020). Similar to Flair, Trankit requires having a development set while performing the fine-tuning task. Therefore, we partition the training data, allocating 90% for training purposes and reserving the remaining 10% as a development set. We refer to the fine-tuned POS model as **N**orthern **K**urdish **XLM**-**R**oBERTa **(NK-XLMR)**.

Chapter 6

Results

In chapter 5, we delved into the technical aspects of the various POS tagging models we have used in this study. In this chapter, we aim to shed light on the performance of the various tokenization methods and the POS models we have used. Specifically, we present a comprehensive evaluation, which includes metrics such as accuracy and F1 scores for the POS model and precision-based metric (BLEU score) for tokenization methods.

In section 6.1, we present the result of the intrinsic evaluation of the tokenization methods we used in this research, quantified via BLEU scores. Section 6.2 presents the evaluation metrics for our proposed POS models, comparing all the models to the baseline model. As we have previously mentioned, the proposed POS models are trained twice, once on the UD Kurmanji original and once on the UD Kurmanji revisited. Therefore, *original* and *revisited* are added to the names of the POS models to denote the version of the training data. Finally, in section 6.3, we present a qualitative analysis for two example sentences annotated by our POS models, combined with an error analysis for the POS models.

POS tagging for Northern Kurdish is far from a straightforward endeavor. It is a task packed with challenges that span data scarcity and quality, unresolved issues in tokenization, and uncertainties in model selection, all of which our research tried to address comprehensively. Therefore, before we present our results and findings, we find it important to outline once again the non-trivial challenges we encountered while carrying out this study, POS tagging for Northern Kurdish. Then, we provide our methodological solutions to overcome those challenges.

1. **Data scarcity and data quality:** One of the foremost challenges is the lack of annotated data for training and testing. In addition, the available annotated dataset for training purposes, UD Kurmanji original, exhibits coarse-grained annotation scheme and several inconsistencies, posing challenges for effective POS tagging for Northern Kurdish.

2. **Tokenization challenge:** Tokenization for Northern Kurdish remains an area largely untouched by prior research, with the exception of the KLPT tokenizer (Ahmadi 2020b).

3. **Model selection challenge**: Choosing the most effective learning paradigm further complicates the task. The spectrum ranges from statistical traditional machine learning models like HMMs and CRFs to more robust approaches like BiLSTMs and transformer-based architectures. The question persists: which offers the best trade-off between computational complexity, accuracy, and data scarcity?

In order to alleviate those challenges, we adopted the following multifaceted approach:

1. **Experimented with two training datasets:** Considering the data quality challenge, we revisited and enhanced the UD Kurmanji original by addressing the case markers explicitly in the treebank and creating a new version of the treebank; we call this version the UD Kurmanji revisited. In addition, we merged the test and train splits of the UD Kurmanji treebanks to be used for training the POS tagging models in our pipeline.

2. **Introduced a gold-standard dataset:** Recognizing the data scarcity for Northern Kurdish and the quality of available data, we collected 136 sentences from multiple news websites and manually annotated them in a fine-grained manner, ensuring the presence of linguistic features in a usable and beneficial way for our task. We called this dataset a gold-standard dataset and used it as a test set for evaluating tokenization methods and POS tagging models in our pipeline.

3. **Explored various tokenization strategies:** The tokenization challenge was further amplified because of the fine-grained annotation style of UD Kurmanji revisited and the gold-standard dataset. Therefore, we explored various tokenization methods with the aim of pinpointing an optimal strategy tailored for the linguistic characteristics of Northern Kurdish. We incorporated the manually tokenized sentences as a ground truth for tokenization, thereby underscoring the performance disparities between existing methods.

4. **Created multiple POS tagging models:** In order to explore what POS tagging model can achieve the best performance for our task, we proposed seven POS tagging models. Those models are trained independently on UD Kurmanji original and UD Kurmanji revisited in order to demonstrate the impact of the presence of linguistic features on the task of POS tagging for Northern Kurdish. We evaluate the POS models on the manually annotated gold-standard dataset.

6.1 Tokenization methods evaluation results

In subsection 2.6.1, we explained how tokenization methods could be evaluated. In this section, we present the intrinsic evaluation of the tokenization methods. However, the extrinsic evaluation of those methods is presented together with the evaluation of the POS models in section 6.2.

Table 6.1 shows the BLEU scores of the tokenization methods we used in this study using the gold-standard dataset as testing data. We see that the BLEU scores for the KLPT tokenizer are the closest to the manual tokenization (ground truth), outperforming other tokenizers by a great margin. In contrast to other tokenizers, the KLPT tokenizer is characterized by its extensive knowledge of Northern Kurdish, enabling it to correctly recognize case markers and handle multi-word expressions like compound verbs and compound prepositions.

Tokenization method	BLEU-score (gold-standard dataset)			
	BLEU-1	**BLEU-2**	**BLEU-3**	**BLEU-4**
manual	*1.00*	*1.00*	*1.00*	*1.00*
KLPT	0.73	0.65	0.59	0.53
unigram	0.54	0.44	0.36	0.29
NLTK	0.50	0.41	0.33	0.25
BPE	0.50	0.39	0.31	0.24
wordPiece	0.45	0.36	0.28	0.21

TABLE 6.1: BLEU scores for all tokenization methods on the gold-standard dataset.

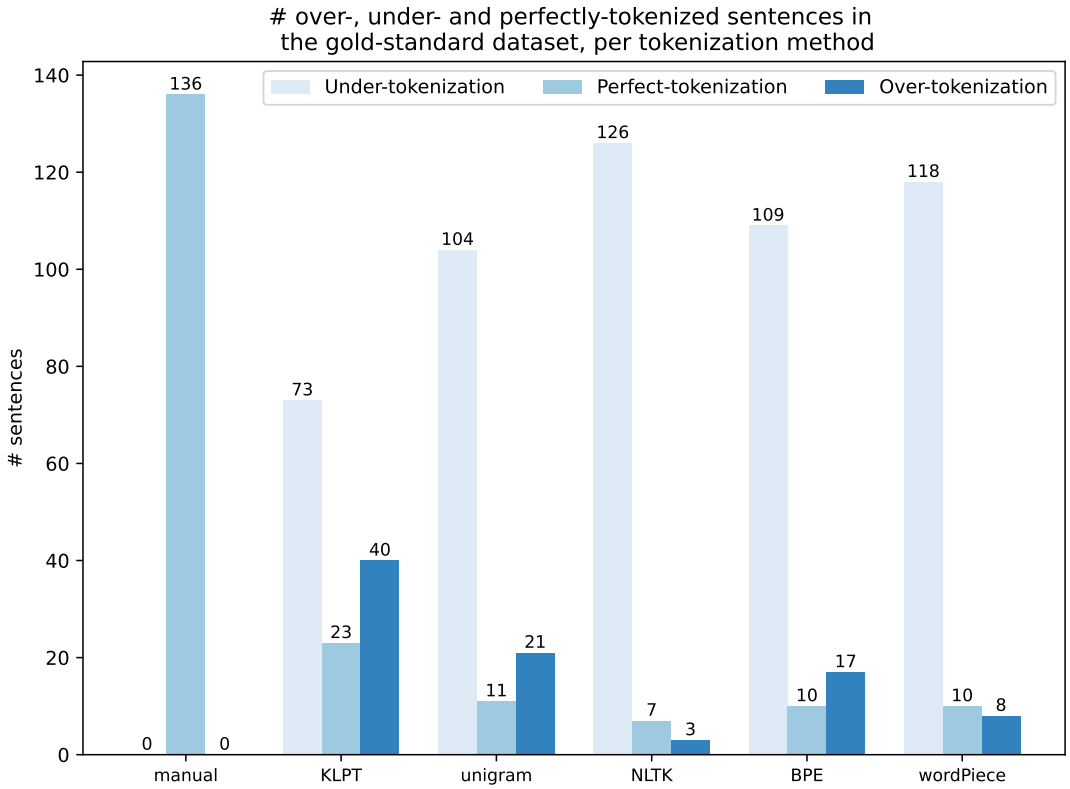


FIGURE 6.1: Number of over-,under-, and perfectly-tokenized sentences in the gold-standard dataset, per tokenization method.

Moreover, figure 6.1 further confirms the performance of the KLPT tokenizer. We observe that the KLPT perfectly tokenized 23 sentences out of the 136 sentences in the gold-standard dataset. In addition, a persistent pattern among the tokenization methods, with the exception of the manual, is that their outputs are under-tokenized. The NLTK has the highest number of

under-tokenized and the lowest number of perfectly-tokenized sentences due to the fact that it is optimized for languages where the words are mainly delimited by white spaces, like English.

Furthermore, we see that the results of the unigram, BPE, and wordPiece tokenization methods regarding over-tokenized sentences are very low despite being sub-word neural models trained on raw Northern Kurdish data by (Ahmadi 2020a). We argue that this is due to the fact the data in Northern Kurdish used for training those models is not tokenized; therefore, they fail to perform the tokenization correctly, taking into account the case markers.

6.2 POS tagging models evaluation results

In this section, we present the evaluation results of our proposed POS models on the gold-standard dataset. We report the macro-averaged F1 score and the accuracy per POS model. In addition, the results we present are two-dimensional because the results of each POS model are also presented with regard to the tokenization method we have used in the testing stage. The inclusion of the tokenization aspect in our results represents the extrinsic evaluation of the various tokenization methods and their impact on the performance of the pipeline as a whole.

Subsection 6.2.1 and subsection 6.2.2 present the evaluation for the baseline and best-performing models. In subsection 6.2.3, we present the evaluation results of the POS models all together in tabular and graphical forms.

6.2.1 Baseline model

Table 6.2 presents the evaluation results (accuracy and macro-averaged F1 score) of our baseline model evaluated on the gold-standard dataset. We see that the highest scores are achieved within the manual tokenization as it serves as the ground truth for the tokenization. The decrease in performance when moving away from manual tokenization is expected and is in line with the same decreasing trend of BLEU scores in table 6.1.

POS model	Tokenization methods											
	manual		**KLPT**		**unigram**		**NLTK**		**BPE**		**wordPiece**	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
Baseline(revisited)	**0.59**	**0.73**	**0.47**	**0.52**	**0.41**	**0.37**	**0.4**	**0.32**	**0.4**	**0.33**	**0.37**	**0.33**
Baseline(original)	0.4	0.51	0.35	0.41	0.36	0.34	0.37	0.32	0.36	0.32	0.34	0.31

TABLE 6.2: The macro-averaged F1 scores and accuracy (Acc) of our baseline model evaluated on the gold-standard dataset. *(revisited)* and *(original)* denote the variation of training data.

6.2.2 NK-XLMR model

Table 6.3 demonstrates the evaluation results (accuracy and macro-averaged F1 score) of our best-performing model, NK-XLMR, evaluated on the gold-standard dataset. By comparing the results presented in table 6.2 and table 6.3, it becomes evident that the employment of the

POS model	Tokenization methods											
	manual		**KLPT**		**unigram**		**NLTK**		**BPE**		**wordPiece**	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
NK-XLMR(revisited)	**0.77**	**0.87**	**0.56**	**0.59**	**0.49**	**0.43**	**0.51**	**0.39**	**0.47**	**0.39**	**0.44**	**0.36**
NK-XLMR(original)	0.57	0.62	0.46	0.47	0.45	0.39	0.47	0.38	0.45	0.37	0.40	0.35

TABLE 6.3: The macro-averaged F1 scores and accuracy (Acc) of our NK-XLMR model evaluated on the gold-standard dataset. *(revisited)* and *(original)* denote the variation of training data.

NK-XLMR model leads to a notable improvement in the task of POS tagging for Northern Kurdish. Specifically, the accuracy improves by 0.14 and 0.11, while the macro-averaged F1 score experiences an increase of 0.18 and 0.17, contingent upon the respective training data.

Similar to the results presented in table 6.2, we observe an apparent decline in performance as we transition away from manual tokenization. The KLPT method ranks second in efficacy among tokenization methods. However, it still falls significantly short of manual tokenization. Taking the results from table 6.3 for the revisited variant and comparing the manual with KLPT, we notice a performance decrease of 0.28 for the accuracy and 0.19 for the F1 score.

6.2.3 Comprehensive evaluation across all models

In this subsection, we present the evaluation results (accuracy and macro-averaged F1 score) of all POS tagging models in our pipeline. In order to make the comparison clearer, we divide the results based on the used training data (revisited and original). While table 6.4 provides a detailed comparison of all models trained on the UD Kurmanji revisited, table 6.5 demonstrates the results of the same POS model but trained on UD Kurmanji original.

In addition, we present the same results in a graphical format. Figure 6.2 offers a visual representation of the evaluation metrics, specifically focusing on accuracy across various training data and tokenization methods. Figure A.3 is dedicated to demonstrate the macro averaged F1 scores. In both figures, solid bars depict models trained on the UD Kurmanji revisited, whereas bars with hatch patterns illustrate models trained on the UD Kurmanji original.

POS model	Tokenization methods											
	manual		**KLPT**		**unigram**		**NLTK**		**BPE**		**wordPiece**	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
Baseline (Unigram)	0.59	0.73	0.47	0.52	0.41	0.37	0.4	0.32	0.4	0.33	0.37	0.33
HMM	0.62	0.77	0.48	0.53	0.4	0.37	0.41	0.33	0.4	0.34	0.38	0.33
ExtraTrees	0.61	0.79	0.49	0.56	0.43	0.4	0.41	0.36	0.41	0.36	0.37	0.34
AveragedPerceptron	0.68	0.83	**0.57**	0.57	0.47	0.41	0.49	0.37	0.45	0.37	0.40	0.35
BiLSTM	0.72	0.83	0.52	0.57	0.45	0.40	0.43	0.36	0.43	0.37	0.41	0.34
CRF	0.74	0.84	0.55	0.59	0.48	0.42	0.48	0.39	0.46	0.38	0.42	0.35
NK-XLMR	**0.77**	**0.87**	0.56	**0.59**	**0.49**	**0.43**	**0.51**	**0.39**	**0.47**	**0.39**	**0.44**	**0.36**

TABLE 6.4: The macro-averaged F1 scores and accuracy (Acc) of the POS tagging models, trained on the UD Kurmanji revisited and evaluated on our gold-standard dataset.

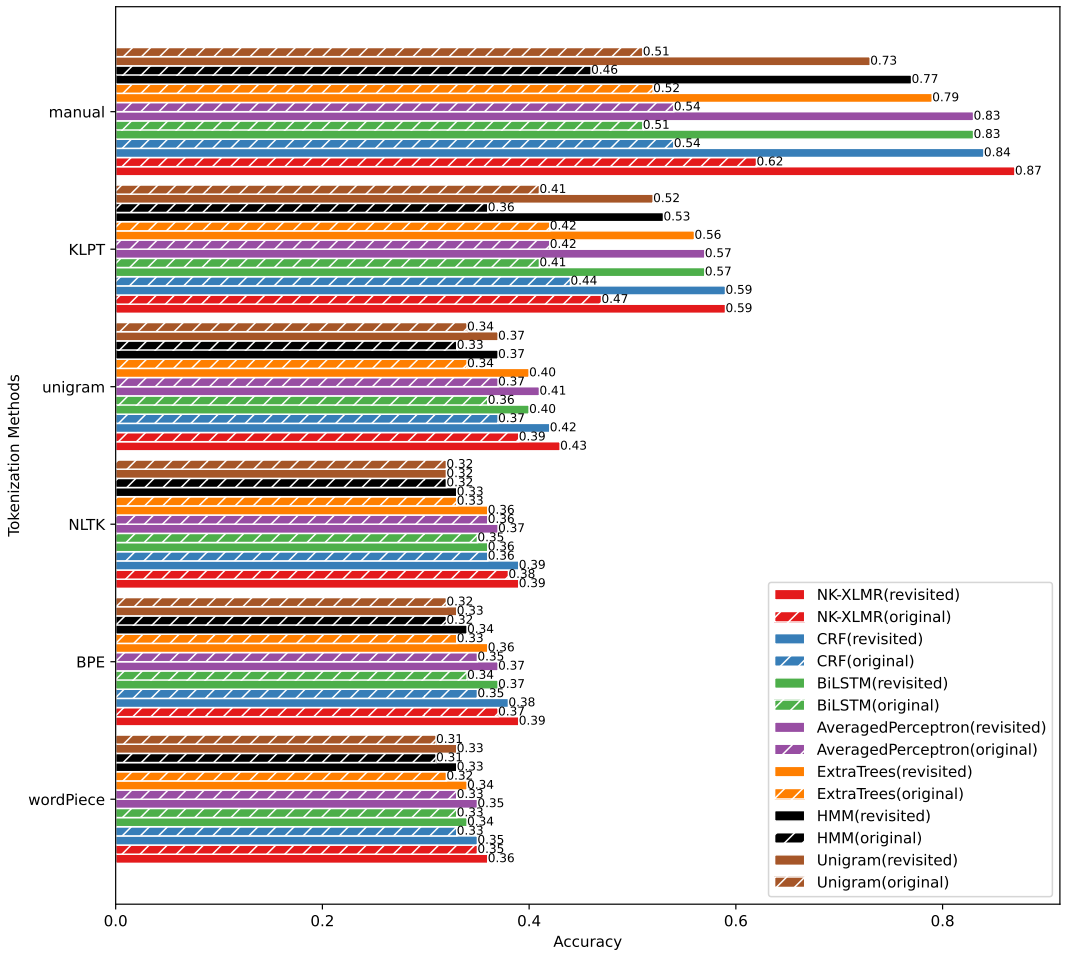


FIGURE 6.2: The accuracy of the proposed POS models in relation to the training data and tokenization methods.

Further observation reveals that within the context of the training on UD Kurmanji revisited, both the BiLSTM and AveragedPerceptron models exhibit identical accuracy scores, although their macro-averaged F1 scores diverge slightly but remain comparable. Conversely, when utilizing the UD Kurmanji original, a similar trend of identical accuracy emerges between the AveragedPerceptron and the CRF models. Additionally, it is notable that the HMM model lags behind, even when compared to the baseline.

By Comparing the results in both tables and regardless of the tokenization method, we observe a performance increase among the models. This increase is the highest within the manual tokenization method and the lowest within the wordPiece tokenization method. This confirms the importance and the impact of the data refinement we did on the UD Kurmanji original treebank for the task of POS tagging. In addition, it stipulates the impact the performance of the tokenization method has on POS tagging for Northern Kurdish.
While this performance increase is in part due to the different annotation scheme, which is explained in section 4.3, the introduction of this richer scheme improved the performance of the POS models on specific POS tags other than IZAFE and DET. A detailed analysis of this improvement is reported in section 6.3.

POS model	Tokenization methods											
	manual		**KLPT**		**NLTK**		**unigram**		**BPE**		**wordPiece**	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
Baseline (Unigram)	0.4	0.51	0.35	0.41	0.37	0.32	0.36	0.34	0.36	0.32	0.34	0.31
HMM	0.37	0.46	0.33	0.36	0.35	0.32	0.34	0.33	0.34	0.32	0.34	0.31
ExtraTrees	0.41	0.52	0.37	0.42	0.38	0.33	0.37	0.34	0.38	0.33	0.34	0.32
AveragedPerceptron	0.44	0.54	0.37	0.42	0.40	0.36	0.38	0.37	0.39	0.35	0.36	0.33
BiLSTM	0.42	0.51	0.40	0.41	0.45	0.35	0.43	0.36	0.44	0.34	**0.42**	0.33
CRF	0.46	0.54	0.41	0.44	0.42	0.36	0.40	0.37	0.40	0.35	0.35	0.33
NK-XLMR	**0.57**	**0.62**	**0.46**	**0.47**	**0.47**	**0.38**	**0.45**	**0.39**	**0.45**	**0.37**	0.40	**0.35**

TABLE 6.5: The macro-averaged F1 scores and accuracy (Acc) of the POS tagging models trained on the UD Kurmanji original and evaluated on our gold-standard dataset.

Moreover, the NK-XLMR POS model is our best model as it outperforms all other models. This was an expected performance, and it is in line with our finding in the chapter 3 where we showed how the transformer-based LLMs achieve state-of-the-art results for multiple NLP tasks, including POS tagging for both HRLs and LRLs. However, comparing the scores of NK-XLMR with the CRF model in table 6.4, we observe very close performance between the two. The difference is very small, 0.03 for the macro-averaged F1 and the accuracy scores. This is a notable result, especially with regard to the computational resources required for fine-tuning XLM-RoBERTa and for training the CRF model from scratch for the task of POS tagging. Based on our experiments in this study, fine-tuning XLM-RoBERTa for POS tagging took notably longer than training the CRF for the same task.

6.3 POS models error analysis

In figure 6.3 and figure 6.4, we provide the outputs of the CRF(revisited) and NK-XLMR(revisited) performing POS tagging on two sentences drawn from the gold-standard dataset. We provide the sentences in untokenized and tokenized (gold) formats in addition to the gold POS tokens. Only the gold tokens (manual tokenization method) are used for both sentences. The outputs of all POS models for both sentences are provided in figure A.5 and figure A.6.

We see how the NK-XLMR(revisited) in figure 6.3 correctly performs the task of POS tagging for the given sentence, while CRF(revisited) incorrectly tags the auxiliary verb *bide* 'give' as an adjective and the oblique case marker *ê* as an Izafe case marker. However, in figure 6.4, we notice that both NK-XLMR(revisited) and CRF(revisited) incorrectly tag the pronoun *evî* 'this', the adverbs *jî* 'too' and *niha* 'now', and the copular verb *in* 'be'. We argue that this is due to the fact that those tokens are either annotated differently in the UD Kurmanji as described in subsection 4.4.1, or they have not been seen during the training, in the case of the pronoun *evî* and the adverb *niha*. Additionally, the CRF(revisited) fails to correctly tag the superlative adjective *mezintirîn* 'the largest' and the preposition *bê* 'without', while NK-XLMR(revisited) tags both of them correctly.

In addition to the two sample outputs, the presented results in the table 6.4, table 6.5, and figure 6.2 unambiguously demonstrate two trends in our results. First, training the POS model on

FIGURE 6.3: Outputs of the CRF and NK-XLMR POS taggers compared to the gold-standard annotation for a sentence from the gold-standard dataset 'Leyla Qasim wanted to make the Kurdish voice heard in the world.'

FIGURE 6.4: Outputs of the CRF and NK-XLMR POS taggers compared to the gold-standard annotation for a sentence from the gold-standard dataset 'thus, they are the largest stateless nation now.'

the UD Kurmanji revisited undeniably results in higher accuracy and F1 scores when compared with outcomes of POS models trained on the UD Kurmanji original. Second, the performance of POS models tends to decline as we transition away from the manual tokenization method. The further we move, the less knowledge of the linguistic characteristics of Northern Kurdish the tokenizers have.
Therefore, it would be helpful if we performed an error analysis based on those two trends, taking the NK-XLMR as an example.

We present two confusion matrices figure 6.5 and figure A.4 demonstrating the performance of the NK-XLMR(revisited) and NK-XLMR(original). The UD Kurmanji revisited is characterized by the enhancements we have introduced that are discussed in detail in section 4.2. Our refinements affected tokens from the following POS tags: NOUN, PROPN, DET, and ADP, which are important elements in the Izafe and oblique cases in Northern Kurdish. In addition, we introduced a new POS tag, IZAFE.

By comparing the confusion matrices, we observe that NOUN and PROPN benefit the most

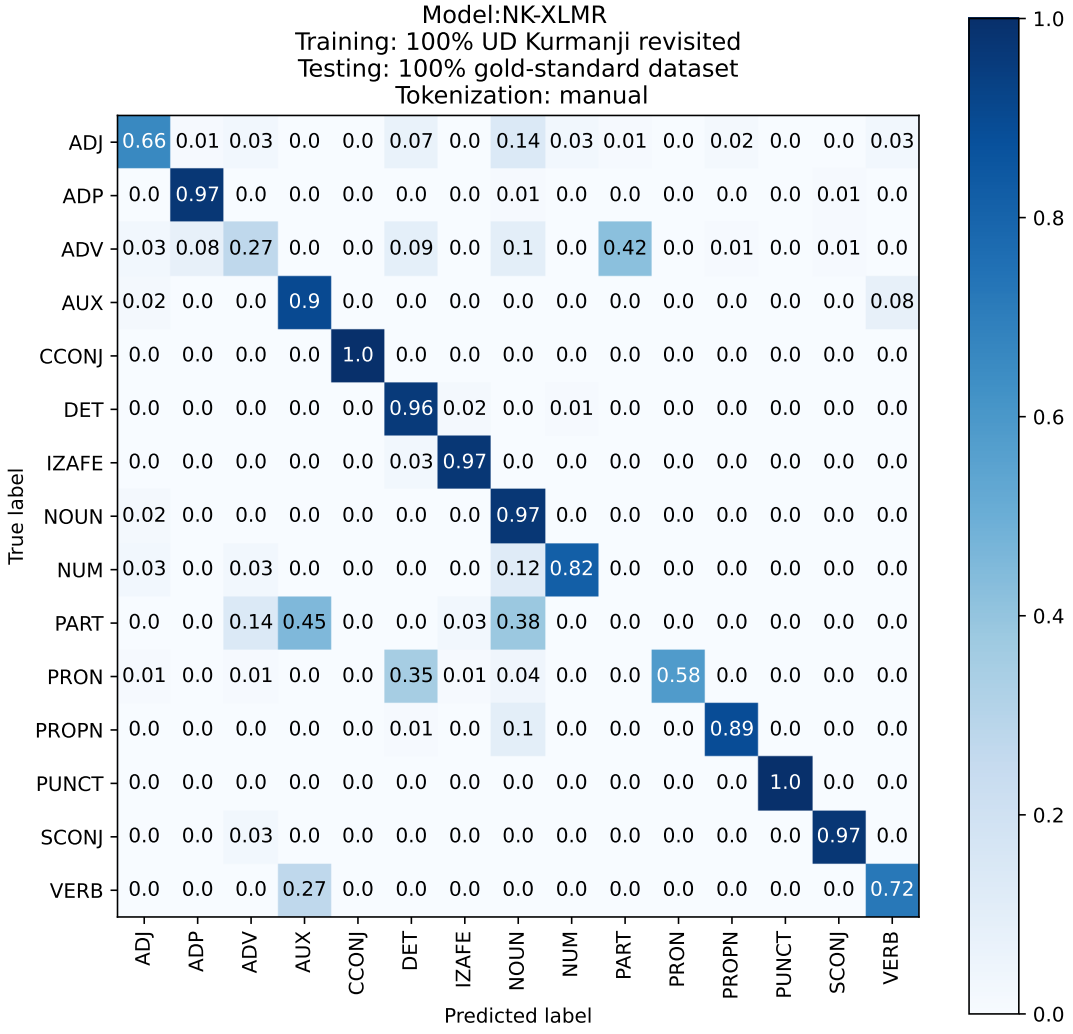
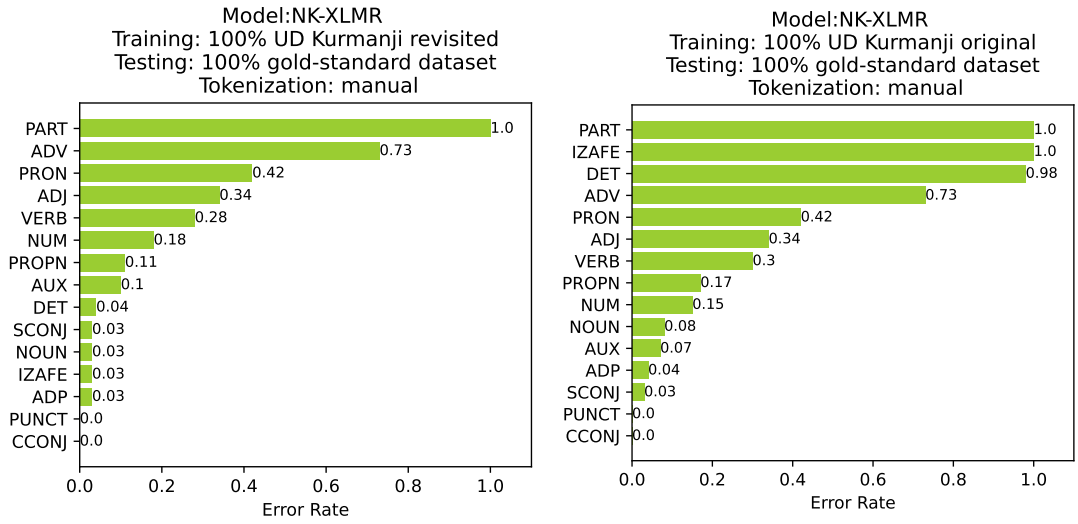


FIGURE 6.5: Confusion matrix of the NK-XLMR(revisited).

from our enhancements, demonstrating 0.05 and 0.06 accuracy improvement, respectively, and the and ADP and VERB to a lesser extent. In addition, we see that the tags DET and IZAFE enjoy huge improvement when trained on the UD Kurmanji revisited. However, we cannot consider it genuine since the IZAFE tag was not present in the UD Kurmanji original.

Nevertheless, it is evident that the NK-XLMR (original and revisited) exhibits a notable inadequacy in handling the PART and ADV tags. The output of NK-XLMR(revisited) shown in figure 6.4 and the error rates presented in figure 6.6a and figure 6.6b also verify this inadequacy. The tag PART has an error rate of 1.0, which means the model completely fails in recognizing tokens belonging to this tag correctly. We argue that this can be attributed to a misalignment in the annotation schemes between the UD Kurmanji and our gold-standard dataset, explained in subsection 4.4.1, rather than a limitation within the model itself.

Additionally, in Figure 6.6b, we see that IZAFE and DET also have error rates of 1.0 and 0.98. This happens due to the fact that NK-XLMR(original) has no knowledge of the Izafe and oblique case markers and, therefore, fails to perform POS tagging correctly when evaluated on the gold-standard dataset where those markers are explicitly represented.



(A) NK-XLMR(revisited)

(B) NK-XLMR(original)

FIGURE 6.6: Error rates of NK-XLMR(revisited) and NK-XLMR(original) models.

Regarding the second trend with regard to the tokenization methods, the most straightforward reason for this is the fact that the tokenization methods are generating, in most cases, either fewer or more tokens than the gold standard tokens. This can be attributed to the linguistic knowledge the tokenizer has about Northern Kurdish and its linguistic characteristics, such as the Izafe, oblique case markers multi-word expressions.

Chapter 7

Conclusions and Future Work

In the previous chapter, we presented the evaluation results of the various tokenization methods and the POS models. We presented our baseline model, highlighted the best-performing model, and compared the performance of the various POS models.
In this chapter, we summarize our study by encapsulating the challenges we faced and the approaches we took to overcome those challenges in section 7.1. In section 7.2, we underscore our contributions, findings, and the performance of our proposed POS pipeline for Northern Kurdish. Finally, we present our recommendations for future work in section 7.3.

7.1 Challenges

We observed that aside from the work of (Walther, Sagot, and Fort 2010), there has been no study dedicated to the task of POS tagging for Northern Kurdish. The absence of research on the language in the field of NLP is coupled with a scarcity of datasets, and the insufficient quality of available datasets, when found, adds another layer of complexity to the task. This is further propagated in the absence of effective tokenization methods for Northern Kurdish, with the exception of the KLPT tokenizer (Ahmadi 2020a), which, despite outstripping other methods, still falls short of the optimal performance.

Considering the mentioned challenges, it was evident that finding an effective fine-grained POS tagging method that offers the best trade-off between computational complexity, accuracy, and data scarcity adds a further dimension to an already multi-dimensional challenge. In order to address those described challenges, we employed the following multifaceted methodology:

1. **Enhanced the available training data**: The UD Kurmanji treebank was the only available annotated dataset that was characterized by its coarser granularity and some inconsistencies. To overcome this challenge, we took it upon ourselves to revisit it and enhance it to a fine-grained level, thus creating a separate version known as UD Kurmanji revisited while calling the initial version UD Kurmanji original.
   To evaluate the impact of data quality on POS tagging performance, our POS pipeline was trained in two separate iterations: initially using the UD Kurmanji original treebank and subsequently using the UD Kurmanji revisited treebank. This resulted in having, for

each POS model, two POS taggers of the same type where their difference is denoted in their names, as an example, CRF(original) and CRF(revisited).

2. **Introduced a gold-standard dataset**: In order to mitigate the data scarcity and to offer a realistic evaluation benchmark for our POS pipeline, we have curated a distinct, fine-grained, manually tokenized, and annotated dataset for the task of POS tagging. Our gold-standard dataset contains 136 sentences (2,937 tokens) written in Northern Kurdish and presented in raw (untokenized) and tokenized formats.

3. **Explored various tokenization strategies**: The tokens in the tokenized format of the gold-standard dataset served as ground truth for the task of tokenization, allowing us to underscore the quality and performance disparities between various tokenization methods when used on the untokenized format of the gold-standard dataset.

4. **Created multiple POS tagging models**: We have undertaken an empirical evaluation across a range of techniques varying from traditional statistical machine learning to more modern neural-based approaches, such as Unigram, HMM, ExtraTrees, AveragedPerceptron, CRF, BiLSTM, and transformer-based models to arrive at an informed judgment about their relative efficacy for the task of POS tagging for Northern Kurdish.

7.2 Conclusions

Thus far, we have managed to summarize the main challenges regarding the task of POS tagging for Northern Kurdish and our effective approaches to overcoming them. Given the challenges we identified, the main objective of this study was to address the task of POS tagging for Northern Kurdish by utilizing the currently available resources and techniques. In order to narrow the scope of our study, we formulated three research questions:

R1 What statistical or neural machine learning models can be effectively used to build a baseline POS tagger for Northern Kurdish?

R2 To what extent can we use the power of transformer-based multilingual language models to create a state-of-the-art POS tagger for Northern Kurdish?

R3 What are the linguistic features of Northern Kurdish that can affect part-of-speech tagging?

Answering our first research question, our multifaceted approach for this study enabled us to establish a baseline POS tagger for Northern Kurdish using the Unigram(revisited) model with an accuracy of 0.73 and a macro-averaged F1 score of 0.59 when evaluated on our gold-standard dataset. On the other hand, the CRF(revisited) model achieves the second-best performance with 0.84 and 0.74 for accuracy and macro-averaged F1 score, making it the best-performing model among statistical POS tagging models.

Regarding our second research question, the transformer-based NK-XLMR(revisited) outperforms all other models with an accuracy of 0.87 and a macro-averaged F1 score of 0.77. Therefore, setting a new state-of-the-art performance for the task of POS tagging for Northern Kurdish. Compared to the work of (Walther, Sagot, and Fort 2010), where their POS tagger for Northern Kurdish was evaluated only on 13 sentences, our results prove to be more reliable considering the granularity of linguistic features presented in our gold-standard dataset and the number of test sentences (136 sentences) we used for the evaluation.

As an integral part of this research, we have detailed various linguistic features of Northern Kurdish, such as the Izafe and oblique case markers and contracted prepositions. We successfully demonstrated the effect of those linguistic features on the task by evaluating both variants of the models (original and revisited). Our POS tagging models trained on the UD Kurmanji revisited showed improvements on NOUN, PROPN, VERB, and ADP POS tags. Thus answering our third research question by highlighting the impact of those linguistic features on the task of POS tagging for Northern Kurdish.

7.3 Future work

Based on our work and observations, we recommend the following future work directions:

1. Creating more consistent and high-quality annotated datasets for Northern Kurdish.
2. Using our NK-XLMR(revisited) POS tagger as a pre-annotation tool for available datasets in order to create large-scale annotated data.
3. Working on the task of tokenization for Northern Kurdish and improving the currently available tokenizers for Northern Kurdish.
4. Working on the challenge of tagging out-of-vocabulary words in Northern Kurdish.
5. Using our NK-XLMR(revisited) POS tagger to perform POS tagging for other dialects of Kurdish, such as Central Kurdish, and potentially for other closely related languages.

Bibliography

Joshi, Pratik et al. (2020). "The State and Fate of Linguistic Diversity and Inclusion in the NLP World." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6282–6293.

Ruder, Sebastian (2020). *Why You Should Do NLP Beyond English*. http://ruder.io/nlp-beyond-english.

Ruder, Sebastian, Anders Søgaard, and Ivan Vulić (2019). "Unsupervised Cross-Lingual Representation Learning." In: *Proceedings of ACL 2019, Tutorial Abstracts*, pp. 31–38.

Magueresse, Alexandre, Vincent Carles, and Evan Heetderks (2020). "Low-resource languages: A review of past work and future challenges." In: *arXiv preprint arXiv:2006.07264*.

Singh, Anil Kumar (2008). "Natural Language Processing for Less Privileged Languages: Where do we come from? Where are we going?" In: *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Cieri, Christopher et al. (2016). "Selection criteria for low resource language programs." In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 4543–4549.

Tsvetkov, Yulia (July 2017). *Opportunities and Challenges in Working with Low-Resource Languages*. https://www.cs.cmu.edu/~ytsvetko/jsalt-part1.pdf. Carnegie Mellon University.

Ahmadi, Sina (2020a). "KLPT–Kurdish language processing toolkit." In: *Proceedings of second workshop for NLP open source software (NLP-OSS)*, pp. 72–84.

Ruder, Sebastian (Oct. 2019). *Unsupervised Cross-lingual Representation Learning*. https://www.ruder.io/unsupervised-cross-lingual-learning/. Carnegie Mellon University.

Gimpel, Kevin et al. (2010). *Part-of-speech tagging for twitter: Annotation, features, and experiments*. Tech. rep. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Schmid, Helmut (1994). "Probabilistic Part-of-Speech Tagging Using Decision Trees." In: *Proceedings of the International Conference on New Methods in Language Processing*. Manchester, UK.

Kanakaraddi, Suvarna G and Suvarna S Nandyal (2018). "Survey on parts of speech tagger techniques." In: *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*. IEEE, pp. 1–6.

Mitkov, Ruslan (2022). *The Oxford handbook of computational linguistics*. Oxford University Press.

Jurafsky, Dan and James H Martin (2023). *Speech and language processing. 3rd ed. draft*. Prentice Hall.

Chiche, Alebachew and Betselot Yitagesu (2022). "Part of speech tagging: a systematic review of deep learning and machine learning approaches." In: *Journal of Big Data* 9.1, pp. 1–25.

Anonby, Erik (2022). "Phonological Variation in Kurdish." In: *Structural and Typological Variation in the Dialects of Kurdish*. Springer, pp. 65–110.

Ashti Afrasyaw, JAF and Sema Koç Kayhan (2021). "A Brief Overview of Kurdish Natural Language Processing." In: *Proceedings of Workshop on Intelligent Information Systems*, p. 185.

Jacksi, Karwan, Ismael Ali, et al. (2023). "The Kurdish Language corpus: state of the art." In: *Science Journal of University of Zakho* 11.1, pp. 127–133.

Walther, Géraldine, Benoît Sagot, and Karën Fort (2010). "Fast development of basic NLP tools: Towards a lexicon and a POS tagger for Kurmanji Kurdish." In: *International conference on lexis and grammar*.

Gökırmak, Memduh and Francis Tyers (2017). "A dependency treebank for Kurmanji Kurdish." In: *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pp. 64–72.

Ahmadi, Sina, Hossein Hassani, and John P McCrae (2019). "Towards electronic lexicography for the Kurdish language." In: *Proceedings of the sixth biennial conference on electronic lexicography (eLex)*. eLex 2019.

Thackston, WM (2006a). *Kurmanji Kurdish:-A Reference Grammar with Selected Readings*. Renas Media.

Eppel, Michael (2016). *A People without a State: The Kurds from the Rise of Islam to the Dawn of Nationalism*. University of Texas Press.

Phillips, David L (2017). *The Kurdish Spring: A New Map of the Middle East*. Routledge.

Allsopp, Harriet and Wladimir van Wilgenburg (2019). *The Kurds of Northern Syria: Governance, Diversity and Conflicts:* International series of monographs on physics. I.B. Tauris. ISBN: 978-1-83860-445-5.

Bozarslan, Hamit, Cengiz Gunes, and Veli Yadirgi (2021). *The Cambridge History of the Kurds*. Cambridge University Press.

Matras, Yaron (2019). "Revisiting Kurdish dialect geography: Findings from the Manchester database." In: *Current issues in Kurdish linguistics* 1, p. 225.

Ahmadi, Sina (2021). "A Formal Description of Sorani Kurdish Morphology." In: *arXiv preprint arXiv:2109.03942*.

Thackston, WM (2006b). "Sorani Kurdish: A Reference Grammar with Selected Readings http://www. fas. harvard. edu/~ iranian." In: *Sorani/sorani_complete. pdf*.

Esmaili, Kyumars Sheykh and Shahin Salavati (2013). "Sorani Kurdish versus Kurmanji Kurdish: an empirical comparison." In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 300–305.

Matras, Yaron, Geoffrey Haig, and Ergin Öpengin (2022). *Structural and Typological Variation in the Dialects of Kurdish*. Springer Nature.

Ahmadi, Sina (2020b). "A tokenization system for the Kurdish language." In: *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pp. 114–127.

Gündoğdu, Songül et al. (2019). *Current issues in Kurdish linguistics*. Vol. 1. University of Bamberg Press.

Aydoğan, Mistefa (2012). *Rêbera rastnivîsînê*. Weşanxaneya Rûpelê. Ziman. Rûpel. ISBN: 9786058651609. URL: https://books.google.nl/books?id=Z7lDnwEACAAJ.

Weqfa, Mezopotamyay (2019). *Rêbera rastnivîsînê*. Weqfa Mezopotamyayê. ISBN: 9786058097506. URL: https://books.google.nl/books?id=Pb-szQEACAAJ.

Tan, Sami (2015). *Rêzimana Kurmancî*. Weşanên Enstîtuya Kurdî ya Stenbolê. ISBN: 9789756282540. URL: https://books.google.nl/books?id=To1vMwEACAAJ.

Haig, Geoffrey (2007). "Grammatical borrowing in Kurdish (Northern group)." In: *Grammatical borrowing in cross-linguistic perspective*, pp. 165–183.

Comrie, Bernard (2017). "Languages of the world." In: *The handbook of linguistics*, pp. 21–38.

Samvelian, Pollet (2007). "A (phrasal) affix analysis of the Persian Ezafe1." In: *Journal of Linguistics* 43.3, pp. 605–645.

Karimi, Yadgar (2007). "Kurdish Ezafe construction: Implications for DP structure." In: *Lingua* 117.12, pp. 2159–2177.

Gutman, Ariel (2016). "Attributive Constructions in North-Eastern Neo-Aramaic: Areal, Typological and Historical Perspectives." PhD thesis.

Schuster, Mike and Kaisuke Nakajima (2012). "Japanese and korean voice search." In: *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 5149–5152.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units." In: *54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL), pp. 1715–1725.

Kudo, Taku (2018). "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." In: pp. 66–75.

Kudo, Taku and John Richardson (2018). "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71.

Kenton, Jacob Devlin Ming-Wei Chang and Lee Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: pp. 4171–4186.

Conneau, Alexis, Kartikay Khandelwal, et al. (2020). "Unsupervised Cross-lingual Representation Learning at Scale." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451.

Solaiman, Irene et al. (2019). "Release strategies and the social impacts of language models." In: *arXiv preprint arXiv:1908.09203*.

Bird, Steven, Ewan Klein, and Edward Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."

Kiss, Tibor and Jan Strunk (2006). "Unsupervised multilingual sentence boundary detection." In: *Computational linguistics* 32.4, pp. 485–525.

Marcus, Mitch, Beatrice Santorini, and Mary Ann Marcinkiewicz (1993). "Building a Large Annotated Corpus of English: The Penn Treebank." In: *Computational Linguistics* 19.2, pp. 313–330.

Rayson, Paul and Roger Garside (1998). "The claws web tagger." In: *ICAME Journal* 22, pp. 121–123.

De Marneffe, Marie-Catherine et al. (2021). "Universal dependencies." In: *Computational linguistics* 47.2, pp. 255–308.

Petrov, Slav, Dipanjan Das, and Ryan McDonald (2012). "A Universal Part-of-Speech Tagset." In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pp. 2089–2096.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). "Contextual string embeddings for sequence labeling." In: *Proceedings of the 27th international conference on computational linguistics*, pp. 1638–1649.

Shen, Libin, Giorgio Satta, and Aravind Joshi (2007). "Guided learning for bidirectional sequence classification." In: *Proceedings of the 45th annual meeting of the association of computational linguistics*, pp. 760–767.

Karlsson, Fred (1990). "Constraint grammar as a framework for parsing running text." In: *COLING 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*.

Brill, Eric (1992). *A simple rule-based part of speech tagger*. Tech. rep. Pennsylvania Univ Philadelphia Dept of Computer and Information Science.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data." In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 282–289. ISBN: 1558607781.

McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity." In: *The bulletin of mathematical biophysics* 5, pp. 115–133.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Collins, Michael (2002). "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms." In: *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002)*, pp. 1–8.

Rumelhart, David E et al. (1986). "Sequential thought processes in PDP models." In: *Parallel distributed processing: explorations in the microstructures of cognition* 2, pp. 3–57.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult." In: *IEEE transactions on neural networks* 5.2, pp. 157–166.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural computation* 9.8, pp. 1735–1780.

Chevalier, Guillaume (2018). "LARNN: linear attention recurrent neural network." In: *arXiv preprint arXiv:1808.05578*.

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM networks." In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 4. IEEE, pp. 2047–2052.

Mikolov, Tomas et al. (2013). "Efficient estimation of word representations in vector space." In: *arXiv preprint arXiv:1301.3781*.

Al-Rfou, Rami, Bryan Perozzi, and Steven Skiena (2013). "Polyglot: Distributed Word Representations for Multilingual NLP." In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pp. 183–192.

Bojanowski, Piotr et al. (2017). "Enriching Word Vectors with Subword Information." In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146.

Vaswani, Ashish et al. (2017). "Attention is all you need." In: *Advances in neural information processing systems* 30.

Radford, Alec et al. (2019). "Language models are unsupervised multitask learners." In: *OpenAI blog* 1.8, p. 9.

Liu, Yinhan et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach." In: *arXiv preprint arXiv:1907.11692*.

Kondratyuk, Dan and Milan Straka (2019). "75 Languages, 1 Model: Parsing Universal Dependencies Universally." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2779–2795.

Penedo, Guilherme et al. (2023). "The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only." In: *arXiv preprint arXiv:2306.01116*.

Papineni, Kishore et al. (2002). "Bleu: a method for automatic evaluation of machine translation." In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318.

Reiter, Ehud (2018). "A structured review of the validity of BLEU." In: *Computational Linguistics* 44.3, pp. 393–401.

Denis, Pascal and Benoît Sagot (2009). "Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort." In: *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*, pp. 110–119.

Plank, Barbara, Anders Søgaard, and Yoav Goldberg (2016). "Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 412–418.

Zeman, Daniel et al. (2022). *Universal Dependencies 2.11*. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. URL: http://hdl.handle.net/11234/1-4923.

Smith, Aaron et al. (2018). "82 Treebanks, 34 Models: Universal Dependency Parsing with Multi-Treebank Models." In: pp. 113–123.

Qi, Peng, Yuhao Zhang, et al. (2020). "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*, pp. 101–108.

Qi, Peng, Timothy Dozat, et al. (2018). "Universal Dependency Parsing from Scratch." In: *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 160–170.

Eskander, Ramy, Smaranda Muresan, and Michael Collins (2020). "Unsupervised cross-lingual part-of-speech tagging for truly low-resource scenarios." In: *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 4820–4831.

ImaniGooghari, Ayyoob et al. (2022). "Graph-Based Multilingual Label Propagation for Low-Resource Part-of-Speech Tagging." In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1577–1589.

Fang, Meng and Trevor Cohn (2017). "Model Transfer for Tagging Low-resource Languages using a Bilingual Dictionary." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 587–593.

Cardenas, Ronald et al. (2019). "A Grounded Unsupervised Universal Part-of-Speech Tagger for Low-Resource Languages." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2428–2439.

Pradhan, Ashish and Archit Yajnik (2023). "Parts-of-speech tagging of Nepali texts with Bidirectional LSTM, Conditional Random Fields and HMM." In: *Multimedia Tools and Applications*, pp. 1–17.

Vries, Wietse de, Martijn Wieling, and Malvina Nissim (2022). "Make the best of cross-lingual transfer: Evidence from POS tagging with over 100 languages." In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7676–7685.

Mollanorozy, Sepideh, Marc Tanti, and Malvina Nissim (2023). "Cross-lingual Transfer Learning with Persian." In: *Proceedings of the 5th Workshop on Research in Computational Linguistic Typology and Multilingual NLP*, pp. 89–95.

Farahani, Mehrdad et al. (2021). "Parsbert: Transformer-based model for persian language understanding." In: *Neural Processing Letters* 53, pp. 3831–3847.

Vries, Wietse de, Martijn Bartelds, et al. (2021). "Adapting Monolingual Models: Data can be Scarce when Language Similarity is High." In: pp. 4901–4907.

Bao, Zuyi et al. (2019). "Low-Resource Sequence Labeling via Unsupervised Multilingual Contextualized Representations." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1028–1039.

Doostmohammadi, Ehsan, Minoo Nassajian, and Adel Rahimi (2020). "Persian Ezafe Recognition Using Transformers and Its Role in Part-Of-Speech Tagging." In: pp. 961–971.

Muller, Benjamin et al. (2021). "When Being Unseen from mBERT is just the Beginning: Handling New Languages With Multilingual Language Models." In: *Proceedings of the 2021*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 448–462.

Suárez, Pedro Javier Ortiz, Benoît Sagot, and Laurent Romary (Jan. 2023). *Open Super-large Crawled Aggregated coRpus-OSCAR 23.01*. https://oscar-project.github.io/documentation/versions/oscar-2301/. Inria, Paris, France.

Straka, Milan (2018). "UDPipe 2.0 prototype at CoNLL 2018 UD shared task." In: *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 197–207.

Martin, Louis et al. (2020). "CamemBERT: a Tasty French Language Model." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7203–7219.

Conneau, Alexis and Guillaume Lample (2019). "Cross-lingual language model pretraining." In: *Advances in neural information processing systems* 32.

Straka, Milan, Jana Straková, and Jan Hajič (2019). "Evaluating contextualized embeddings on 54 languages in POS tagging, lemmatization and dependency parsing." In: *arXiv preprint arXiv:1908.07448*.

Straka, Milan, Jakub Náplava, et al. (2021). "RobeCzech: Czech RoBERTa, a monolingual contextualized language representation model." In: *Text, Speech, and Dialogue: 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6–9, 2021, Proceedings 24*. Springer, pp. 197–209.

Arkhipov, Mikhail et al. (2019). "Tuning multilingual transformers for language-specific named entity recognition." In: *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pp. 89–93.

Nguyen, Minh Van et al. (Apr. 2021). "Trankit: A Light-Weight Transformer-based Toolkit for Multilingual Natural Language Processing." In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, pp. 80–90. DOI: 10.18653/v1/2021.eacl-demos.10. URL: https://aclanthology.org/2021.eacl-demos.10.

Nivre, Joakim et al. (2020). "Universal Dependencies v2: An evergrowing multilingual treebank collection." In: *arXiv preprint arXiv:2004.10643*.

Pfeiffer, Jonas, Andreas Rücklé, et al. (Oct. 2020). "AdapterHub: A Framework for Adapting Transformers." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 46–54. DOI: 10.18653/v1/2020.emnlp-demos.7. URL: https://aclanthology.org/2020.emnlp-demos.7.

Pfeiffer, Jonas, Ivan Vulić, et al. (Nov. 2020). "MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 7654–7673. DOI: 10.18653/v1/2020.emnlp-main.617. URL: https://aclanthology.org/2020.emnlp-main.617.

Micallef, Kurt et al. (2022). "Pre-training Data Quality and Quantity for a Low-Resource Language: New Corpus and BERT Models for Maltese." In: *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pp. 90–101.

Gessler, Luke and Amir Zeldes (2022). "MicroBERT: Effective Training of Low-resource Monolingual BERTs through Parameter Reduction and Multitask Learning." In: *Proceedings of the The 2nd Workshop on Multi-lingual Representation Learning (MRL)*, pp. 86–99.

Suárez, Pedro Javier Ortiz, Benoît Sagot, and Laurent Romary (2019). "Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures." In: *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache.

Palen-Michel, Chester, June Kim, and Constantine Lignos (July 2022). "Multilingual Open Text Release 1: Public Domain News in 44 Languages." In: *Proceedings of the Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, pp. 2080–2089. URL: https://aclanthology.org/2022.lrec-1.224.

Esmaili, Kyumars Sheykh, Donya Eliassi, et al. (2013). "Building a test collection for Sorani Kurdish." In: *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*. IEEE, pp. 1–7.

Buchholz, Sabine and Erwin Marsi (2006). "CoNLL-X shared task on multilingual dependency parsing." In: *Proceedings of the tenth conference on computational natural language learning (CoNLL-X)*, pp. 149–164.

Silveira, Natalia et al. (2014). "A Gold Standard Dependency Corpus for English." In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Seddah, Arij Riabi Menel Mahamdi Djamé (2023). "Enriching the NArabizi Treebank: A Multifaceted Approach to Supporting an Under-Resourced Language." In: p. 266.

Seddah, Djamé et al. (2020). "Building a user-generated content North-African Arabizi treebank: Tackling hell." In: *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pp. 1139–1150.

Rust, Phillip et al. (2021). "How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models." In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3118–3135.

Akiba, Takuya et al. (2019). "Optuna: A Next-generation Hyperparameter Optimization Framework." In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Huang, Xuedong et al. (2001). *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Akbik, Alan, Tanja Bergmann, et al. (2019). "FLAIR: An easy-to-use framework for state-of-the-art NLP." In: *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 54–59.

Huang, Zhiheng, Wei Xu, and Kai Yu (2015). “Bidirectional LSTM-CRF models for sequence tagging.” In: *arXiv preprint arXiv:1508.01991*.

Grave, Édouard et al. (2018). “Learning Word Vectors for 157 Languages.” In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Okazaki, Naoaki (2007). *CRFsuite: a fast implementation of Conditional Random Fields (CRFs)*. URL: http://www.chokkan.org/software/crfsuite/.

Haspelmath, Martin (2014). “The Leipzig style rules for linguistics.” In: *Max Planck Institute for Evolutionary Anthropology, Leipzig*.

Loshchilov, Ilya and Frank Hutter (2018). “Decoupled Weight Decay Regularization.” In: *International Conference on Learning Representations*.

Appendix A

Appendix

A.1 Linguistic symbols

Symbol	Meaning
.SG	singular noun
.PL	plural noun
.F	feminine noun
.M	masculine noun
.OBL	the noun, proper noun or pronoun in the oblique case
-OBL	oblique case marker
.EZF	the noun, proper noun in the Izafe case
-EZF	Izafe case marker
-INDF	indefinite marker
-VOC	vocative marker
.ADP	adposition/preposition
-PART	particle present as suffix, mostly *î*

TABLE A.1: Linguistic symbols and their meanings used throughout this study, based on the Leipzig Glossing Rules (Haspelmath 2014).

A.2 Tokenization methods performance

The following three plots represent the performance of the various tokenization methods on revisited and original versions of the UD Kurmanji treebank. For the performance on the UD Kurmanji revisited in figure A.1, the KLPT tokenizer has the highest number of perfectly tokenized sentences. While, for the UD Kurmanji original, in figure A.2, the NTLK tokenizer outperforms the rest with the highest numbers of perfectly tokenized sentences. The performance is expected since the sentences in the UD Kurmanji original are, by default, separated by whitespaces without taking Izafe or oblique case markers into account.

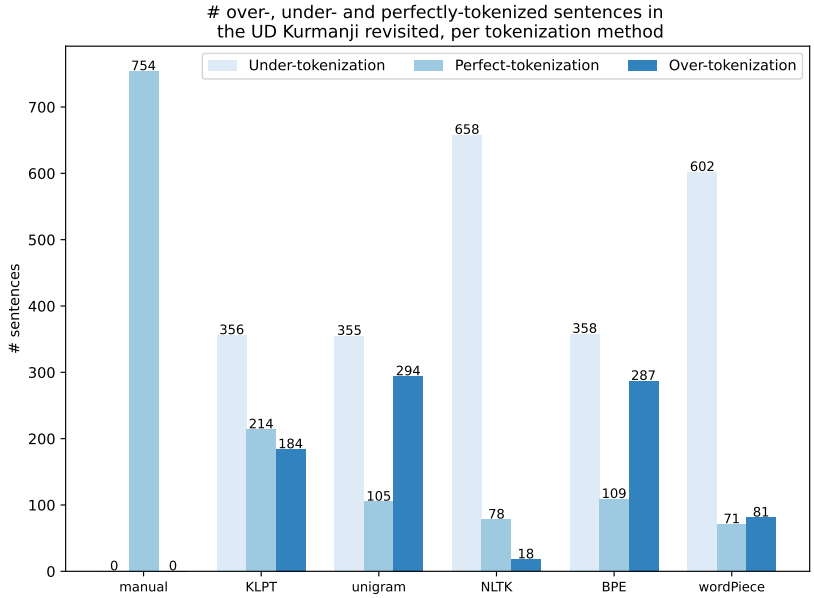


FIGURE A.1: Number of over-,under-, and perfectly-tokenized sentences in the UD Kurmanji revisited, per tokenization method.

A.2.1 Tokenization methods BLEU scores

Table A.3 and table A.2 provide the BLEU scores of various tokenization methods when applied on the UD Kurmanji original and revisited, respectively. On the one hand, we see in table A.3 that the NLTK tokenizer outperforms all other tokenizers. This performance is expected since the tokens in the UD Kurmanji original agree with tokens (morphological words) delimited by whitespace in the raw (untokenized) sentences in the treebank. On the other hand, within table A.2, we see that the KLPT tokenizer outperforms all other tokenizers because it has sufficient knowledge of the finer linguistic features of Northern Kurdish such as case markers that present in the UD Kurmanji revisited.

Tokenization method	BLEU-score (UD Kurmanji revisited)			
	BLEU-1	**BLEU-2**	**BLEU-3**	**BLEU-4**
manual	*1.00*	*1.00*	*1.00*	*1.00*
KLPT	**0.80**	**0.73**	**0.68**	**0.63**
NLTK	0.65	0.57	0.50	0.44
unigram	0.66	0.56	0.48	0.40
BPE	0.61	0.50	0.42	0.34
wordPiece	0.57	0.48	0.40	0.33

TABLE A.2: BLEU scores for all tokenization methods on the UD Kurmanji revisited.

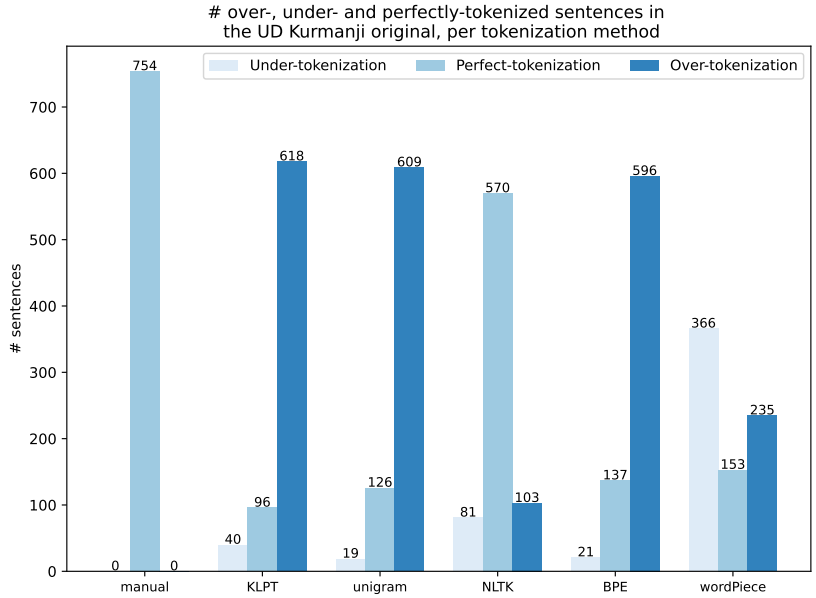


FIGURE A.2: Number of over-,under-, and perfectly-tokenized sentences in the UD Kurmanji original, per tokenization method.

Tokenization method	BLEU-score (UD Kurmanji original)			
	BLEU-1	**BLEU-2**	**BLEU-3**	**BLEU-4**
manual	*1.00*	*1.00*	*1.00*	*1.00*
KLPT	0.72	0.64	0.58	0.51
NLTK	**0.97**	**0.95**	**0.94**	**0.93**
unigram	0.70	0.63	0.57	0.51
BPE	0.70	0.63	0.58	0.53
wordPiece	0.76	0.69	0.62	0.56

TABLE A.3: BLEU scores for all tokenization methods on the UD Kurmanji original.

A.3 POS models details and hyperparameters

A.3.1 HMM hyperparameters

The optimized hyperparameter of the HMM model:

- smoothing parameter (gamma): 0.1

A.3.2 ExtraTrees hyperparameters

The optimized hyperparameters of the ExtraTrees model:

- Random state: 42
- Number of estimators: 200
- Max features: 8

A.3.3 ExtraTrees features set

The set of features we have manually created and used for the ExtraTrees POS model.

```
def extract_features(sentences):
    features = []
    for sentence in sentences:
        for index in range(len(sentence)):
            k = index
            features_dict = {}
            token,tag = sentence[k]
            is_number = False
            try:
                if float(token):
                    is_number = True
            except:
                pass
            features_dict['token'] = token
            features_dict['tag'] = tag
            features_dict['lower_cased_token'] = token.lower()
            features_dict['suffix1'] = token[-1]
            features_dict['suffix2'] = token[-2:]
            features_dict['suffix3'] = token[-3:]
            features_dict['is_capitalized'] = token.isalpha() and token[0].
    isupper()
            features_dict['is_number'] = is_number
            features_dict['is_first'] = k == 0
            features_dict['is_last'] = k == len(sentence) - 1
            if k == 0:
                features_dict['prev_tag'] = "<start_tag>"
                features_dict['prev_prev_tag'] = "<start_tag>_<start_tag>"
                features_dict['prev_token'] = "<start_token>"
            if k == 1:
                token1, tag1= sentence[k-1]
                features_dict['prev_tag'] = tag1
                features_dict['prev_prev_tag'] = "<start_tag>_"+ tag1
                features_dict['prev_token'] = token1
            elif k > 1:
                token1, tag1= sentence[k-1]
                token2, tag2= sentence[k-2]
                features_dict['prev_tag'] = tag1
                features_dict['prev_prev_tag'] = tag2+"_"+ tag1
                features_dict['prev_token'] = token1
            if k < len(sentence)-1:
                token,tag = sentence[k+1]
                features_dict['next_tag'] = tag
                features_dict['next_token'] = token
            features.append(features_dict)
    return features
```

Listing A.1: 'The ExtraTrees features extractor function for Northern Kurdish.'

A.3.4 Averaged Perceptron hyperparameters

The optimized hyperparameters of this model:

- Number of training iterations: 1024

A.3.5 BiLSTM hyperparameters

The optimized hyperparameters of the BiLSTM model:

- RNN type: BiLSTM
- BiLSTM parameters:
  - Input features: 300 (the same as the fastText embeddings dimensions)
  - RNN layer hidden size: 128
  - Number of RNN layers: 3
- Max epochs: 100
- Dropout: 0.2
- Optimizer: Adam optimizer (Loshchilov and Hutter 2018)
- Learning rate:0.002915119831676156
- Mini batch size: 24
- Patience: 8
- Main evaluation metric: (”macro avg”, ”f1-score”)

A.3.6 CRF hyperparameters

The optimized hyperparameters of the CRF model:

- L1 regularization (c_1): 0.10
- L2 regularization (c_2): 0.20
- Number of training iterations: 200

A.3.7 NK-XLMR hyperparameters

The following hyperparameters are employed during the fine-tuning procedure:

- Max epoch: 100
- Batch size: 16

A.4 POS models results

A.4.1 Macro-averaged F1 scores for all proposed POS models

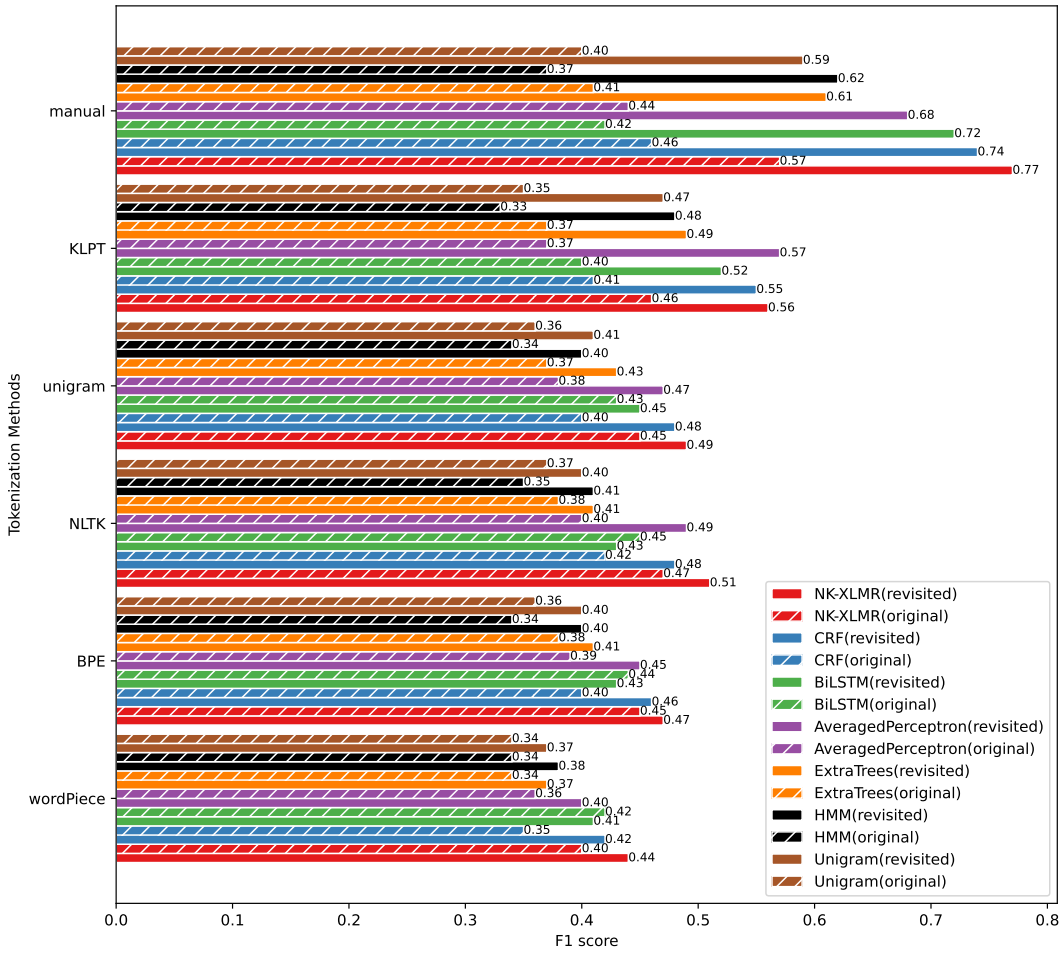


FIGURE A.3: The macro-averaged F1 scores of the proposed POS models in relation to the training data and tokenization methods.

A.4.2 Confusion Matrix of NK-XLMR(original) model

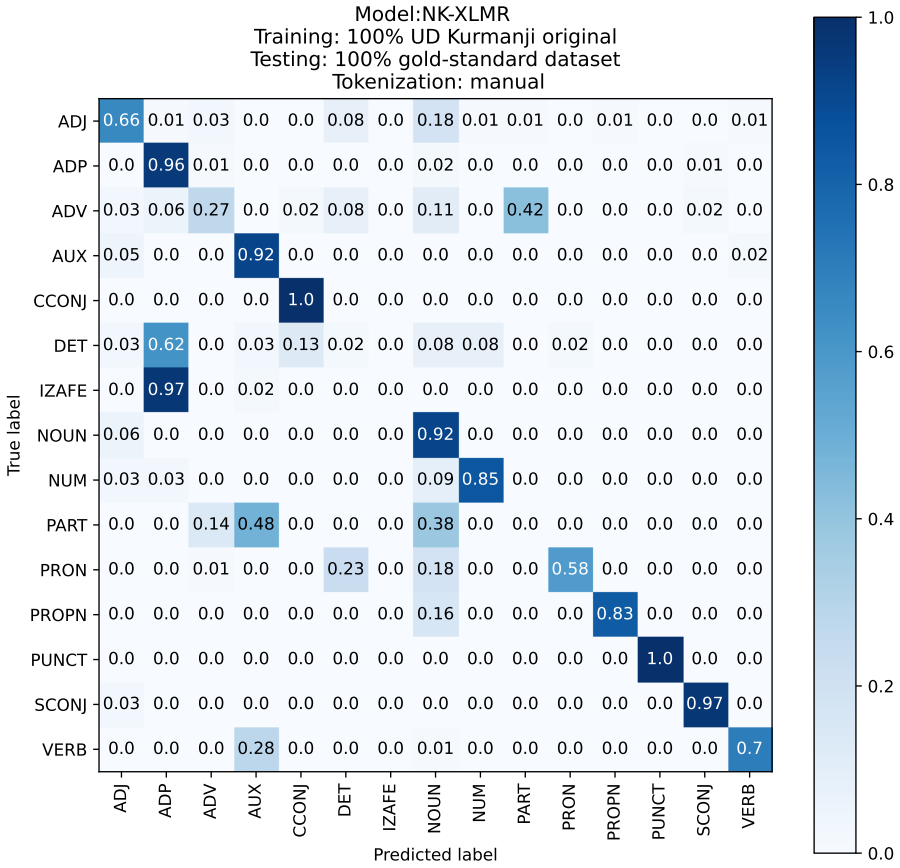


FIGURE A.4: Confusion matrix of the NK-XLMR(original).

A.4.3 POS Models output

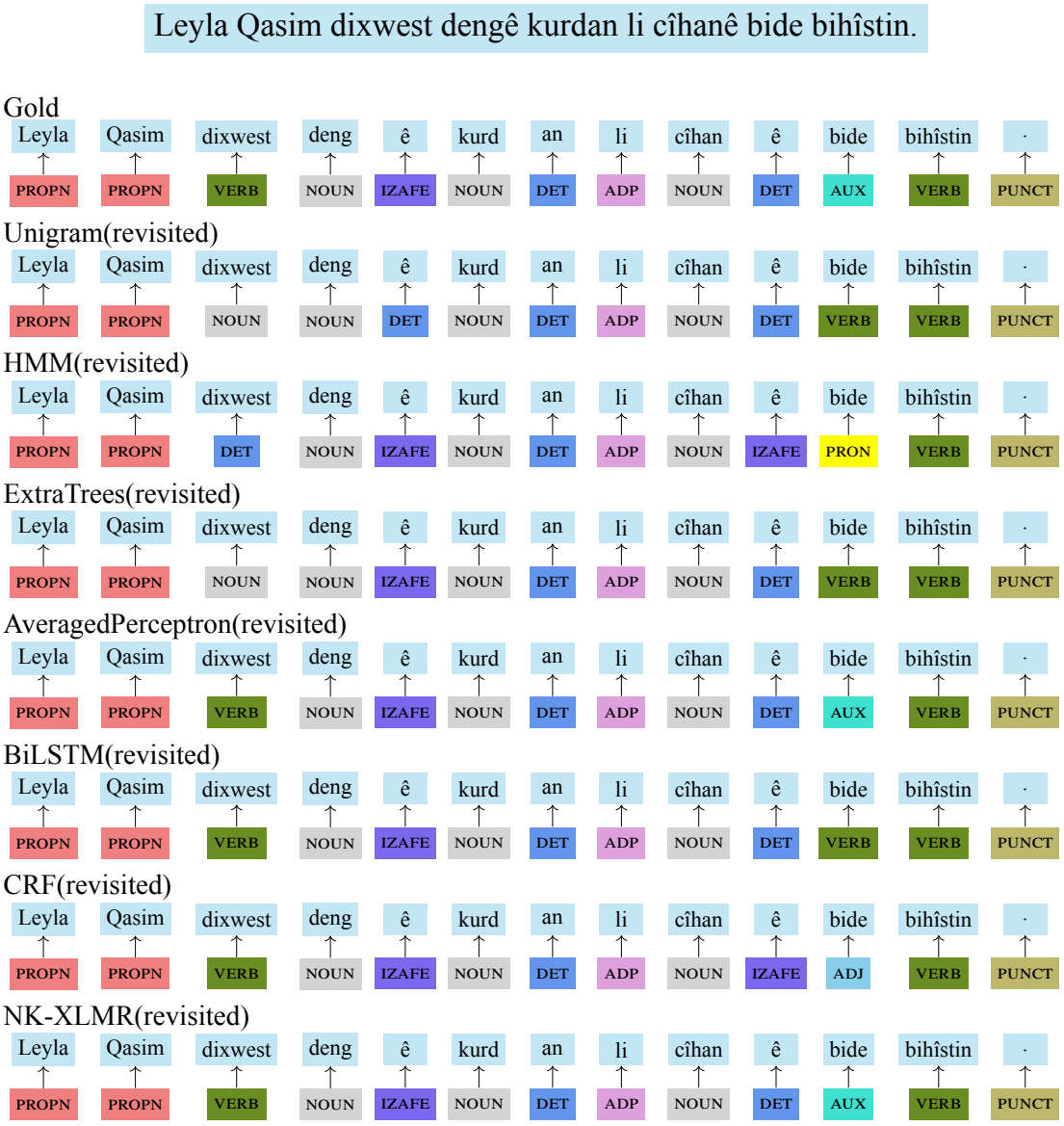


FIGURE A.5: Outputs of all proposed POS models (trained on UD Kurmanji revisited) compared to the gold-standard annotations for a sentence drawn from the gold-standard dataset *Leyla Qasim dixwest dengê kurdan li cîhanê bide bihîstin.* 'Leyla Qasim wanted to make the voice of the Kurds heard in the world.'

FIGURE A.6: Outputs of all proposed POS models (trained on UD Kurmanji revisited) compared to the gold-standard annotations.*bi evî awayî jî, ew mezintirîn neteweya niha ya bê dewlet in.* 'thus, they are the largest stateless nation now.'

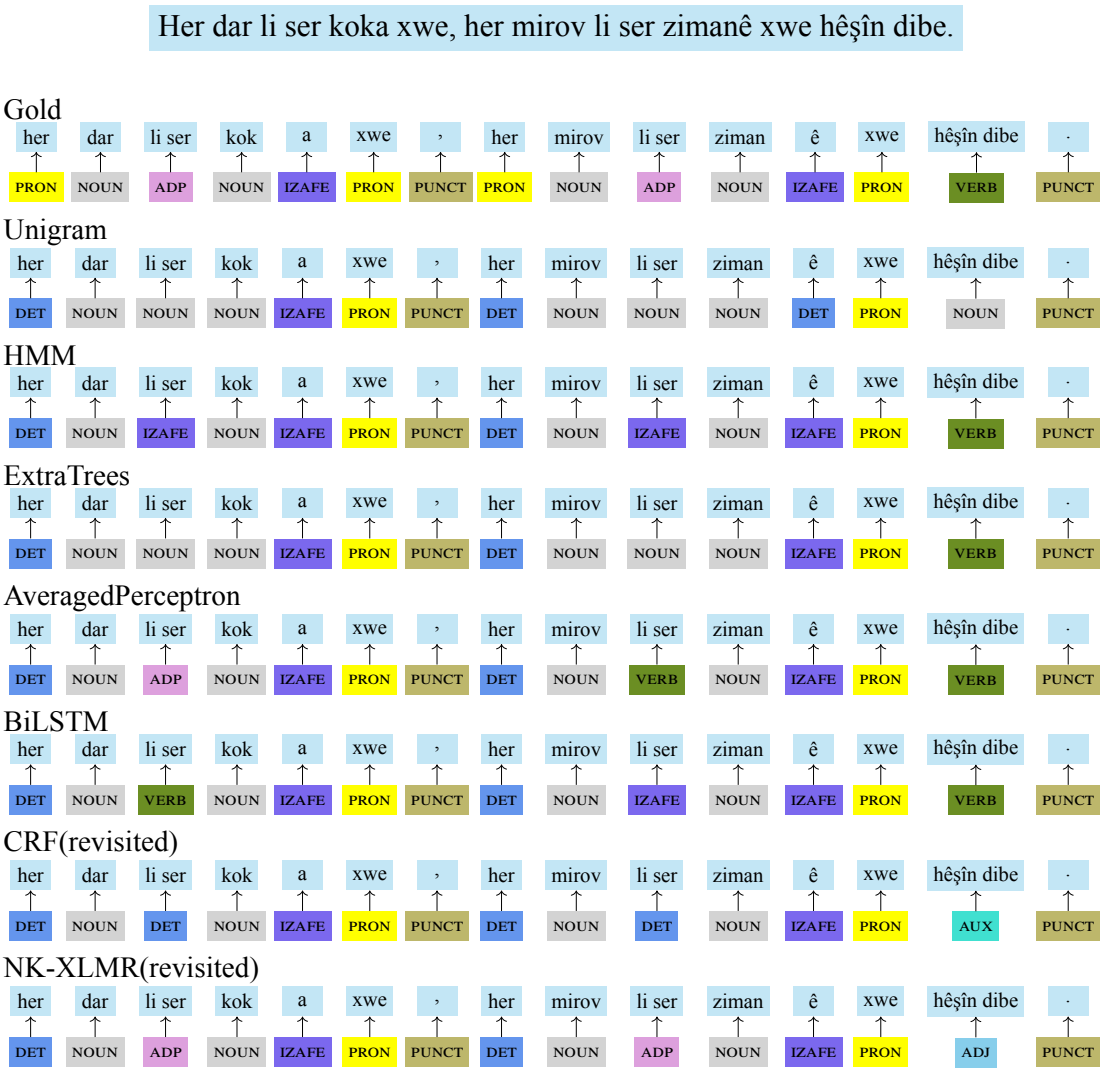


FIGURE A.7: Outputs of all proposed POS models (trained on UD Kurmanji revisited) compared to the gold-standard annotations of the Kurdish proverb: *Her dar li ser koka xwe, her mirov li ser zimanê xwe hêşîn dibe.* 'As every tree stands over its roots, so does every human blossom with their mother tongue.'