Universidad Michoacana de San Nicolás de Hidalgo

División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica

CORPUS COMPILATION AND NATURAL LANGUAGE MODEL EVALUATION FOR THE PURÉPECHA LANGUAGE

THESIS

A dissertation submitted for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Computer Systems

by

Daniel Castro Pineda

Dr. Jaime Cerda Jacobo

Thesis Advisor

Dr. José Ortiz Béjar

Thesis Co-advisor

Morelia, Michoacán, September 2025

CORPUS COMPILATION AND NATURAL LANGUAGE MODEL EVALUATION FOR THE PURÉPECHA LANGUAGE

Los Miembros del Jurado de Examen de Grado aprueban la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de **Daniel Castro Pineda.**

Dr. Luis Eduardo Gamboa Guzman
Presidente del Jurado

Dr. Jaime Cerda Jacobo
Director de Tesis

Dr. José Ortiz Bejar
Co-director

Dr. Alejandro Zamora Méndez
Vocal

Dr. Mario Graff Guerrero
Revisor Externo (INFOTEC)

Dr. J. Aurelio Medina Rios
Jefe de la División de Estudios de Posgrado
de la Facultad de Ingeniería Eléctrica. UMSNH
(Por reconocimiento de firmas)

UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Junio 2025

Acknowledgments

I express my gratitude for the support provided by Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) through Project CF-2023-I-1174 within the framework of the "Ciencia de Frontera 2023".

This work could not be done without access to the libraries, historic and newspapers archives of Centro de Cooperación Regional para la Educación de Adultos en América Latina y El Caribe (CREFAL), Universidad Intercultural Indígena de Michoacán (UIIM), Escuela Normal Indigena de Michoacán (ENIM), and Universidad Michoacana de San Nicolás de Hidalgo (UMSNH). Special thanks to the Linguistics Department of UMSNH and UIIM for their support, access to files, printed materials, and advice on the translation of some texts.

With great love to my family, girlfriend, and friends for their unconditional love and support through the highs and lows of my postgraduate studies.

To my thesis advisors, Dr. Jaime and Dr. Jose, for their patience, advice, teachings, and friendship.

Publication list

Cerda-Flores, J., **Castro-Pineda, D.**, Juarez, M. G., Hernandez-Mazariegos, R. I., Cerda-Jacobo, J., Graff, M., & Ortiz-Bejar, J. (2024, December 12–15). *Text classification based on ensembles of pre-trained models for sarcasm identification in Dravidian languages.* In *n Forum of Information Retrieval and Evaluation FIRE - 2024. DAIICT, Gandhinagar.* (CEUR Workshop Proceedings).

Resumen

La lengua Purépecha es una de las 68 lenguas indígenas en México y es hablada principalmente en el estado de Michoacán. Este documento presenta el primer corpus monolingüe de acceso público para la lengua purépecha. Los documentos recolectados para ensamblar el conjunto de datos fueron obtenidos de diferentes fuentes, tales como colegios normales indígenas, instituciones de gobierno y fuentes periodísticas, buscando reflejar el uso y estado general de esta lengua. Este trabajo busca ser un elemento clave para lingüistas e investigadores, presentando estadísticas sobre el conjunto de datos tales como mayor frecuencia en frases y palabras, decaimiento en la frecuencia de las mismas (Ley de Zipf) además de hápaxes. Adicionalmente, se entrenaron modelos de lenguaje como lo son los modelos transformer RoBERTA (que nombramos como RoBERTataka) y los basados en n-gramas, que posteriormente serán utilizados durante la evaluación del corpus y estarán disponibles de forma pública como modelos preentrenados. Finalmente, se evaluó el rendimiento de los modelos de lenguaje mediante una tarea de modelado de lenguaje mediante enmascarado (MLM) utilizando exclusivamente el conjunto de datos recabado.

Palabras Clave

- Corpus
- Procesamiento de Lenguaje Natural
- Purépecha
- Lenguas Poco Representadas
- Modelos de Lenguaje

Abstract

Purépecha is one of the 68 indigenous languages in Mexico and is spoken primarily in the state of Michoacán. This document presents the curation for the first publicly available monolinguistic corpora for the purépecha language. The documents collected to build this dataset were gathered from different sources, such as indigenous colleges, government institutions, and news journals, to reflect the general state and use of the language. The purpose of this project is to serve as a critical resource for linguists and researchers by presenting statistics on the dataset such as higher frequency in phrases and words and decay in their frequency (Zipf's Law) and hapaxes. In addition, the corpus was used to train a state-of-the-art deep learning transformer model RoBERTa (that we named as RoBERTataka) and an n-gram-based approach; those language models will be publicly available as pretrained models. Finally, the performance of the trained text models are compared on a Masked Language Modeling (MLM) task exclusively on the curated datasets.

Key Words

- Corpus
- Natural Language Processing
- Purépecha
- Low Resources Languages
- Language Models

Contents

List of Figures

List of Tables

Symbol list

$\sum$	Sum
$\prod$	Product
∞	Infinity
$\log(a)$	Logarithm of a
$P(B\|A)$	Probability of B given A
$\log P(a)$	Log probability of a
$Counts(w)$	Frequency of a word inside a given corpus
$Counts(w_1...w_i)$	Frequency of a n-gram inside a given corpus
w	word or token
w_i	ith word or token of a sentence.
$w_{n,\ldots,i-1}$	The context behind the token or word w_i.
W	sentence.
$\|W\|$	sentence length.
$\frac{\partial f}{\partial x}$	Partial derivative of f with respect of x
∇f	Gradient
η	Learning rate
σ	Sigmoid Activation Function
λ_i	Weighted values for n-gram interpolation
$\max(a, b)$	Maximum value between inputs

List of Acronyms

ACL Annual Meeting of the Association for Computational Linguistics

AN Artificial Neuron

ANN Artificial Neural Network

ASCII American Standard Code for Information Interchange

BERT Bidirectional Encoder Representation from Transformers

BLiMP Benchmark of Linguistic Minimal Pairs

BLUE Biomedical Language Understanding Evaluation

BP Backpropagation

BPE Byte Pair Encoding

Byte-level BPE Byte-level Byte Pair Encoding

CONAHCYT Consejo Nacional de Humanidades, Ciencias y Tecnologías

CREFAL Centro de Cooperación Regional para la Educación de Adultos en América Latina y El Caribe

DL Deep Learning

EACL European Chapter of the Association for Computational Linguistics

ELP Endangered Languages Project

EMNLP Empirical Methods in Natural Language Processing

ENIM Escuela Normal Indigena de Michoacán

FFNN Feedforward Neural Network

FILAC Fondo para el Desarrollo de los pueblos Indígenas de América Latina y Caribe

GD Gradient Descent

GLUE General Language Understanding Evaluation

HTML HyperText Markup Language

INAH Instututo Nacional de Antropología e Historia

INALI Instituto Nacional de Lenguas Indigenas

INEGI Instituto Nacional de Estadística, Geografía e Informática

IPA International Phonetic Alphabet

LLMs Large Language Models

LRLs Low Resource Languages

MBGD Mini-Batch Gradient Descent

MC Multiple-Classification

ML Machine Learning

MLM Masked Language Modeling

NLP Natural Language Processing

NSP Next Sentece Prediction

OCR Optical Character Recognition

OOV Out-of-Vocabulary

PDF Portable Document Format

PLL Pseudolog-likelihood

PPPL Pseudo-perplexity

QA Question-Answering

QQP Quora Question Pairs

RegEx Regular Expression

ReLU Rectified Linear Unit

RoBERTa Robustly Optimized BERT Pretraining Approach

RTL Right-To-Left

SVO Subject Verb Object

tanh hyperbolic tangent

TS Token Similarity

UIIM Universidad Intercultural Indígena de Michoacán

UMSNH Universidad Michoacana de San Nicolás de Hidalgo

UNAM Universidad Nacional Autónoma de México

URL Uniform Resource Locators

Chapter 1

Introduction

1.1 Problem Statement

The recent growth in the Natural Language Processing (NLP) field has been impressive, with advances in deep learning models such as transformers, which have greatly improved language understanding and generative tasks. However, growth has focused on high-resource languages such as English, Mandarin, Hindi, and Spanish which benefit from large public-available datasets and significant research attention [Khatter22]. Unfortunately, this focus has left out languages with low representation, such as many indigenous languages. These languages often lack comprehensive and accessible datasets for training effective NLP models [Tonja23a]. Additionally, the limited number of speakers and the lack of digital resources pose significant challenges to the development of NLP tools that focus directly on these languages.

The marginalization of Low Resource Languages in the digital age risks diminishing their cultural and linguistic heritage, making it crucial for the NLP community to prioritize inclusion, diffusion, and preservation by developing methods and resources that support a wider range of languages. Collaborative efforts involving native speakers, linguists, and technologists are needed to bridge this gap and foster the growth of NLP applications that benefit all languages. According to Fondo para el Desarrollo de los pueblos Indígenas de

América Latina y Caribe (FILAC), in Latin America, 550 indigenous languages survive, with Mexico having the second highest number with 68 registered indigenous languages [FILAC20], although only 22 of them are represented in NLP research [Tonja24].

Our research will focus on the native language *Purépecha* of the state of Michoacán. The project aims to create the first publicly available monolinguistic corpus of the Purépecha language, which will serve as a resource for linguists, researchers, and language preservationists. The corpus will be developed by collecting written texts in Purépecha.

Once the corpus is constructed, we will perform evaluations using Natural Language Processing techniques. Such as training a state-of-the-art deep learning NLP transformer model (RoBERTa) and an n-gram based approach. And finally, evaluate the performance of the text models on Masked Language Modeling (MLM).

The corpus is expected to help researchers gain in-depth insight into the linguistic structure of Purépecha, preservation, and diffusion and contribute to the development of more effective and inclusive language processing tools. The results of these evaluations will not only improve our understanding of the Purépecha language but also pave the way for future research and technological applications, such as automatic machine translation and multilingual tokenization.

1.2 Background

The field of computational linguistics and corpora creation is fairly new; the first computational corpus published was the Brown corpus in 1964 [Francis64], with a size of approximately 1 million words. However, other related work, such as lexicons and dictionaries were developed centuries ago [Reppen10].

The first written works and lexicons developed for the Purépecha language were made in the XVI century thanks to the work of friars like Jerónimo de Alcalá (between 1539 and 1541) with his work "La Relación de Michoacán" and Maturino Gilberti (1558 - 1559) with his work "Arte de la lengua de Michuacan" and "Vocabulario en lengua de Mechuacan" [Blázquez15]. According to the previously mentioned work and Benjamin Warren[Warren00], these titles may also be the first texts developed directly for this language.

In recent years, due to the achievements in Natural Language Processing and Deep Learning (DL), there is a blooming in the field. The recent release of Large Language Models (LLMs) such as Open AI's GPT, Facebook's LLAMA, and Google Gemini, has made huge advancements in the field like summarization, text-prediction, text-generation, sentiment analysis, named entity recognition, etc. Deep Learning Models have shown great scores and performance in different benchmarks, as shown in [Torfi21]. However, all these models need to be trained with huge datasets. A significant issue facing low-resource languages is that, despite having rich oral traditions, they often lack written documentation or are not easily accessible [Tonja24]. Indigenous languages exemplify this challenge as shown in [Ward18] and [Stap23]. Fortunately, there are some NLP conferences, workshops, and chapters showing great interest in the field, such as AmericasNLP, the Annual Meeting of the Association for Computational Linguistics (ACL), the European Chapter of the Association for Computational Linguistics (EACL), the Empirical Methods in Natural Language Processing (EMNLP), among others. The efforts of these individuals and collectives are greatly appreciated. A more in-depth analysis of the achievements and number of publications can be consulted at [Tonja24].

1.3 Thesis Objectives

1.3.1 General Objectives

This thesis aims to build the first publicly accessible monolingual corpus for the Purépecha language, thus facilitating the testing and training of various language models such as n-grams and Transformers.

1.3.2 Specific Objectives

In order to successfully complete the general objectives, a set of particular goals is defined in the following list.

- Locate text documents between different sources.
- Perform data cleansing of the corpus, such as deletion of duplicate text and normalization of text.
- Get insights of the corpus such as Zipf's law, sentence length, and common bigrams.
- Generate the corresponding data sets for training, validation, and evaluation.
- Train and evaluate text models with their corresponding splits.
- Compare the performance of the different text models.

1.4 Chapter Descriptions

This thesis is organized into the following five chapters. Chapter 1 serves as an introduction, presenting the problem statement, a general background on NLP, its relevance to low-resource languages, and the general and specific objectives of the study. Chapter 2 explores key concepts, relevant literature, and prior research in related fields, including language models, text preprocessing, deep learning, n-grams, datasets, and web scraping. Chapter 3 describes the methodology used to create a corpus for the Purépecha language. It includes a concise introduction to the distinctive linguistic features of Purépecha, corpus analysis, a comparison with corpora of other indigenous languages, and relevant statistics such as word frequency and Zipf's law. Chapter 4 outlines the experimental setup and methodology used to compare language models trained on the curated dataset. Finally, Chapter 5 concludes the thesis with a summary of findings, a discussion of limitations, and suggestions for future work.

Chapter 2

Theoretical Framework

2.1 Deep Learning

Deep Learning (DL) is a subfield of Machine Learning (ML), where it is intended to recreate the learning process of the biological brain. In a similar fashion, the connections formed by neurons are replicated by an ensemble of artificial neurons, these are used as the basic building blocks in order to create a weave or net. Similarly to the way a weave is created, these nets are organized by layers that share connections between them. DL algorithms are used in a plethora of applications such as text translation, object recognition in images, stock forecasting, text generation, emotion categorization, and speech synthesis [Russell22].

2.1.1 Artificial Neuron

As stated previously, the Artificial Neuron (AN) is used as the building block of neural networks. In its core, an AN performs the sum of the multiplication of its inputs by some weights and a bias term [Jurafsky24]. It is important to detail that the bias term will always be 1 in order to later shift the activation function curve left or right, much like the intercept term in a linear equation shifts the line up or down [Russell22]. The weighted sum is represented as in Equation 2.1.

$$z = b + \sum_i w_i x_i \tag{2.1}$$

Equation 2.1: weighted sum z, inputs $x_1, \ldots, x_n$, weights $w_1, \ldots, w_n$, bias b

Another representation of the weighted sum performed by an AN using vectors is shown also in Equation 2.2

$$z = \mathbf{w} \cdot \mathbf{x} + b \tag{2.2}$$

Equation 2.2: input vector $\mathbf{x}$, weight vector $\mathbf{w}$, scalar bias term b

Activation Function

As described in the V7 Manual on Activation Functions, the activation function inside an Artificial Neuron will decide whether or not it should fire, and map its output between a certain threshold. The activation function also provides a way to learn more complex relationships within the training data [V721]. According to Dubey's survey and benchmark about activation functions, the most common and popular activation functions are: Sigmoid, Tanh, and ReLU [Dubey22]. These activation functions are also shown in Figure 2.1.

The sigmoid function is represented with σ and the output of the function is determined as detailed in Equation 2.3.

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \sigma(z) = \frac{1}{1 + e^{-(z)}} \tag{2.3}$$

The hyperbolic tangent (tanh) function is similar to the sigmoid, yet its values ranges from -1 to 1 and is detailed in Equation 2.4.

$$y = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.4}$$

Finally, the Rectified Linear Unit (ReLU) activation function simply clamps all values to 0 if they are less than or equal to 0 and leaves the z values as positive otherwise, as shown in Equation 2.5.

$$y = ReLU(z) = \max(z, 0) \tag{2.5}$$

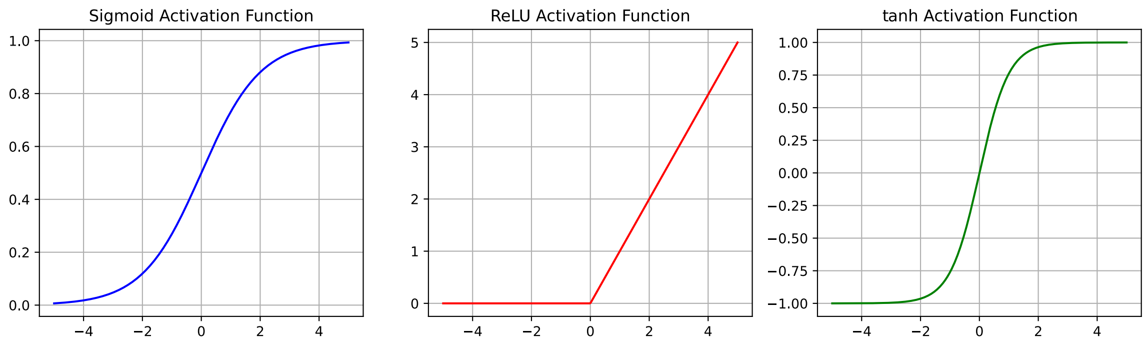

Figure 2.1: Most common Activation Functions [Dubey22].

The complete process of an artificial neuron, from its input, weighted summation, and the output of the activation function, is graphically detailed in Figure 2.2.

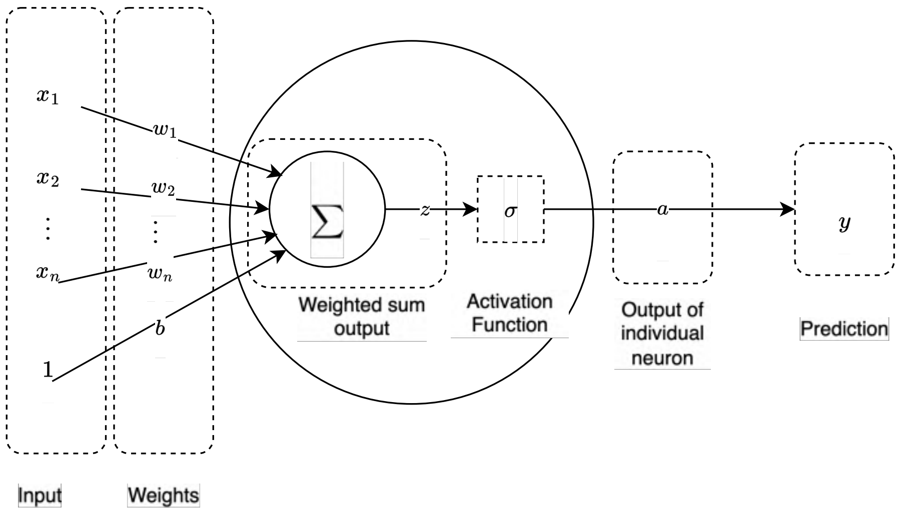

Figure 2.2: A single neuron processing the inputs $x_1, \ldots, x_n$, weights $w_1, \ldots, w_n$ and bias term b generating the output z that subsequently feeds the activation function σ producing the single neuron output a. This output can be inferred as the prediction y

2.1.2 Artificial Neural Network

The Artificial Neural Network (ANN) uses the neuron as the basic building block, taking advantage of its processing and feeding its output to other artificial neurons. These stacks or nets are usually referred to as layers. Information flows from the input layer to the output layer and typically through several intermediate layers called hidden layers.

Feedforward Neural Network

The Feedforward Neural Network (FFNN) is used as the standard architecture. In the FFNN each unit of each layer takes the output of previous layers as the input. This architecture is also known as fully connected and is shown in Figure 2.3.

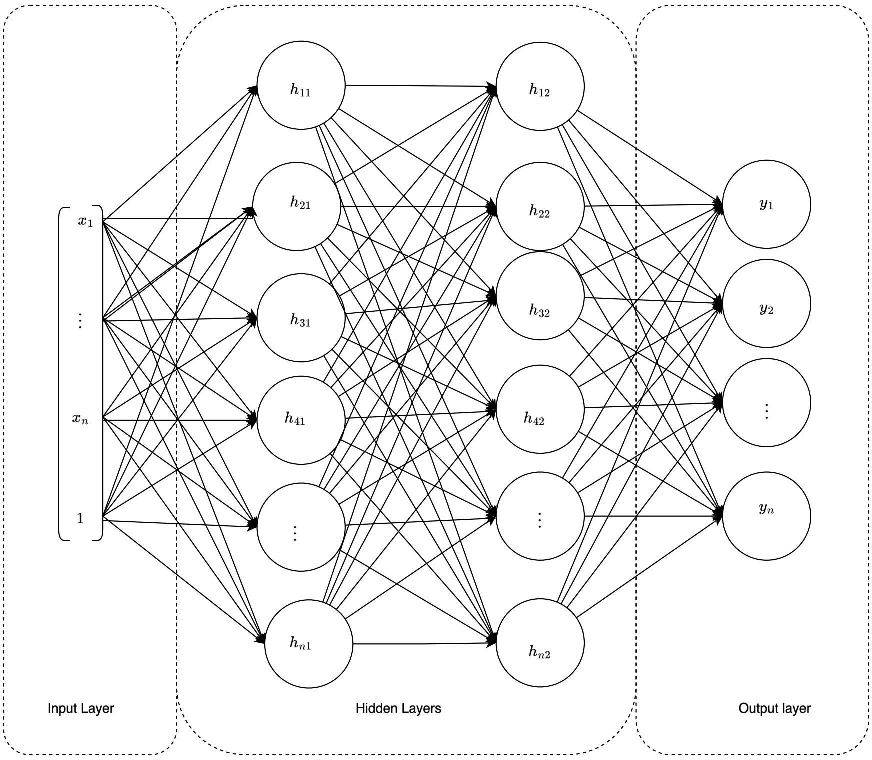

Figure 2.3: A fully connected Feedforward Neural Network displaying how the input layer receives parameters that then feeds the hidden layers.

The process is extended from the main concept of an Artificial Neuron with the following key changes. The weights of each neuron inside a layer are stored inside the weight matrix W, the inputs and bias are processed as a vector as described in Equation 2.6

$$\mathbf{h} = \boldsymbol{\sigma}(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.6}$$

Equation 2.6: hidden layer output vector $\mathbf{h}$, input vector $\mathbf{x}$, weight matrix $\mathbf{W}$, activation function $\boldsymbol{\sigma}$ (applied element-wise $\boldsymbol{\sigma}[z_1, z_2, z_3] = [\sigma(z_1), \sigma(z_1), \sigma(z_2)]$)

It is common practice that the output of FFNN does not include a bias vector [Jurafsky24]. This output is stored in another weight matrix $\mathbf{U}$ that is multiplied by its input vector $\mathbf{h}$ as defined in Equation 2.7 and as shown graphically in Figure 2.4

$$\mathbf{z} = \mathbf{U}\mathbf{h} \tag{2.7}$$

Equation 2.7: output $\mathbf{z}$, input vector $\mathbf{h}$, weight matrix $\mathbf{U}$.

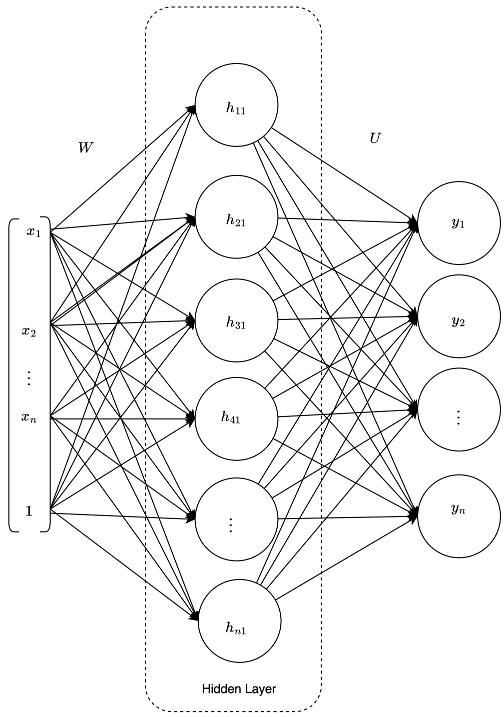

Figure 2.4: It is worth noticing how $\mathbf{W}$ includes a bias term, while U does not.

Softmax Activation Function

In order to map classes to the inputs of the network, a vector of probabilities is required to assess how well the output contains characteristics of each class; however, the output $\mathbf{z}$ contains a vector of real numbers. The softmax function transforms the input values to positive by taking the exponential of each element and then normalizes them by dividing by the sum of all the exponential values for all the number of classes in the multiclass classifier, as shown in Equation 2.8 [Sin24].

$$sofmax(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}} \tag{2.8}$$

Equation 2.8: e^{z_i} exponential function for the input vector, e^{z_i} exponential function for the output vector, d number of classes in the multiclass classifier

Finally, the output of the FFNN can be represented as shown in Equation 2.9.

$$\hat{\mathbf{y}} = softmax(\mathbf{z}) \tag{2.9}$$

2.1.3 Training

As introduced previously, ANN have links or connections between layers and the weights of these links are calculated during network training. Usually, weight values are randomly initialized in order to avoid identical weight updates and longer training times [Hug24b]. In the training phase, the values within the weight matrix are calculated.

Loss Function

A loss function is needed to measure the difference between the prediction of the ANN $\hat{y}$ and the ground truth y [Jurafsky24]. The cross-entropy loss function is used for multiclass classification [Ravichandiran21].

$$f_{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -(\hat{y_i}) \tag{2.10}$$

Equation 2.10: where i being the correct class assigned by the model.

For tasks such as those faced in NLP, where it is common that the correct class is mapped to the probability of a certain word between a dictionary assigned by the model. Equation 2.11 extends the same principle as Equation 2.10 using a context window for tokens.

$$f_{CE} = -\log P(w_i | w_{i-n+1}, .., w_{i-1}) \tag{2.11}$$

Equation 2.11: where w_i being the correct token assigned by the model.

Backpropagation

The algorithm of Backpropagation (BP) was proposed by Werbos [Werbos82] as a technique to reduce the difference between the predicted output of the model and the actual output by adjusting the weights and biases in the network . The BP algorithm is divided primarily into two steps, the forward and backward passes.

In the forward pass, values flow sequentially through the hidden layers, with each layer's output serving as input to the next one. The backward pass then is used to compute the derivatives for the weights and bias update. The derivatives in the backward pass are calculated using the chain rule. The output of a node y_n can be modeled by how much a previous node h_n affects it [Jurafsky24]. The general process of forward and backward passes in the BP algorithm is shown in Figure 2.5.

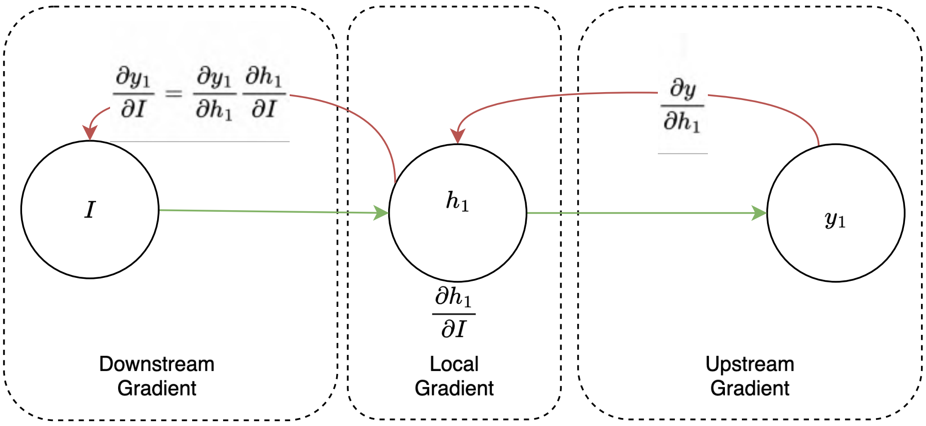

Figure 2.5: In green the information flow for the forward pass. In red the backward pass. The node h_1 takes the upstream gradient from y_1 and multiplies it by its local gradient in order to perform the chain rule and calculate the downstream gradient. A node can have multiple local gradients if it is connected to more nodes.

Gradient Descent

The Gradient Descent (GD) is an iterative method that is used to find the values of variables that minimize a target function.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \eta \nabla f(\mathbf{x}) \tag{2.12}$$

In DL and ML, GD is used to find the values of θ that minimize the error of the loss function $f(\theta)$ over the complete training data set.

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \nabla f_{\mathrm{CE}}(\boldsymbol{\theta}^k) \tag{2.13}$$

The learning rate η should be a small value between 1×10^{-4} and 1×10^{-6} to prevent overshooting the minimum or unnecessarily prolonging training time [Hug24a]. Although a too small value for η may descend smoothly, it may not find the optimal values. The previous situations are shown in Figure 2.6.

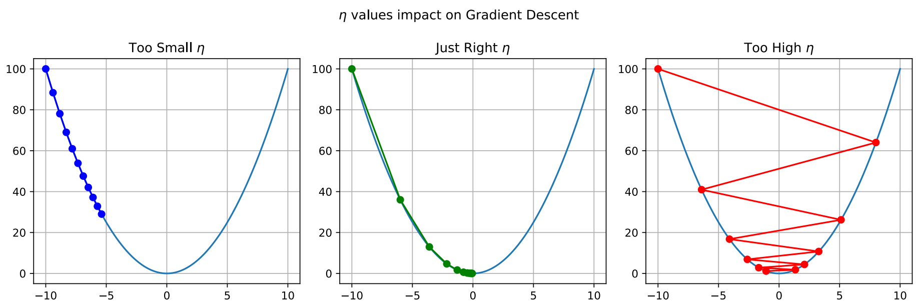

Figure 2.6: Learning rates ands their impact on GD

Mini-Batch Gradient Descent

Mini-Batch Gradient Descent (MBGD) divides the training data into small subsets called mini-batches, allowing for more frequent parameter updates and is useful for training sets that are too large to store entirely in memory [IBM24].

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \frac{\eta}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} f_{\mathrm{CE}}(\boldsymbol{\theta}; \hat{y}_i, y_i) \tag{2.14}$$

Equation 2.14: where m is the mini-batch size, $\hat{y}_i$ the correct class or word at the i-th data point in the mini-batch, η learning rate

2.2 Datasets

A dataset is an assembly of data. It is common for a dataset to be constructed by joining different database tables, with each table column representing a specific variable and each row representing a particular record of that dataset. Datasets are used in machine learning and deep learning to fit models to classify or forecast data [Raschka15].

Corpus

A corpus is a type of text dataset consisting of a collection of linguistic resources such as documents, textbooks, newspapers, websites, and research papers. Corpus may also include transcriptions of natural speech, such as informal face-to-face conversations, lectures, meetings, or interviews [Atkins92].

Web Scraping

This method, also referred to as "Web crawling" or "Web harvesting" is frequently employed in the collection of data, including corpora creation [Mitchell15]. Typically, a web scraping bot makes use of a website's semantic structure, as this automated agent searches for specific content within an HyperText Markup Language (HTML) document. The data saved is defined by the main objective of the scraping bot, such as the text contained inside the paragraph tag *<p>*, other hyperlinks located within the anchor tag *<a>*, or the graphic resources inside the image tag *<img>*.

Python libraries like BeautifulSoup [Richardson07], Scrapy [Kouzis-Loukas16], and Selenium [Sharma14] are popular for this task and utilize a similar approach [Mitchell15]. First, make a GET request and store the response in a variable, then check the status code of the previous request that may be typically 200 for success, 404 if the file is not found in web server, or 505 if there is an internal server error. If the request was successful, the HTML content of the response can be prepared and stored in another variable. Finally, the desired elements can be filtered by HTML tags or attributes such as classes or id.

2.3 Natural Language Processing

Natural Language Processing (NLP), a subfield of Machine Learning and linguistics, aims to provide computational models of various linguistic phenomena by understanding not only single words but also their context [Hug24a], and is widely used in day-to-day tasks such as speech recognition systems, text-to-speech synthesis, automated voice response systems, web search engines, and text editors [Fruehwald22a].

2.4 Language Models

A language model is defined as a probability distribution that describes the likelihood of a given string. Language models can also make grammar or spelling suggestions, compute the most probable translation of a text, summarize text or calculate the answer of a question [Jurafsky24].

2.4.1 Text Preprocessing

Text Preprocessing is refereed as a group of actions performed on text to improve the quality of models or text analysis techniques. The most common text-processing techniques can be listed as tokenization, stopword removal, resolving contractions, deleting duplicated text, correcting spelling mistakes, expanding abbreviations, text normalization, and mapping out-of-vocabulary words to in-vocabulary standard words [Chai23, Zhang13].

Tokenization

In NLP, tokenization is the process of breaking down the information present in a document into smaller segments [Kasper Welbers17]. These smaller segments are called tokens. They can be: paragraphs, word sequences (n-grams), complete words (space separated tokens), word subsequences (character pairs, characters), characters, and punctuation marks [Manning08].

A piece of software that manages the input text and splits it in order to feed a model is called a tokenizer. The tokenizer maps the text input to known entries inside a dictionary or vocabulary, changing the input from an alphanumeric list to a number list or vector. The vocabulary size is defined by the total number of independent tokens present in the corpus. A comparison between different tokenizing strategies is shown in Figure 2.7.

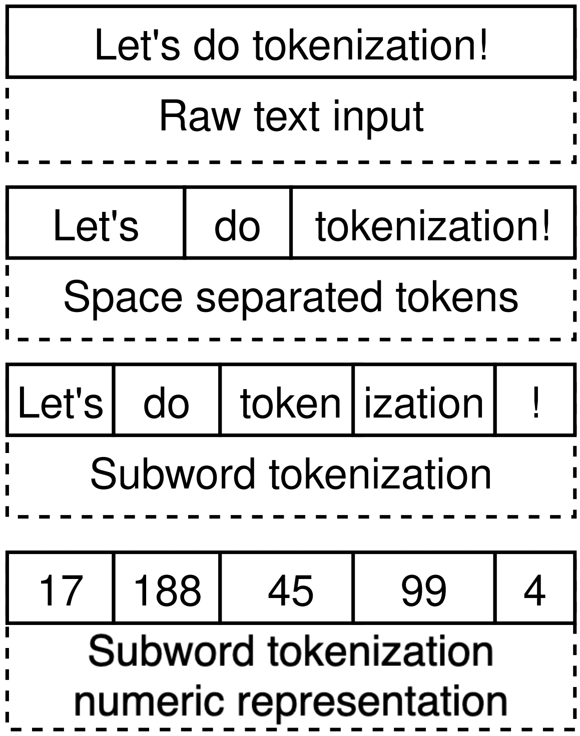

Figure 2.7: Different tokenization aproaches for a given input

A tokenizer may also have the capacity to perform the previously mentioned task in reverse. In other words, a tokenizer may translate the numeric representation back into human-readable text. Subword tokenizers such as wordpiece were introduced in Google's neural machine translation work [Wu16] and Byte-level Byte Pair Encoding (Byte-level BPE) showing great performance with agglutinate languages in [Wang19].

2.4.2 N-grams

In the field of natural text processing, n-grams refer to the tasks of generating word sequences that appear naturally in texts. These are classified into: monogram (single words), bigrams (couple of words), trigrams (triplets of words), from here any higher number is called n-gram, which are n length of consecutive words. N-grams typically have a target word and a given context. For a bigram $w_{i-1}w_i$, its probability can be calculated as described in Equation 2.15 [Manning99]

$$P(w_i|w_{i-1}) = \frac{P(w_i w_{i-1})}{P(w_{i-1})} \tag{2.15}$$

For higher order n-grams, the probability is calculated as described in Equation 2.16.

$$P(w_i|w_{i-n+1:i-1}) = \frac{Counts(w_{i-n+1:i-1}w_i)}{Counts(w_{i-n+1:i-1})} \tag{2.16}$$

Equation 2.16: n n-gram order

A limitation of the prior approach is that while feasible, certain n-gram combinations may never occur in any corpus, resulting in a very low probability [Fruehwald22a].

Smoothing

While training n-gram models, some token pairs may have a high frequency, therefore its probability can be calculated without any issues. However, there is the probability that the models will face tokens that were not present during the training phase, and its probability would be zero. Solutions like cut-off, or Out-of-Vocabulary (OOV), manage low-frequency or unseen tokens by replacing them with the *<unk>* token. Another smoothing alternative is a Back-off model, where it is normally calculated the n-gram count, and for sequences with low or zero counts, the model reverts to a lower order n-gram [Russell22].

2.4.3 Transformer Based

Introduced in Vaswani et al.'s research paper [Vaswani23], the transformer architecture is a kind of neural network that focuses on the attention mechanism rather than recurrence and utilizes a stack of encoder blocks, each comprising a self-attention mechanism and a FFNN; the encoder can be stacked N times as required, with the research paper using a stack of 8 encoder blocks.

2.4.4 Encoder

The encoder serves as the building blocks for the transformer model and each encoder block is composed of three main features, multi-head attention, add-and-norm, and feedforward. These components are graphically described in Figure 2.8.

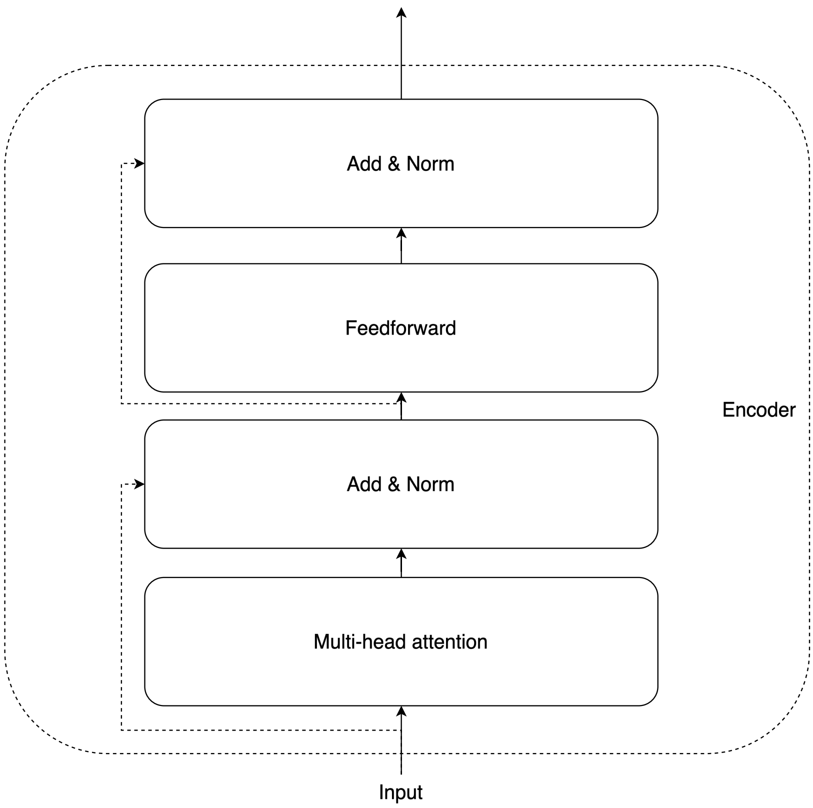

Figure 2.8: Encoder's main components

Self-attention

In summary, the self-attention mechanism processes a given sentence to compute the representation of each word, calculating the relationship between each word and all other words within the sentence. Figure 2.9 shows the information flow of a complete input and how it is processed in parallel by a casual or Right-To-Left (RTL) transformer layer.

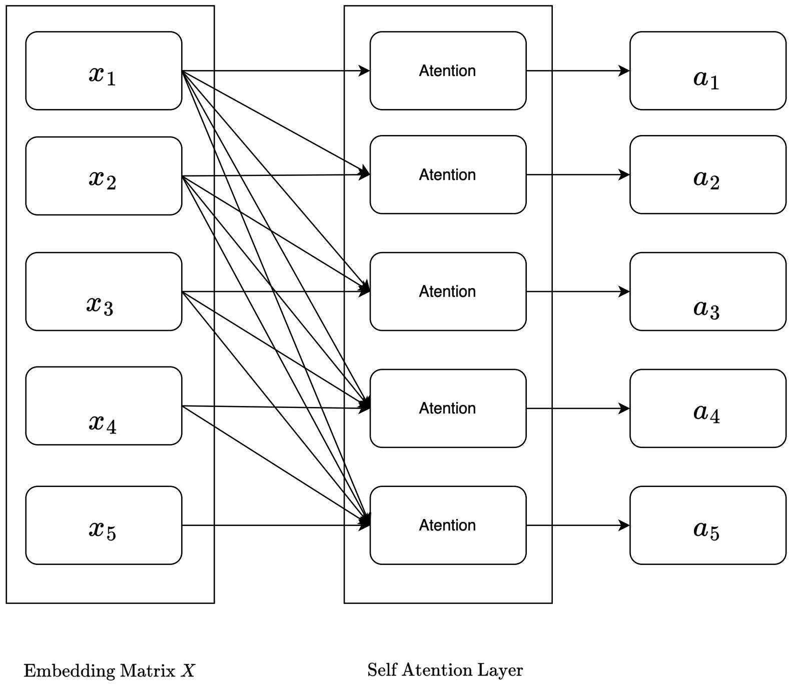

Figure 2.9: The information flow for a causal or right to left self-attention layer.

As detailed by Ravichandiran each of the inputs x is adjusted by the attention layer and then assigned to a output of the same size a. The input matrix $\mathbf{X}$ was previously tokenized and contains the vector representation of the words. The input matrix is sized by the length of the sentence and the embedding dimension used by the model (usually 512) [Ravichandiran21].

In order for the attention mechanism to calculate the weight or relationship between the tokens, the following matrices are required: the Key matrix $\mathbf{K}$, Query matrix $\mathbf{Q}$, Value matrix $\mathbf{V}$, Attention matrix $\mathbf{Z}$, and the weight matrices$\mathbf{W}^K$, $\mathbf{W}^Q$, $\mathbf{W}^V$. The weight matrices are randomly initialized, and their values are optimized during the model training phase.

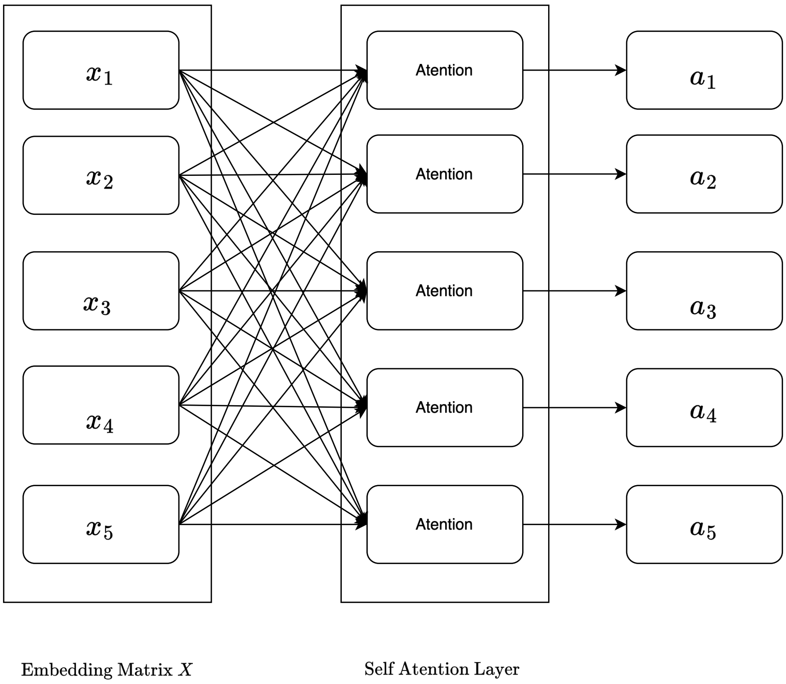

Figure 2.10: The information flow for a bi-directional self-attention layer.

The key, query and value matrices are calculated by multiplying their respective weight matrices $\mathbf{W}$ by embedding matrix $\mathbf{X}$ as follows:

$$\mathbf{K} = \mathbf{X}\mathbf{W}^K$$
$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}^V \tag{2.17}$$

In Equation 2.17, every row within the Key, Query, and Value matrices corresponds to the value, query, and value vectors associated with each token in the input sentence. These matrices will later be used to compute the relationship between tokens by multiplying the Query and Key matrices, as shown in Equation 2.18.

$$\mathbf{Q}\mathbf{K}^T \tag{2.18}$$

The $\mathbf{Q}\mathbf{K}^T$ matrix comprises the dot products of query vectors with key vectors, effectively representing the token similarities. However, these values need to be scaled in order to obtain stable gradients and avoid gradient exploding [Ravichandiran21]. This scaling is performed by dividing by the root of the dimension of the key vector $d_{\mathbf{k}}$ and normalize the values using the softmax function, as shown in Equation 2.19.

$$softmax(\frac{\mathbf{QK}^T}{\sqrt{d_{\mathbf{k}}}}) \tag{2.19}$$

Finally, the Attention matrix, $\mathbf{Z}$ can be computed by multiplying the score matrix $softmax(\frac{\mathbf{QK}^T}{\sqrt{d_{\mathbf{k}}}})$ by the Value matrix $\mathbf{V}$.

$$\mathbf{Z} = softmax\left(\frac{\mathbf{QK}^T}{\sqrt{d_{\mathbf{k}}}}\right)\mathbf{V} \tag{2.20}$$

The process of a single self-attention mechanism is graphically detailed in Figure 2.11. However, transformer models use several attention heads, also known as multi-head attention mechanism.

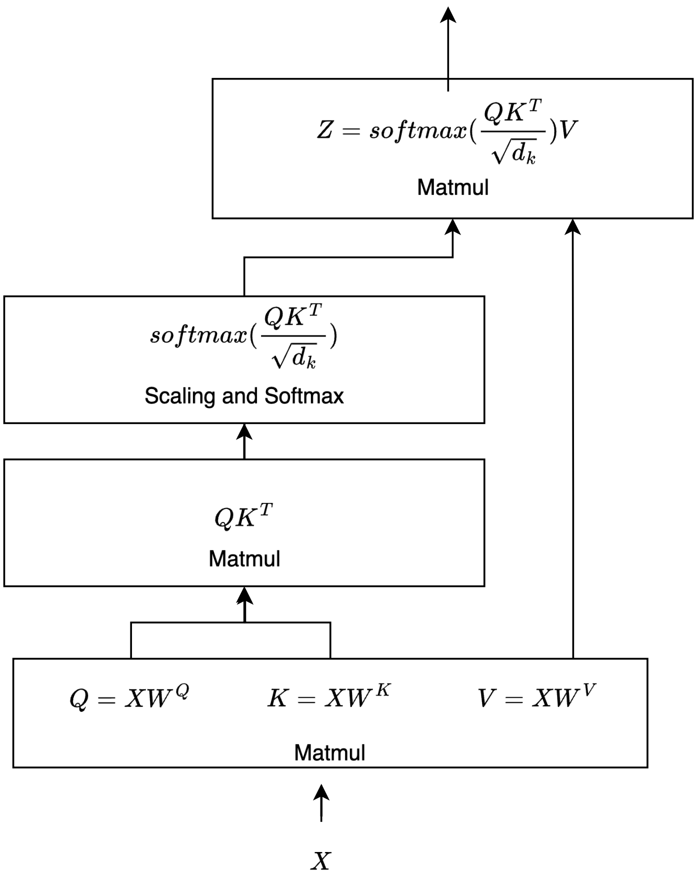

Figure 2.11: Self-attention operation blocks.

Multi-head attention

The multi-head attention mechanism utilizes multiple attention matrices **Z**, with the underlying idea that each head could focus on the context for various objectives [Jurafsky24]. Each head may specialize in capturing various linguistic relationships between context components and the current token, or in identifying specific patterns within the context window. A diagram of a multi-head attention is shown in Figure 2.12 where the concatenation block for the attention matrices is present within the last steps.

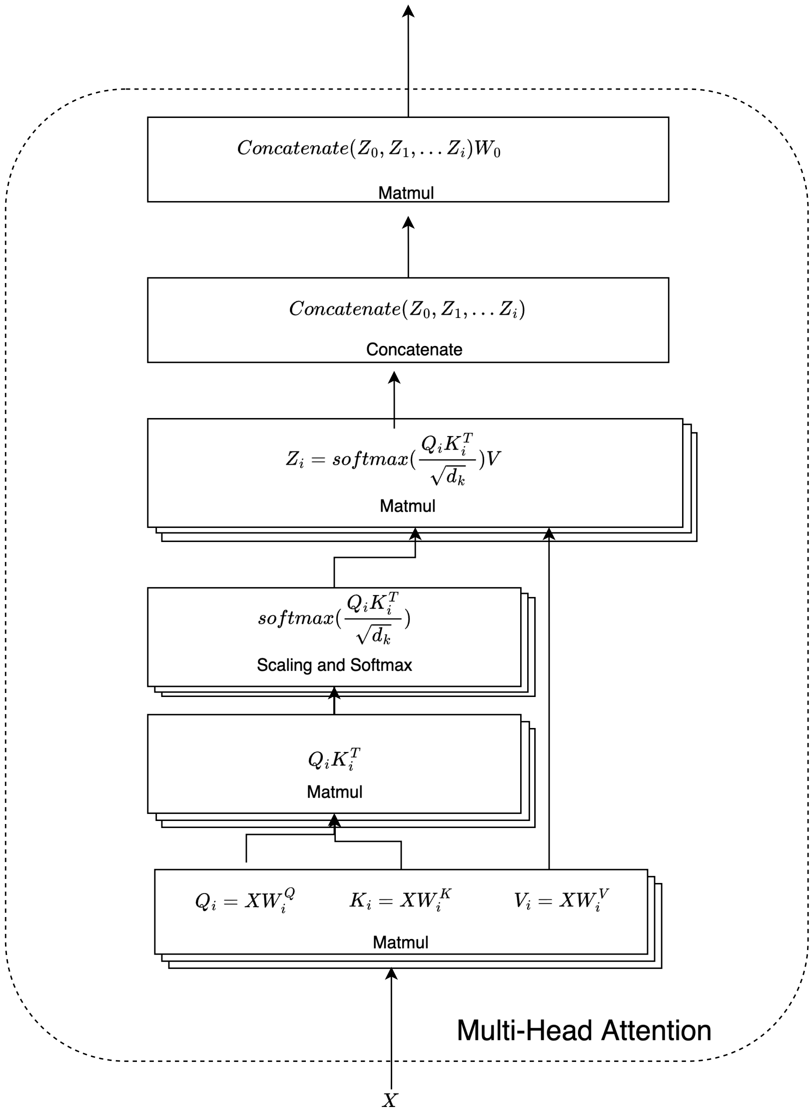

Figure 2.12: Multi-head-attention block and its respective operation blocks.

Feedforward

The Feedforward block inside the encoder consists of a FFNN with Rectified Linear Unit (ReLU) activation functions. BERT and its derivatives use two dense layers in the feedforward step and their parameters are shared between encoder blocks [Ravichandiran21].

Add and Norm

As previously shown in Figure 2.8 the Add and Norm block is present twice in the encoder block. This block connects the input **X** to the multi-head attention output. Then it connects the input of the FFNN to its own output. The Add and Norm block works as a residual connection followed by layer normalization. According to various authors, layer normalization facilitates quicker training by stabilizing the values within each layer, preventing large fluctuations.

BERT

The Bidirectional Encoder Representation from Transformers (BERT) model was introduced by Jacob Devlin et al. in [Devlin19]. BERT's main feature is that it is focused on a deep bidirectional representation of text by jointly conditioning both left and right contexts between all layers. This model was trained with Masked Language Modeling (MLM) and Next Sentece Prediction (NSP) approaches. The MLM phase works by randomly selecting a percentage of text (recommended about 15%) in order to predict those masked tokens; the NSP phase will randomly select sentence pairs, and the model will evaluate the probability that these sentences are presented one after the other. A diagram of the BERTs architecture displaying the encoder blocks is present in Figure 2.13.

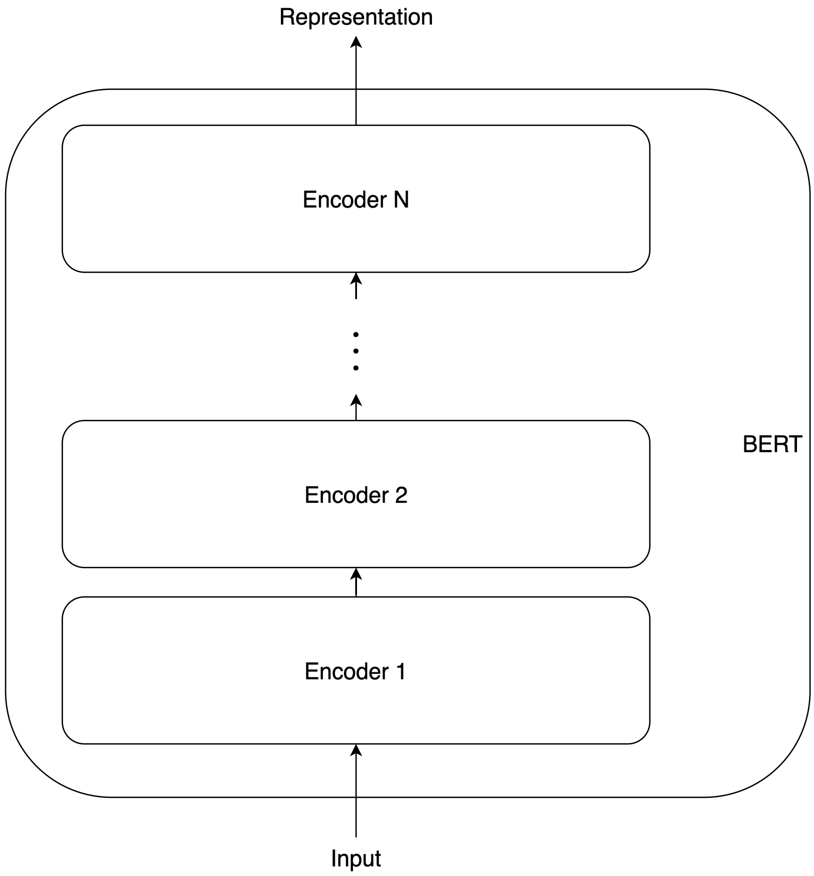

Figure 2.13: A Stack of N encoder blocks in BERTs architecture.

RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa) was introduced by Lui et al. in [Liu19] with a slightly different approach at the pretraining stage than BERT. One of RoBERTa's main differences is removing the NSP and focusing on the MLM, changing the tokenizer from wordpiece to Byte-level Byte Pair Encoding [Ravichandiran21].

2.5 NLP for Low Resources Languages

NLP research plays an important role in advancing multilingualism and preserving linguistic diversity worldwide. However, only a handful of the more than 7000 languages around the world are featured in recent NLP research [Joshi20], and indigenous languages face much less representation [Tonja24].

Low Resource Languages (LRLs) refers to tongues that receive less academic focus, have limited resources, lack computational tools, are underprivileged, are less frequently taught, or have a low number of active speakers [Magueresse20, Tsvetkov17]. Incorporating LRLs into NLP research can help to assess the generalization of NLP models, democratize their use, and contribute to the conservation and study of those languages [Tonja23a]. Using the previously mentioned problem, there is a growing interest in NLP conferences and workshops, such as AmericasNLP, the annual meetings of ACL and EACL, and EMNLP [Tonja24], and even some open competitions such as Sarcasm Identification for Dravidian Languages [Chakravarthi23].

In México there are 68 indigenous languages recognized by Instituto Nacional de Lenguas Indigenas (INALI) [INALI09], however, only a few corpora are available for indigenous languages such as the Universidad Nacional Autónoma de México (UNAM) Nahuatl corpus [Gutierrez15], the Mazatec-Mixtec Parallel Corpus [Tonja23b] and the Large Mayan Corpus V [Lou24].

2.6 Final Remarks

This chapter discusses the efforts of ML to replicate the biological neuron through the perceptron and how creating layered networks, interconnections between layers and the impact of activation functions such as: Sigmoid, ReLU, tanh are the main study fields of DL. It is also disscused the training process including the forward and backward pass, how optimization algorithms like MBGD and GD iteratively change internal paramaters of the network in order to minimize the error and reduce training times.

In NLP, datasets and corpora are vital for training models. Data preprocessing techniques like tokenization and text normalization improve model performance, aiding in grammar correction and translation. DL shows that Transformer architectures, using attention mechanisms, outperform previous models by effectively capturing token relationships in sequences.

Additionally, there is an ongoing effort in NLP research to include Low Resource

Languages, which often lack resources and computational tools. This inclusivity aims to promote multilingualism and preserve linguistic diversity. Notably, in Mexico, indigenous languages are acknowledged, but only a few have available corpora for NLP research.

Chapter 3

Corpus creation

3.1 Purépecha

Purépecha (also P'horé, P'urhépecha, P'hore, or Phorhé) is one of the 68 indigenous languages in Mexico and is spoken mainly in the highlands of Michoacán (among 13 municipalities of 113 present in the state)[INALI09]. It is also present with less representation in some regions of the states of Mexico, Baja California, Guanajuato, Jalisco, and Mexico City. The distribution of speakers across Mexican states is shown in Figure 3.1.

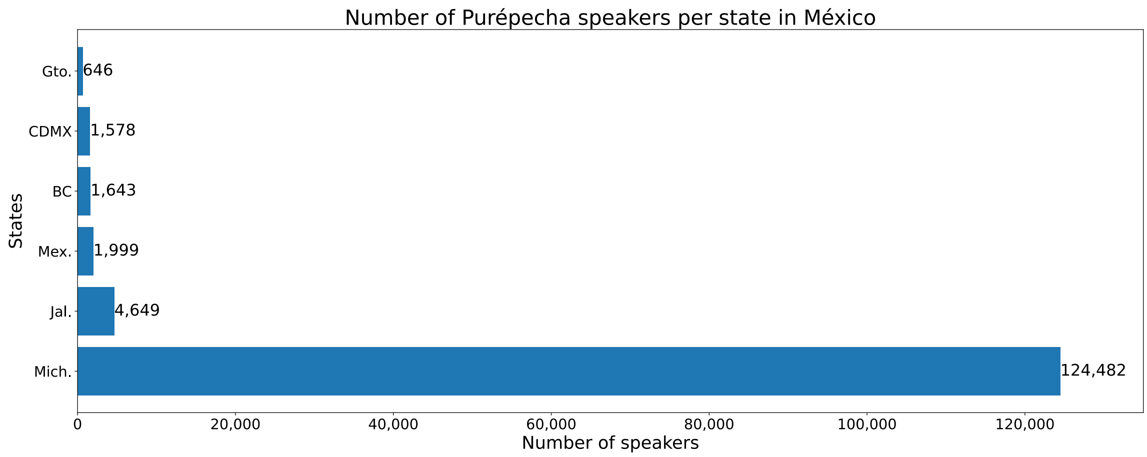

Figure 3.1: Distribution of Purépecha speakers in states with the largest number of speakers in Mexico.

The 2020 Instituto Nacional de Estadística, Geografía e Informática (INEGI) population and housing census estimated a total of 7,364,645 indigenous language speakers in Mexico, the amount of speakers for each registered language can be consulted in Table 3.1. Purépecha is located in the 15^{th} place on the list of native languages spoken in Mexico with a total of 142,459 speakers older than 3 years (92% reside in the state of Michoacán) [INEGI20]. Based on the current number of speakers, the state of preservation and transmission led the Endangered Languages Project (ELP) to tag the language as “Threatened” [Project24], other sources have tagged the language “At risk” [Capistran15]. It is also noted that the language faces an aging process, with 14% of the speakers aged 60 and over; however, it still has a higher percentage of speakers aged 10-19 (23.7%)[INEGI09].

The Purépecha language is classified as an isolated language, without sharing any nexus with other Mesoamerican languages. Some authors have found common words between the Quechua language and Purépecha, however, there is no grammar-wise relationship [Chamoreau23] and Liedtke [Liedtke01] proposes that this alleged vocabulary relationship needs further investigation.

3.1.1 Writing System

Purépecha Alphabet

Since the XVI centuries, the Purépecha language adopted the Latin graphs due to the work of linguistic friars like Maturino Gilberti. Over time, there have been different approaches to normalize the written word for the language, such as including phonetic symbols or trying to adjust the symbols that were present in the mainstream typewriter and printing press. These different approaches had left the language without a universal standard about it. However, in this work, we will try to adhere to the guidelines proposed by United Nations Centro de Cooperación Regional para la Educación de Adultos en América Latina y El Caribe (CREFAL) in 1979 [Alberto Medina20], this rule set is also used in “Guia lingüistica del idioma P’urepecha” and in the Linguistics Department at Universidad Michoacana de San Nicolás de Hidalgo (UMSNH) [Medina Perez06]. It is worth mentioning that some documents may follow the standard established by the Universidad Intercultural Indígena de Michoacán (UIIM), and these documents will be retained in their original form.

Indigenous language (as it appears in the INALI catalogue)	Total speakers
Náhuatl	1,651,958
Maya	774,755
Tseltal	589,144
Tsotsil	550,274
Mixteco	526,593
Zapoteco	490,845
Otomí	298,861
Totonaco	256,344
Ch'ol	254,715
Mazateco	237,212
Huasteco	168,729
Mazahua	153,797
Tlapaneco	147,432
Chinanteco	144,394
Purépecha	**142,459**
Mixe	139,760
Tarahumara	91,554
Zoque	74,018
Tojolabal	66,953
Chontal de Tabasco	60,563
Huichol	60,263
Amuzgo	59,884
Chatino	52,076
Tepehuano del sur	44,386
Mayo	38,507
Popoluca de la Sierra	36,113
Cora	33,226
Triqui	29,545
No especificado	22,777
Yaqui	19,376
Huave	18,827
Popoloca	17,274
Cuicateco	12,961
Pame	11,924
Mam	11,369
Q'anjob'al	10,851
Tepehuano del norte	9,855
Tepehuano del norte	9,855
Tepehua	8,884
Popoluca insuficientemente especificado	8,427
Chontal de Oaxaca	5,613
Sayulteco	4,765
Chuj	3,516
Akateko	2,894
Otras lenguas indígenas de América	2,453
Chichimeco Jonaz	2,364
Tlahuica	2,238
Guarijío	2,139
Chontal insuficientemente especificado	1,704
Q'eqchi'	1,599
Matlatzinca	1,245
Pima	1,037
Chocholteco	847
Lacandón	771
Seri	723
K'iche'	589
Kumiai	495
Jakalteko	481
Texistepequeño	368
Tepehuano insuficientemente especificado	317
Paipai	231
Pápago	203
Ixcateco	195
Cucapá	176
Kaqchikel	169
Qato'k	126
Ixil	117
Teko	78
Oluteco	77
Kiliwa	76
Ayapaneco	71
Kickapoo	63
Awakateko	20
National Total	7,364,645

Table 3.1: Mexican indigenous languages and their amount of speakers [INEGI20]

.

This decision is due to ongoing discussions within the speaking communities and scholars. To accurately represent the current state of the language and prevent any bias, these documents will maintain their existing standard.

The CREFAL and UMSNH propose the following characters for the language:

Vowels

The Purépecha language has the following 6 vowels:

I A U E O Ï

As shown previously, the language has 5 vowels in common with Spanish, plus a sixth, the "ï". This "ï" is not an accented "i", but rather a distinct sound.

Consonants

B CH CH' D J K K' L M N NH P P' R RH S T T' TS TS' X

The letters "B", "D", "NH", "R", and "RH" are not used at the beginning of words. The letter "L" is present only in 2 words "pale" (young man) and "male" (young woman).

Numerals

Purépecha, like the other Mesoamerican cultures, uses a vigesimal numeral system (base-20) which is a key feature proposed by Campbell, Kaufman, and Smith [Campbell86]. This system is also highly used in the calendar, as explained by Joyce Marcus [Marcus00]. The list of numbers from 1 to 20 is present in Table 3.2

Agglutinative Structure

One key characteristic of this language is how it builds words. It is classified as an agglutinative language. This means that in Purépecha a root word is used and then adds suffixes and infixes, each with a different meaning, to form words. Some representative cases of this feature are presented in Table 3.3.

1	ma	11	témbini ka ma
2	tsimani	12	témbini ka tsimani
3	tanimu	13	témbini ka tanimu
4	t'amu	14	témbini ka t'amu
5	iumu	15	témbini ka iumu
6	kuimu	16	témbini ka kuimu
7	iumu tsimani	17	témbini ka iumu tsimani
8	iumu tanimu	18	témbini ka iumu tanimu
9	iumu t'amu	19	témbini ka iumu t'amu
10	témbini	20	ekuatsï

Table 3.2: Purépecha numerals.

Final Word	Root Word	Suffixes & Infixes
meiamukusïngaksï (we thank to you)	meiamu (to thank)	• ku (subject to you) • sïnga (present 1st. person) • ksï (plural ending)
k'uiripiskajtu ([you] have grown)	k'uiripini (to grow)	• ska (past 1st person) • jtu(your)
t'irenkuekasïngani (I want to eat)	t'ireni (to eat)	• kueka (want) • sïnga (present 1st. person) • ni (me)
juchinio (our house)	jucha (ours)	• o (house)

Table 3.3: Example of how larger words are created using suffixes and infixes [Chamoreau23].

Vocabulary

Approximately 70% of the nouns in this language are derived from verbs. In order to transform a verb into a noun, the root word of a verb is selected, the "ni" suffix (that shows that the verb is in infinitive) is removed, and finally, the "kua" suffix (which shows that the word is now a noun) is added [Medina Perez70]. This feature is present in Table 3.4

with some examples. However, not all nouns are created in this way; there are some nouns that are not related to verbs; an example of this can be found in the words for animals, parts of the day, colors, and family members. Some of these nouns are shown below at Table 3.5.

Verb		Noun	
ch'anani	(to play)	ch'anakua	(toy)
janini	(to rain)	janikua	(rain)
pireni	(to sing)	pirekua	(song)

Table 3.4: Nouns created from verbs

Noun	
achaati	(male adult)
ch'anchaki	(donkey)
tsakapu	(stone)
parakata	(butterfly)
tataka	(kid)
charhapiti	(red)
itsï	(water)

Table 3.5: Special nouns not related to verbs.

Word Order

Commonly Purépecha uses a Subject Verb Object (SVO) as a constituent order while building sentences; but other structures may be present depending on the context.

Example Sentence

Imá sapichu jukasïndi ma kajtsïkua auanda jasï.

(The kid is wearing a light blue hat) - SVO order.

3.2 Construction

In the vast and ever-evolving realms of Artificial Intelligence subfields such as Machine Learning, Deep Learning, and Natural Language Processing, datasets are a principal element. These datasets are not merely background actors; they play a leading role inside different stages of these workflows. From training to validation and performance metrics for the multitude of algorithms, datasets play a key element within AI applications [Challa20]. Inside NLP, datasets are also known as corpus. Torruella and Llisterri describe two different types of corpus. The oral corpus that is composed of voice recordings and is mainly focused on studying how a language is used verbally. The textual corpus is composed of written words present in textbooks, journals, novels, poems, etc., and its objective is to represent the language through its use [Torruella99].

As stated previously, a corpus is built with a wide set of documents from different sources, and it should represent the target language all-round. In other words, it should be made up of real data and be as large, complex, and inclusive as possible to display the unique features, variations appropriate to the language, and cultural heritage of the people who speak that language. As Torruella states that a corpus needs to be extensive to capture the primary traits and encompass all aspects of the language being studied, incorporating a wide range of sources to accurately reflect the actual usage of the language [Torruella99].

The construction of a linguistic corpus is already a challenge of its own. Even when dealing with widely available languages such as English and Spanish [Aarts93] [Torruella99]. However, building a corpus for LRLs, such as indigenous languages in the Americas, brings a new set of challenges to be considered. First, almost all indigenous languages in Latin America show great oral tradition, but written resources are lacking or scarce [Tonja24] . Second, as a result of these languages being poorly represented, most of them lack a standardization or a complete set of rules on how words should be written, a great example of this problem with an indigenous language with much more native speakers like Nahuatl is addressed in UNAM-Instututo Nacional de Antropología e Historia (INAH)'s article [Gutierrez15].

For the Purépecha language, consultation to databases focused on linguistic corpora like University of Helsinki's Open Parallel Corpora (OPUS) [Tiedemann12], National Institute for Research in Digital Science and Technology's (INRIA) Open Super-large Crawled Aggregated coRpus (OSCAR) [OSCAR23], UNAM's Corpus Paralelo de Lenguas Mexicanas [UNAM23], and Kaggle Datasets [KAGGLE24], although no publicly parallel or monolinguistic dataset was found.

In order to effectively train and evaluate text models, datasets are fundamentally required. However, as has been emphatically stated earlier, corpora for this particular language regrettably remain inaccessible to the public. With this purpose in mind and in an effort to contribute with documentation, preservation, and language diffusion, the effort to construct the corpora was undertaken. The documents were collected principally from the following domains: religious, legal, and educational. Additional resources also take part of the corpus from textbooks, poems and songbooks, novels, newspapers, etc.

Methodology

While working with LRLs, documents are sparse, there is low penetration of them on the Internet, and books tend to be non-digital, a methodology is proposed targeting the two distinct types of documents: digital and non-digital. This methodology was inspired by Gutierrez-Vazques' research [Gutierrez-Vasques15] and was adapted to meet the specific needs of the project and the Purépecha language. An overview of the process and its steps is visible at the process map in Figure 3.2

The first step in the methodology used is Document Location. As previously stated, documents for the language are sparse, and it is crucial to pinpoint the location of both physical or digital documents. In order to obtain the documents, an on-site search was performed. Visits were scheduled between October and November 2023 to UN CREFAL library, the UIIM linguistics department, Instituto Tecnológico Superior P'urhépecha (ITS P'urhépecha) language department and library, Escuela Normal Indígena de Michoacán library and UMSNH language department, and newspaper archive. These visits granted access to some databases, repositories, and websites has been provided where digital documents were available to download. In the specific case of websites, a Python web scraping script was developed to navigate through web pages and locate paragraphs, articles, and anchor HTML tags. This Python script utilizes the Beautiful Soup library to manage the

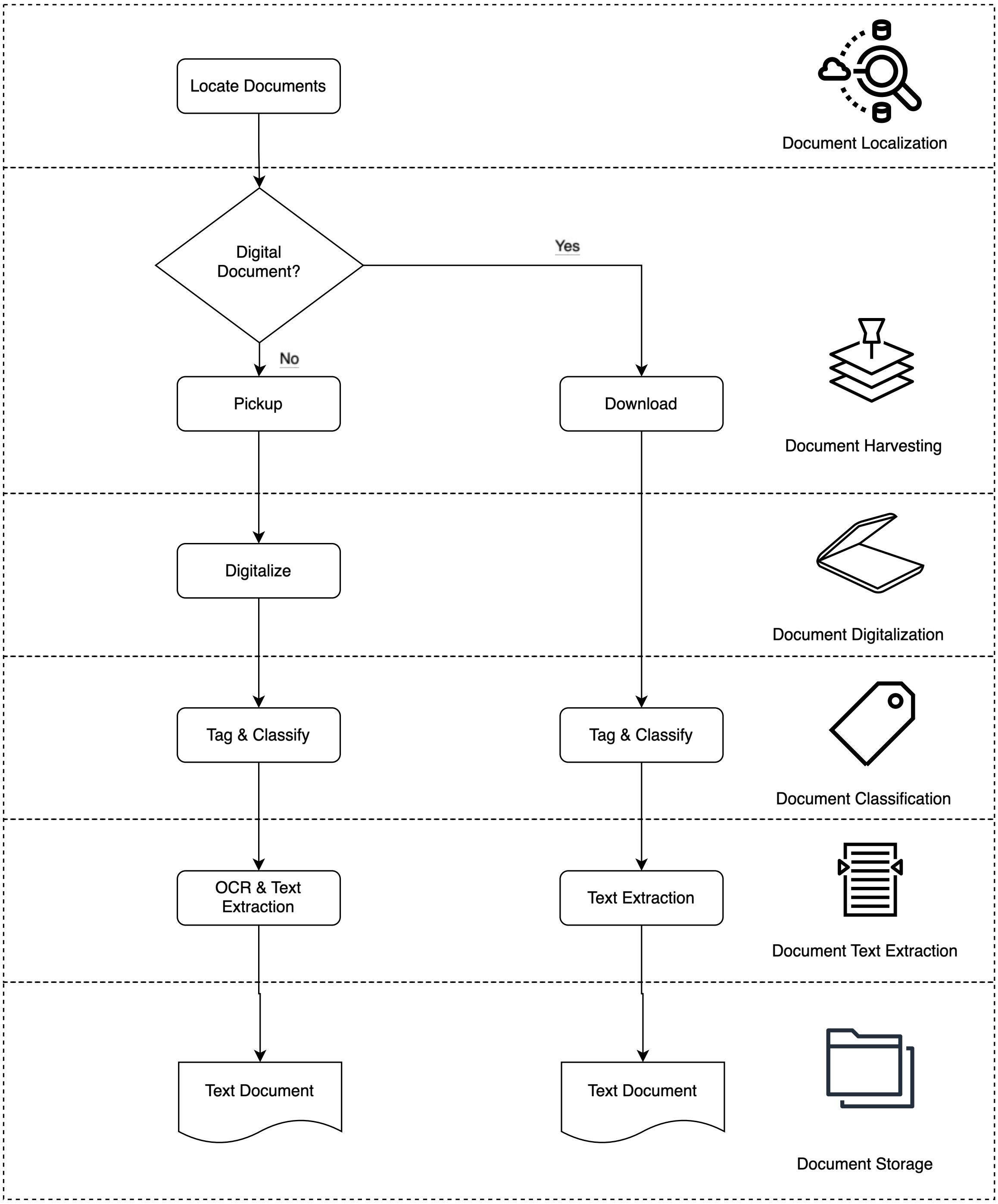

Figure 3.2: Methodology for including text documents in the corpus.

requests and responses from various websites.

After the documents were located, the next step is to carry out the relevant procedures for their loan (on site or remote), or to download files if they are already available digitally. It is worth mentioning that some of CREFAL's books were unique and there were no copies left. For example, the book "Cuentos y Leyendas" was written on a typewriter with no remaining copies left.

The following step only applies to non-digital documents. After the finalization of procedures and the agreement on the loan terms, the documents are subsequently digitized. If the documents were loaned off-site then the scanning was done with a flat Hewlett-Packard scanner and the Vue Scan software. In case that the loan was permitted only within the library itself, the scanning was performed with a cellphone camera. The settings for the scanning followed IBM [IBM21] and Illinois University Library [Uni24] recommendations and best practices for Optical Character Recognition (OCR). These settings are shown in Table 3.6.

Settings	Value
Resolution	300 dpi
File type	PDF
Color Space	Grayscale

Table 3.6: Settings for Portable Document Format (PDF) files to include in the OCR pipeline.

Once documents have been digitized or downloaded, they are classified into one of the following categories based on their content and origin. The content tag has the following categories: Legal, religious, narrative, lyrics, and others. The origin tag contains the sources of the documents. For example, the domain name if the document was gathered using a Web crawler, or the institution name if it was provided by some government institution or university. In addition, in this step, the documents that were digitized are tagged and stored in a different folder for reasons that will be addressed later.

The document text extraction is then done with the help of Googles Tesseract OCR software when dealing with non digital documents. A notable limitation of this method is that the software makes numerous errors when applied to a language that is not included in its training set. These errors involve accents and apostrophes, typewriter fonts, and the software's tendency to adhere to patterns from other languages, leading to the creation of incorrect words. This limitation is also reported by Vilchis at their Nahuatl corpus creation [Gutierrez15]. One possible fix is to manually correct each document after being processed by OCR, however, that approach is far from the time scope of this project. With the limitation in mind, the preceding text OCR documents are stored in a different corpus and will not be used to train any model. On the other hand, when dealing with digital documents a Python script was developed with help of the library PDFReader [Polshcha24] to access the PDF files and extract the text. To avoid text duplication while parsing a PDF file, a decision was taken to leave out the headers and footers of the documents.

Finally, text document storage is performed. Each document is stored in 2 folders "OCR" and "doc" according to the origin of the document inside these main folders, they are stored in another folder keeping the institution as the folder name, and inside the folder the documents are enumerated, followed by an underscore, the title of the document, and finally the document domain/category.

3.2.1 Text Preprocessing

In machine learning and deep learning the term "data preprocessing" refers to the task of data cleansing, filling missing values, and treating duplicates. In order to reduce artifacts, noisy data, and bias. In NLP tasks in order to reduce lexical and spelling variations, text preprocessing is an essential step. It is commonly customized depending on the application needs, tasks, language, and corpus nature. It involves normalizing punctuation marks, capitalization, and resolving contractions, among other tasks [Chai23].

During the construction of the corpus, text documents were collected from different sources, such as web archives and websites. While there are encoding standards for text in order to preserve and correctly display information, some websites do not use them. This causes redundancy when different symbols and encodings are used to represent the same character. This was mainly present in characters with apostrophes or accents, quotes, and is also wide-spread in the space and tabulation character. Also, as previously noted, there are also different standards for writing in Purépecha, some of them using glyphs from the International Phonetic Alphabet (IPA). In order to address these language-specific challenges (E.G. text encoding, redundant symbols in the corpus), the text normalization task required specific normalization techniques [Chai23]. These normalization tasks will adhere to the extended American Standard Code for Information Interchange (ASCII) character set as the primary standard [INJ24] using the Decimal (Dec *Value*) representation.

The following tables will describe the work done to normalize character representations in the text.

Apostrophe and accents

Given the different sources of the documents, there were a plethora of different symbols that describe the accent and the apostrophe. Although some authors propose to clean the special characters while building and preprocessing the corpora, accents and apostrophes were kept. This is mainly because Purépecha relies heavily on accents. A feature present in the Purépecha language is that words can be spelled really similarly, with the exception of the accent or apostrophe. Therefore, a word has a totally different meaning. Some examples of the before-mentioned feature are shown in Table 3.7 .

w_1	translation for w_1	w_2	translation for w_2
kárani	to fly	**karáni**	to write
uérani	to leave	**ueráni**	to cry
k'uini	to sleep	**kuini**	bird

Table 3.7: Accents and context, Source : [Medina Perez06]

Table 3.8 shows the ASCII encoding of the different accents found in the corpus. The character selected to replace the redundant symbols is the single quote (Dec. 39) present in ASCII to represent the apostrophe, and combined with a vowel it represents the accent.

Dec.	Symbol	Description
39	**'**	**Single quote**
91	‘	Left single quotation mark
92	’	Right single quotation mark
96	‘	Grave accent
146	’	Single quote

Table 3.8: Accents and apostrophes normalization

Hyphen

To minimize the duplication of symbols representing the hyphen, the ASCII character “hyphen - minus” (Dec. 45) was chosen as the substitute, as illustrated in Table 3.9.

Dec.	Symbol	Description
45	**-**	**Hyphen-minus**
150	–	En dash
151	—	Em dash
175	¯	Spacing macron - overline

Table 3.9: Hyphen normalization

Space

To denote the space and homogenize its representation, the ASCII character “Space” (Dec. 32) was chosen to remove additional tabs and spaces.

Dec.	Symbol	Description
32		**Space**
160		No-break space
9		Horizontal Tab

Table 3.10: Space normalization

Dot

Table 3.11 illustrates the different characters present and used as points. The ASCII character "Period, dot or full stop" (Dec. 46) was selected as the replacement.

Dec.	Symbol	Description
46	**.**	**Period, dot or full stop**
183	·	Middle dot
38 35 56 50 50 56 59	.	Leader
133	...	Horizontal ellipsis

Table 3.11: Dot normalization

Quotation marks

Similarly quotation marks show redundancy of symbols. These symbols are present in Table 3.2.1 and the replacement for those is the ASCII character "Double quotes" or "speech marks" (Dec. 34)

Dec	Symbol	Description
34	**"**	**Quotation mark**
147	“	Left double quotation mark
148	”	Right double quotation mark
171	«	Left-pointing double angle quotation mark
187	»	Right-pointing double angle quotation mark

Table 3.12: Quotation mark normalization

International Phonetic Alphabet

Based on the research and publication of "JUCHARI UANDÁKUECHA nueva cartilla de lectura y escritura purepecha" from the now extinct "Instituto Nacional Indigenista" (INI) of Mexico, the following International Phonetic Alphabet (IPA) characters were previously used in the written word for the Purépecha language. Mainly by the research done by Maxwell Lathrop between the years of 1935 and 1970. Although these writing rules are more present in printed documents from that time, there were some digital documents that had this writing convention. The following symbols shown at Table 3.13 were replaced by the corresponding new characters in the Latin alphabet [Medina Perez70].

Old	Description	new	Dec.	Description
ʌ	IPA letter turned v	ï	207	Latin letter i with diaeresis
ɽ	IPA letter retroflex flap	rh	207	Latin letter r and Latin letter h
š	Latin letter s with caron	x	170	Latin letter x
ŋ	IPA letter eng	ng	110 103	Latin letter n and Latin letter g

Table 3.13: Previously used glyphs and their replacements

Hyphenated Sentence

In order to maintain the correct column width, some words are commonly cut at the end of the line, therefore the word is split by a dash. This pattern is more common in double or more columns like the ones widely present at newspaper or religious content. A Regular Expression (RegEx) filter was used to correct this situation. The Regular Expression filter search for a word pattern followed by dash, end of line, new line, and a word pattern. Examples of this filtering are present in Table 3.14

Hyphenated Sentence	Treated Sentence
... újpini o inchajpaani indi- jenechani niátakuecharhu ...	... újpini o inchajpaani indijenechani niátakuecharhu ...
... irechikuaeri, 1930 uéx- urhini ...	... irechikuaeri, 1930 uéxurhini ...

Table 3.14: RegEx filter example. The suspension points (...) represent sentence continuity.

Other symbols and non textual elements

Noisy data is shown to have a tendency to degrade performance among different NLP models and tasks [Bagla21]. In order to remove as much noise as possible inside the dataset, a Python script using a RegEx filter was developed to find other symbols in the corpus and store extended ASCII characters and their frequencies in a dictionary after manual review and cleaning of the replaced symbols; these characters are present in Table 3.15.

Symbol	Dec.	Description
•	8226	Bullet
○	9675	White Circle
●	9679	Black Circle
®	174	Registered Sign
▸	9656	Black Right-Pointing Small Triangle
□	9633	White Square
©	169	Copyright Sign
º	186	Masculine Ordinal Indicator
◌	9676	Dotted Circle
°	176	Degree Sign
ǂ	135	End Of Selected Area
	142	Single Shift Two
◼	9724	Black Medium Square
▪	9642	Black Small Square
�	65533	Replacement Character
⇒	8658	Rightwards Double Arrow
✔	10004	Heavy Check Mark
™	8482	Trade Mark Sign

Table 3.15: Other removed symbols

Also, a Python RegEx filter was developed to remove other remaining non-textual elements such as HTML tags, Uniform Resource Locators (URL), e-mail addresses, vignettes, extra spacing between words and empty line breaks.

3.3 Corpus Analysis

The next section will review the contents of the digital corpus, using various metrics to present the most frequent words and their distribution, the count of unique words, n-grams, and the average length of sentences in the documents.

3.3.1 Corpus size

In the next subsection, the corpus analysis will be performed. The total amount of documents included in the corpus, total words contained and the algorithm used to tokenize the whole corpus. Finally a comparison between the compiled corpus and other corpora for LRLs in the Americas.

Number of Documents

Data was gathered from different sources and domains. Table 3.16 shows the statistics of the documents collected for the digital and digitized corpus.

To calculate the number of sentences and tokens in each document, sentence segmentation was performed using various punctuation marks (including full stops, exclamation marks, question marks, periods, and colons), and a Python script split the documents into sentences and stored them with their sources. Subsequently, token segmentation was performed by splitting sentences into space-separated tokens using a counter to keep track of the source and token count. The processes mentioned above (sentence segmentation and space-separated tokens) was used by Tyers in its work with the Florentine Codex [Tyers23].

Source	Digital Corpus			Digitized Corpus		
	#Documents	#Sentences	#Tokens	#Documents	#Sentences	#Tokens
Religious	2,640	93,647	1,482,276	-	-	-
Educational	987	5,049	76,387	804	2,582	39,111
Science	486	3,409	98,547	48	207	3,514
News	-	-	-	347	2,236	41,022
Narrative	286	2,803	44,855	572	1,909	50,375
Legal	237	2,027	59,498	26	65	1,433
Lyric	77	333	6,797	-	-	-
Total	4,713	107,268	1,768,360	1,797	6,999	135,455

Total Documents	6,510
Total Sentences	114,267
Total Tokens	1,903,815

Table 3.16: Dataset distribution in the corpus

The Purépecha corpus, as shown at the end of Table 3.16, is composed of 114,267 unique sentences and 1,903,815 tokens. A comparison between other corpora for Indigenous languages in the Americas is also present in Table 3.17. It is worth noting how languages with much more active speakers such as Mayan and Quechua have such a great representation of their written word and how they even have specific corpora for their specific variations. It is also worth noticing how Inuktitut (Inuit Languages) although having a smaller active speakers population has developed a large corpus. On the other hand, languages with less than 2000 active speakers such as Waikhana and Kotiria struggle to have written representation of their language.

Total words

To calculate the number of words, a simple tokenization was performed. This process splits sentences and treats space-separated tokens as words. In order to achieve this task, it was required to traverse the segmented sentence data set, storing the source and number of tokens within each sentence. The overall process is described in Algorithm 1

Language/Corpus Name	Number of Sentences
Cherokee (Cherokee Corpus)	10,868
Guarani	14,531
Inuktitut (Nunavut Hansard Inuktitut–English Parallel Corpus 3.0)	1,450,094
Kotiria	198
Mayan (MAYAN V)	1,230,470
Mayan k'iche' (Bible UDEIN)	7,895
Mayan of Yukatan	263,806
Mazatec - Spanish	9,799
Mixtec (Kolo)	1,900
Mixtex - Spanish	13,235
Nahuatl (Axolotl)	12,207
Nahuatl (Bible UEDIN)	7,821
Otomi (TSUNKUA)	5,515
Purépecha	**114,267**
Quechua Chanka (jw300_2013)	124,038
Quechua of Ayacucho	128,434
Quechua Southern (Monolingual-Quechua-IIC)	384,184
Waikhana	122

Table 3.17: Sentence distribution across different Indigenous languages corpora of the Americas.

and this method accounts for 1,605,574 space-separated tokens for the data set containing originally digital documents and 135,455 for digitized documents.

Input: Dataset df with field `type` and field `string`
Output: $totalWords$, the total number of space-separated tokens

$totalWords \leftarrow 0$
$sources \leftarrow$ unique values of $df.type$
for each $currentType \in sources$ **do**
  $counter \leftarrow 0$
  for each $document \in df$ **do**
    $tokens \leftarrow$ split($document$, "space")
    $counter \leftarrow counter + |tokens|$
  end for
  $totalWords \leftarrow totalWords + counter$
end for

Algorithm 1: Space-separated tokens algorithm

3.3.2 Document Length

Average document length

The box plot in Figure 3.3 shows the dispersion of words per sentence in the different types or categories of text within the corpus, and the rightmost shows the same metric throughout the corpus (Y and X axes, respectively).

Religious, educational, narrative, and other text types have lower median word counts, generally between 10-20 words per sentence. This suggests simpler or more straightforward sentence structures. On the other side of the spectrum, legal and science text show relatively high word counts, with a median of about 25 words per sentence and a broad range of up to 70 or 60 words per sentence. This reflects how these texts tend to be more verbose and their formal structure. Lyric texts exhibit much shorter word counts, reflecting a tendency toward brevity and simplicity, which is common in poetry and lyrics.

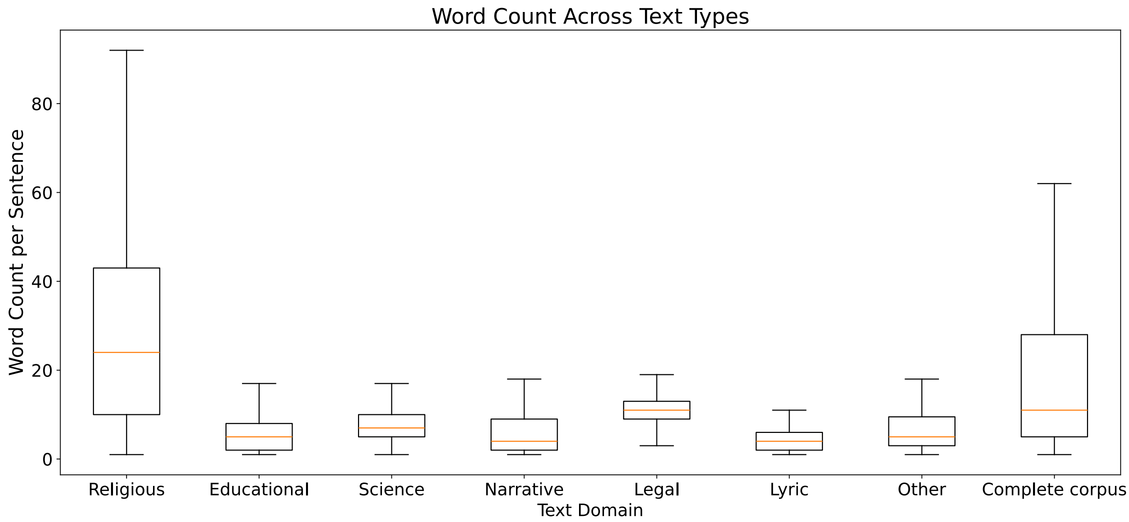

Figure 3.3: Corpus sentence length across, different text types.

Average sentences per document.

The box plot in Figure 3.4 represents the length of the sentence of the documents on various types of text (Y and X axes, respectively). Religious texts have the widest range of sentence lengths, with some documents containing up to 80 sentences with a median of approximately 20, however, there is significant variability. This is likely because of the different writing styles or purposes. Educational, science, narrative, legal, lyric, and other text types have much shorter and consistent sentence lengths compared to religious texts. Their median sentence counts range between 5 and 15 sentences, with little variation.

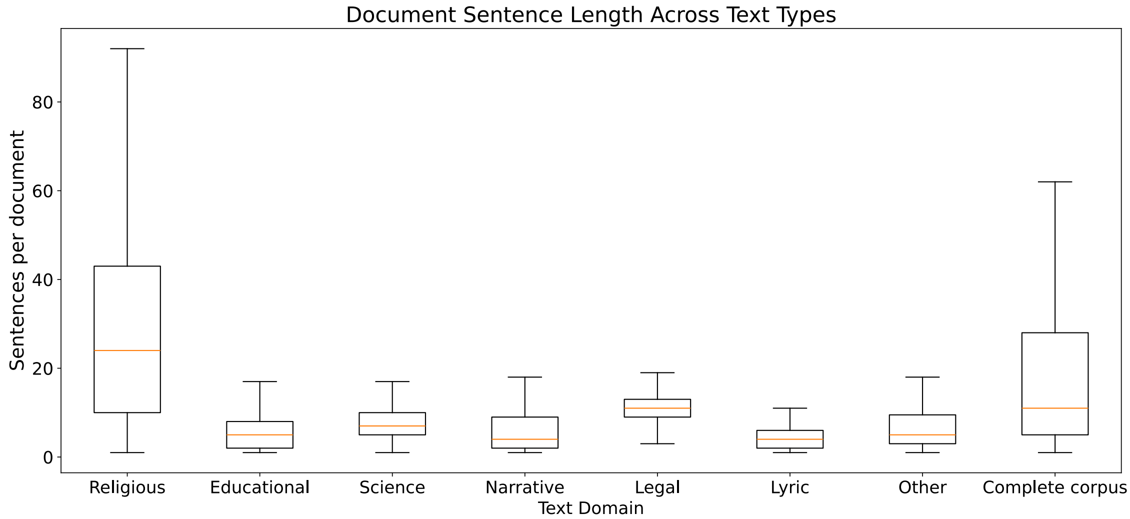

Figure 3.4: Average document sentence size across text types in corpus.

3.3.3 Word Frequencies

Most frequent words

Different natural phenomena such as physical, social, or biological systems can be modeled or represented as power series [Newman05]. In the realm NLP it is commonly referred to as Ziff's law, which states that the word that appears most frequently will occur about twice as often as the second most frequent word, which itself appears roughly twice as often as the fourth most frequent word, continuing in this pattern [Newman05]. This pattern is modeled according to Equation 3.1. Where variables a and b usually have the following values $a \approx 1$, $b \approx 2.7$

$$\textbf{token frequency} \propto \frac{1}{(rank + b)^a} \tag{3.1}$$

Figure 3.5 compares the 100 most frequent tokens in the digital corpus, highlighting a general decreasing trend that follows Zipf's law.

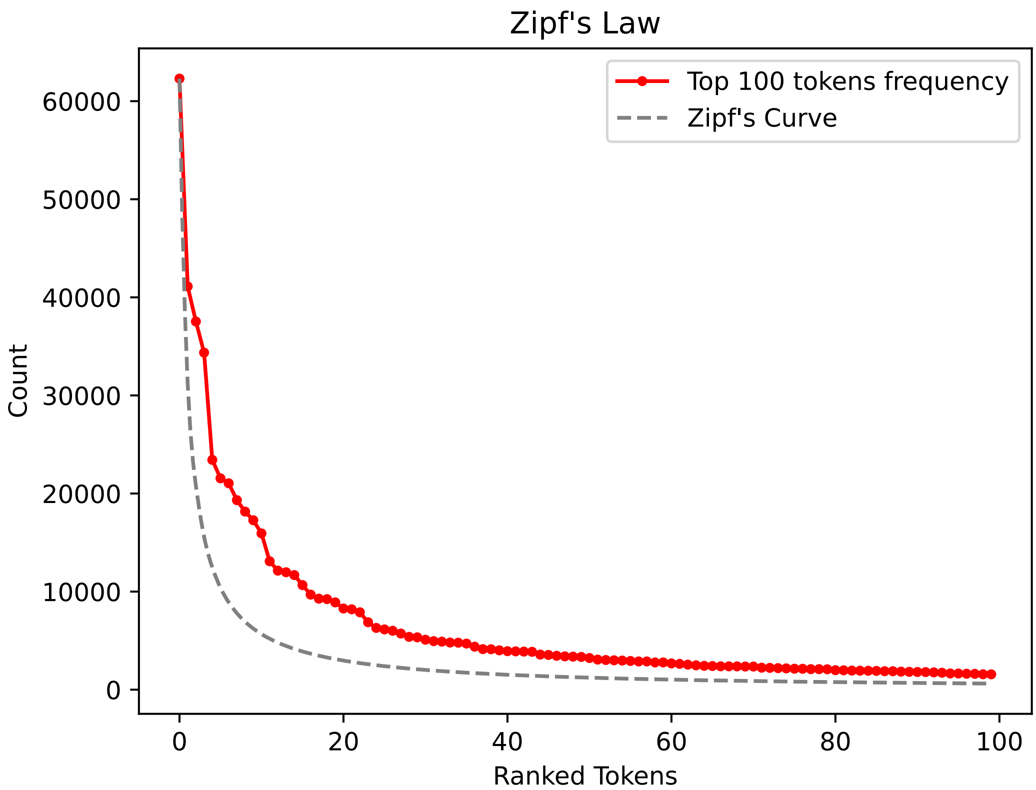

Figure 3.5: Top 100 Token frequency decay in digital corpus

In order to calculate the most frequent words present in the different data sets a frequency distribution was calculated using a Python dictionary where words were added and in case the element was already present, an increment in the counter was done with that specific key. This methodology follows the work presented in [Villena20] analysis of multilingual corpora of medical text.

In the following Figure 3.6 shows the 10 most frequent words within the digital corpus and their respective counts decrease. The most frequent word in the Purépecha language is “ka” followed by “ambe”.

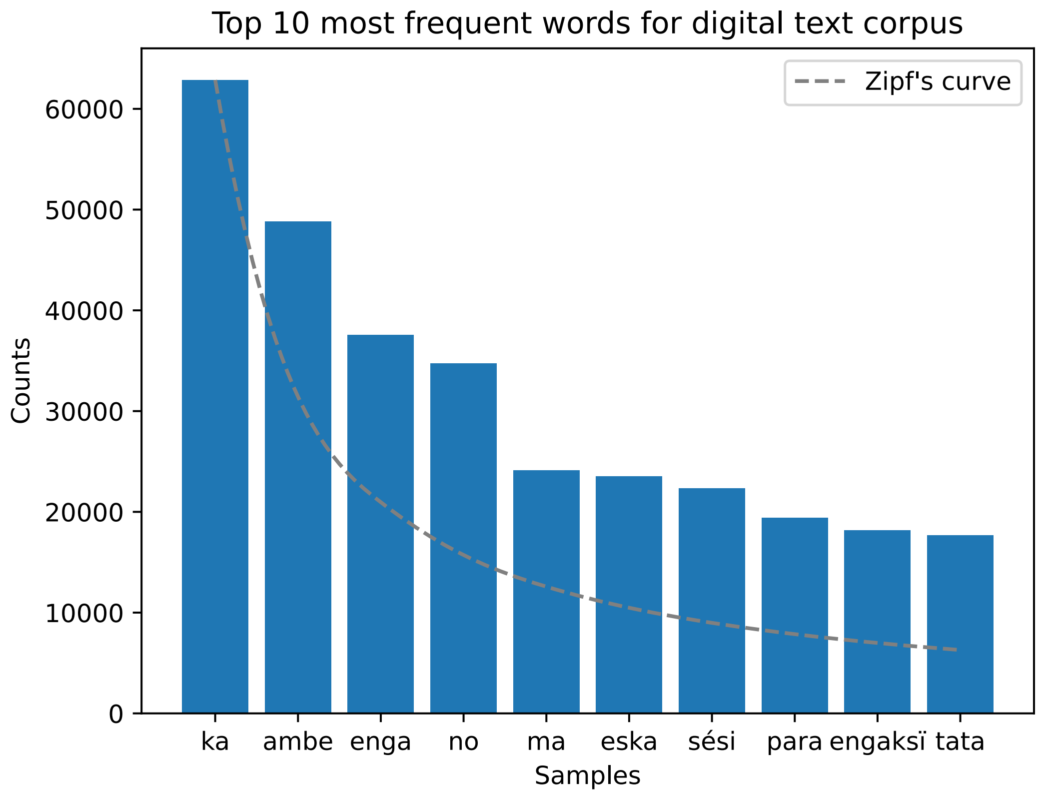

Figure 3.6: Top 10 most frequent words for digital text corpus.

The nondigital or digitized corpus shows a similar composition with words like "ka", "ma", "no" and "enga" being again in the first six places; however, the word " jimbo" is present instead of "tata" in the most common words as shown in Figure 3.7. This is mainly due to the absence of religious text in which the word "tata" is frequently used to name God or any other notable person.

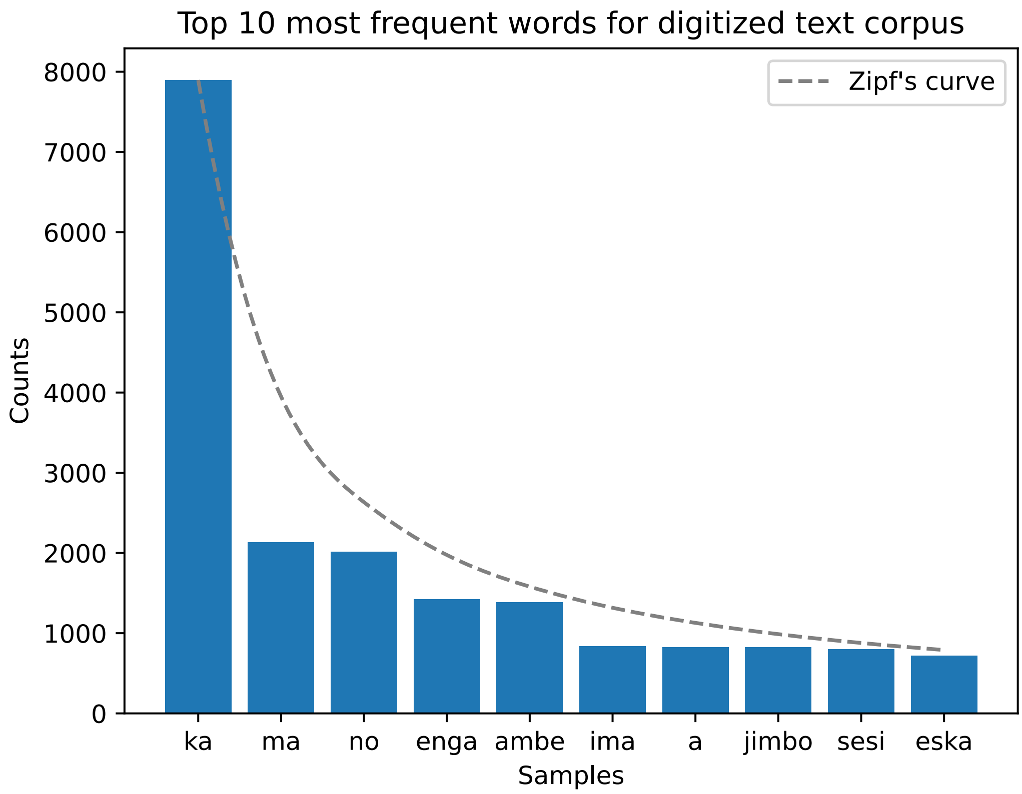

Figure 3.7: Top 10 most frequent words for digitized text corpus.

3.3.4 Legomena

The terms *Hapax Legomena* and *Dis Legomena* refers to space-separated tokens that appear only once in a corpus and the ones that appear only twice, respectively, as explained in Kornai's book [Kornai07]. The counts of space-separated Hapax and Dis Legomenon tokens for each corpus are shown in Table 3.18. A comparison between both corpora and their respective count decrease is shown in Figure 3.8; in red and blue color, both the Hapax and the Dis Legomenon are shown, in black color the tokens with a higher than 3.

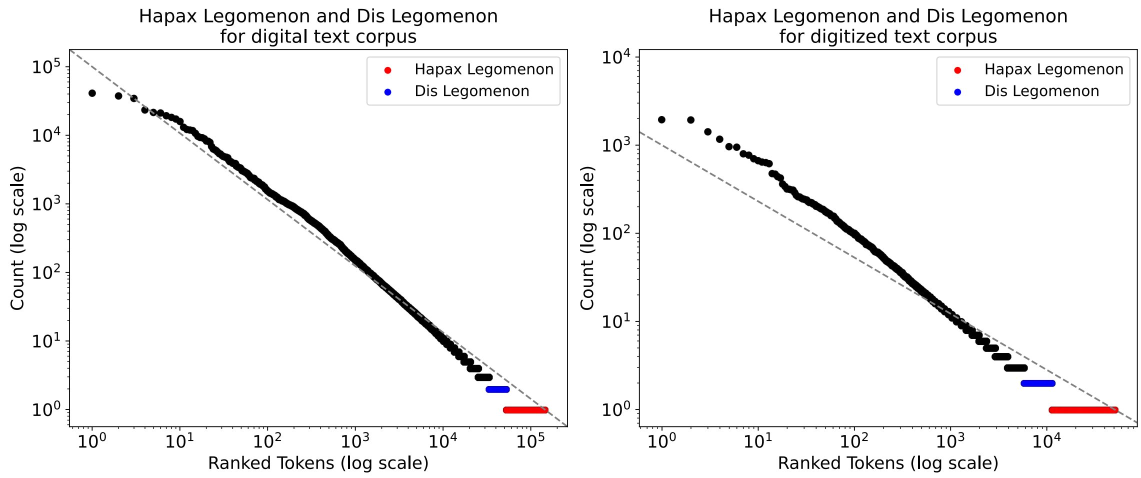

Figure 3.8: Hapax and Dis Legomenon in corpus.

The digital corpus has a slightly above-average percentage (63.87%) of single terms, typically found between 40 to 60%, while tokens appearing twice are within common parameters, which are between 10 to 15% [Kornai07]. In contrast, the digitized corpus has a much higher occurrence of single tokens (clearly shown in the right side of Figure 3.8), although twice-appearing tokens align with typical percentages.

Corpus	Hapax Legomenon		Dis Legomenon	
	Count	Percent (%)	Count	Percent (%)
Digital	92,404	63.87	18,848	13.02
Digitized	39,544	77.89	5,452	10.73

Table 3.18: Hapax and Dis Legomenon distribution in corpus

3.3.5 N-Gram Frequencies

In order to perform N-gram counting in the following subsection, the text was lowercased and the punctuation was removed as suggested by Jurafsky's textbook [Jurafsky24].

Top 10 Bigrams

In Figure 3.9 the 10 most frequent bigrams are shown in the digital corpus. The impact of the religious text within the corpus is clear with bigrams such as "*tata diosï*" (God Father), "*tata diosïri*" (To/For God Father), "*diosï jeoba*" (Jehovah God). There are also references to a subject or a way of referring to a trade or profession without using the suffix "*ri*", for example "*ambe enga*" (the one(who)), "*ambe ma*" (one who), "*ma enga*" (one who), "*imani ambe*" (to that one/that one (who)). There is also present "*ju je*" (let us do/go) a term that is also widely used in educational texts.

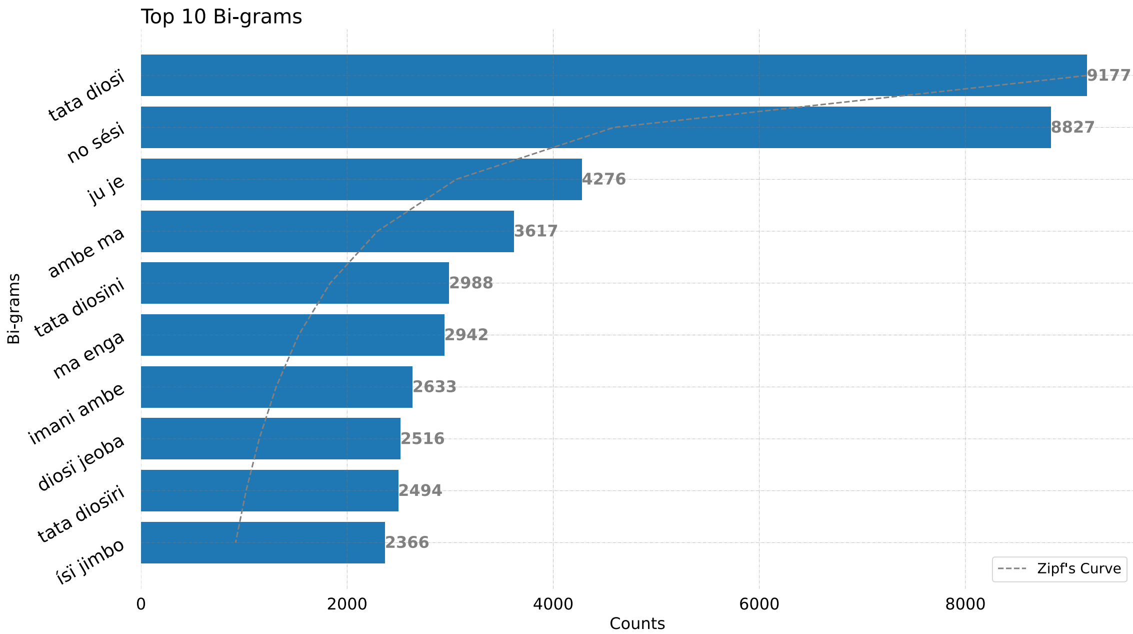

Figure 3.9: Top 10 most frequent bigrams for digital text corpus.

The bar graph present in Table 3.10 shows the top 10 frequent bigrams inside the digitized corpus. The following phrases in common use appear widely in the corpora: "*ka isi*" (and so), "*ka ma*" (and one), "*enga no*" (no one), "*ka ima*" (and he/she), "*no sesi*" (wrong), "*ia ka*" (have and).

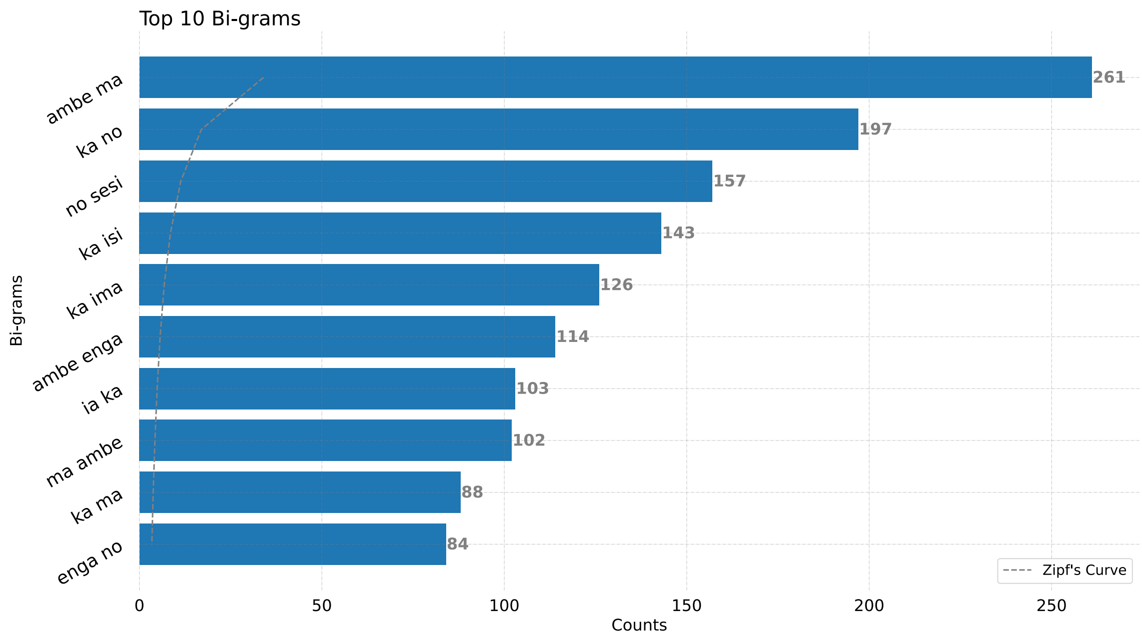

Figure 3.10: Top 10 most frequent bigrams for digitized text corpus.

Top 10 Trigrams

The graph in Figure 3.11 shows the top 10 more common 3-grams and their counts in the digital corpus. Again the impact of religious text is clear with 3-grams like "*tata diosï jeoba*" (God Father Jehovah), "*tata diosï jeobani*" (of God Father Jehovah), "*enga tata diosi*" (the one who god father), the following "*no sési jási*", "*no sesi ambe*", "*enga no sési*" are used to describe bad behavior or curses also commonly present in religious and narrative text. And finally, also present "*ju je exeni*" (let us see) a term that is also widely used in educational texts as an introduction or an example.

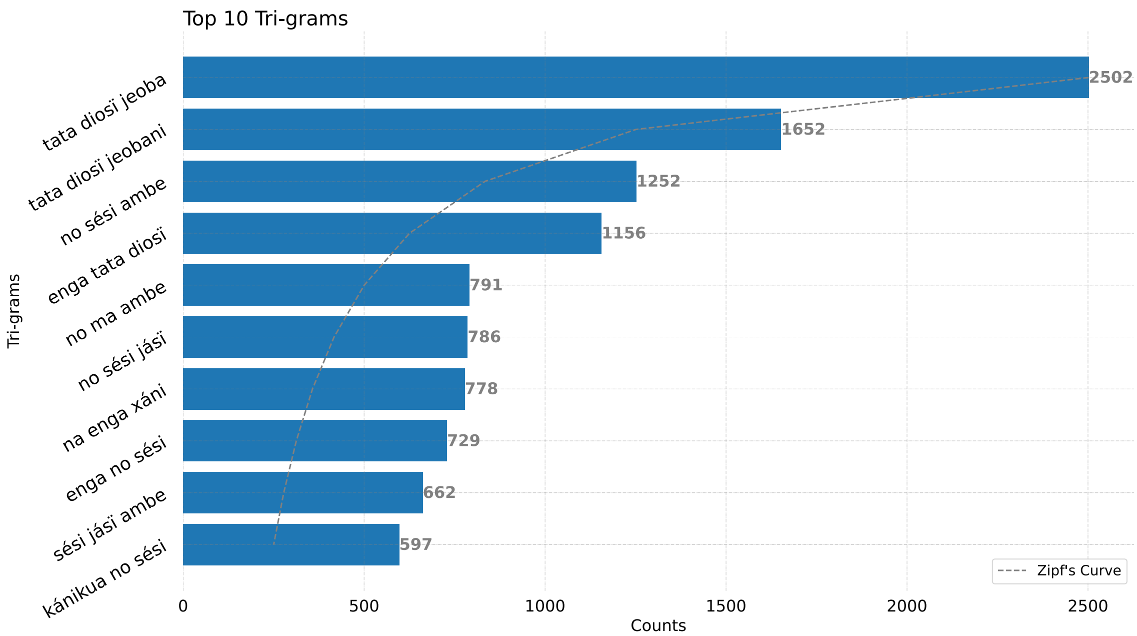

Figure 3.11: Top 10 most frequent trigrams for digital text corpus.

The bar graph in Figure 3.12 shows the 10 most common trigrams in the digitized corpus. The most frequent is "*ambe ma enga*" (that one who), followed by "*ka isi jimbo*" (and that is why), "*no ma ambe*" (no one who). It also appears "*felipe chavez cervantes*" the name of an award-winning journalist and promoter of the Purépecha language. "*karani ka arhintani*" (to write and to read) is frequently used in educational texts. Some connection words are also present, such as "*ka eka na*" (and when) and "*ka isi jimbo*" (and the reason why).

Top 10 Trigrams

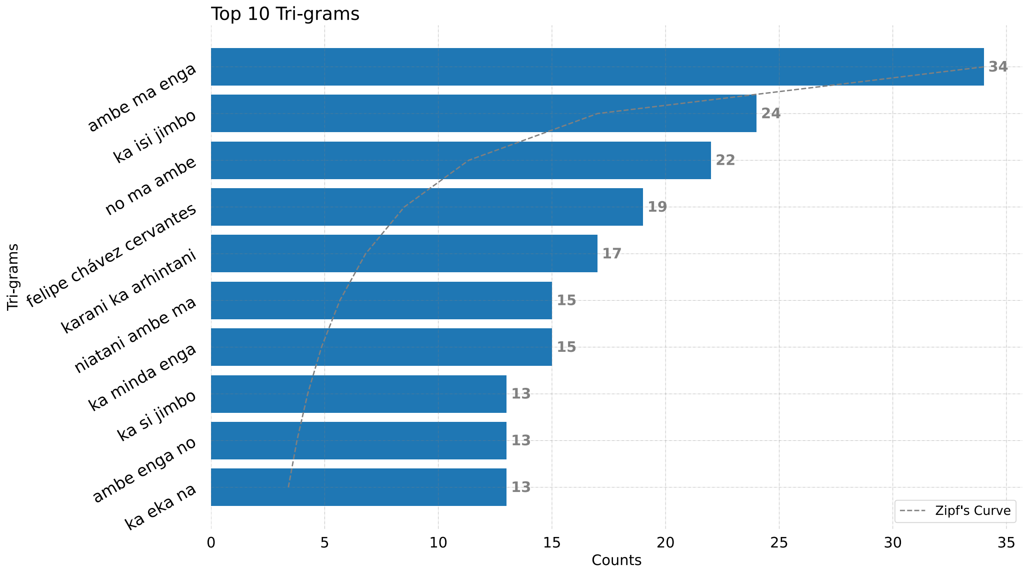

Figure 3.12: Top 10 most frequent trigrams for digitized text corpus.

3.4 Final Remarks

In this chapter, the main features of the Purépecha language were discussed, such as the distribution and demography of its speakers in the whole of Mexico, as well as its numerals, alphabet, the different writing standards, agglutinative structure, and how it relies heavily on accents and apostrophes. The lack of a publicly open corpus for this language is also exposed and why it aims to help preserve and study this language.

The proposed methodology used to locate, gather, and process text documents for this language. Among text processing and data cleansing, the importance of regular expressions and how they play a key role in finding patterns and helping normalize characters a task that must be almost tailor-fit for this target language.

Finally, a corpus analysis was performed for both the digital and the digitized corpus. Showing the document, sentence, and token count for both text domain and corpus-wide. Other metrics such as average sentence length, token and n-gram frequencies, hapaxes, and how they show decrease tendency according to Zipf law.

Chapter 4

Experimental Setup and Methodology

4.1 Introduction

The development of Language Models has resulted in different evaluation metrics, which are grouped into three categories: Multiple-Classification (MC) metrics, Token Similarity (TS) metrics, and Question-Answering (QA) metrics. The initial MC denotes the proficiency of a model in categorizing text in at least two distinct classes. TS evaluates the performance of a model predicting text and then aligning it with the expected output. Finally, QA assesses how well a model can respond to a particular question using the information provided within a specified context [Hu24].

In order to evaluate how text models perform across the previously mentioned domains, benchmarks have been developed such as: General Language Understanding Evaluation (GLUE) [Wang18], Biomedical Language Understanding Evaluation (BLUE) [Peng19], Benchmark of Linguistic Minimal Pairs (BLiMP) [Warstadt20], Quora Question Pairs (QQP) [DataCanary17], Semantic Textual Similarity Benchmark (STS-B) [Cer17]. An example of the NLP tasks present in the before-mentioned benchmarks is presented in Table 4.1.

Dataset	Description	Data example
CoLA	Is the sentence grammatical or ungrammatical?	"This building is than that one." = Ungrammatical
SST-2	Is the movie review positive, negative, or neutral?	"The movie is funny , smart and visually inventive." = .93056 (Very Positive)
MRPC	Is the sentence B a paraphrase of sentence A?	A) "Yesterday , Taiwan reported 35 new infections." B) "Taiwan reported another 35 probable cases yesterday." = A Paraphrase
STS-B	How similar are sentences A and B?	A) "Elephants are walking down a trail." B) "A herd of elephants are walking along a trail." = 4.6 (Very Similar)
WNLI	Sentence B replaces sentence A's ambiguous pronoun with one of the nouns - is this the correct noun?	A) "Lily spoke to Donna, breaking her concentration." B) "Lily spoke to Donna, breaking Lily's concentration." = Incorrect Referent

Table 4.1: Summary of GLUE Tasks

However, these benchmarks focus mainly on the MC and QA metrics. Commonly require an existing corpus, require tagging of the corpora for the specific task in order to fine-tune a pre-trained model, or are only available with a narrow language list. Unfortunately, Purépecha is not among the supported languages, and tagging the corpus for tasks such as similarity between sentences, grammatical correctness, paraphrasis, or other language inference tasks are not within the scope of the project.

4.2 Experimental Setup

Evaluating TS performance of a language model without relying on specific downstream NLP tasks can be accomplished through other specific evaluation metrics. Those key metrics include Pseudolog-likelihood, Pseudo-perplexity or surprisal. These metrics provide insights into the model's ability to capture the probability distribution of a given text.

The Pseudolog-likelihood (PLL) evaluates how well the model assigns probabilities to sequences of tokens, while Pseudo-perplexity (PPPL) measures the model's uncertainty in predicting tokens across sequences, providing an estimate of how "surprised" the model is by new data. Surprisal quantifies the unexpectedness of specific tokens or sequences, which reflects the model's comprehension of language patterns. These metrics offers an effective alternative to task-based evaluations for language models [Fruehwald22b].

4.2.1 Pseudolog-likelihood

The evaluation scheme selected for this work is based on the methodologies described by Salazar for scoring MLM [Salazar20]. The overall process is as follows. A token within a sentence is masked and $\log(P_{MLM}(w_i|W_{/i};\Theta))$ is calculated, where MLM specifies a Masked Language Model, w_i is the masked token, $W_{/i}$ represents the context window, and Θ denotes the intrinsic parameters of the model. The PLL of a sentence is then calculated by sliding the masked token one position within the sentence on each iteration, and accumulating the values as shown in Equation 4.1.

$$PLLs(W) = \sum_{i=0}^{|W|} \log(P_{MLM}(w_i|W_{/i};\Theta)) \tag{4.1}$$

However, this metric relies on masking tokens, which sometimes can be within sub-word units, which can have compatibility issues with an N-gram-based approach. A comparison between token masking and whole-word-masking for the PLL evaluation is present in Table 4.2

Target Sentence	PLL original	PLL (whole-word-masking)
misitu uarhime itsïrhu (the cat fell into the water)	['misitu', '<**mask**>' ,'*##rhi*', '*##me*', 'itsïrhu'] ['misitu', '*ua*' ,'<**mask**>', '*##me*', 'itsïrhu'] ['misitu', '*ua*' ,'*##rhi*', '<**mask**>', 'itsïrhu']	['misitu', <**mask**>, 'itsïrhu']

Table 4.2: Comparison between whole world masking and the original sub-word masking for PLL evaluation

In order to overcome this incompatibility issue while evaluating, a different approach was taken, it consist of evaluating the PLL using full-word masking [Kauf23]. A side-by-side comparison of the full-word masking PLL evaluation of the models can be observed in Figure 4.1

The evaluation process takes a sentence from the dataset, and a language model. Then evaluates the PLL of the entire sentence by calculating the probabilities of each token given by their context. This step is then repeated for all sentences within the designated dataset. The individual PLL values are then accumulated, contributing to a global PLL counter that reflects the overall performance of the model across the data set.

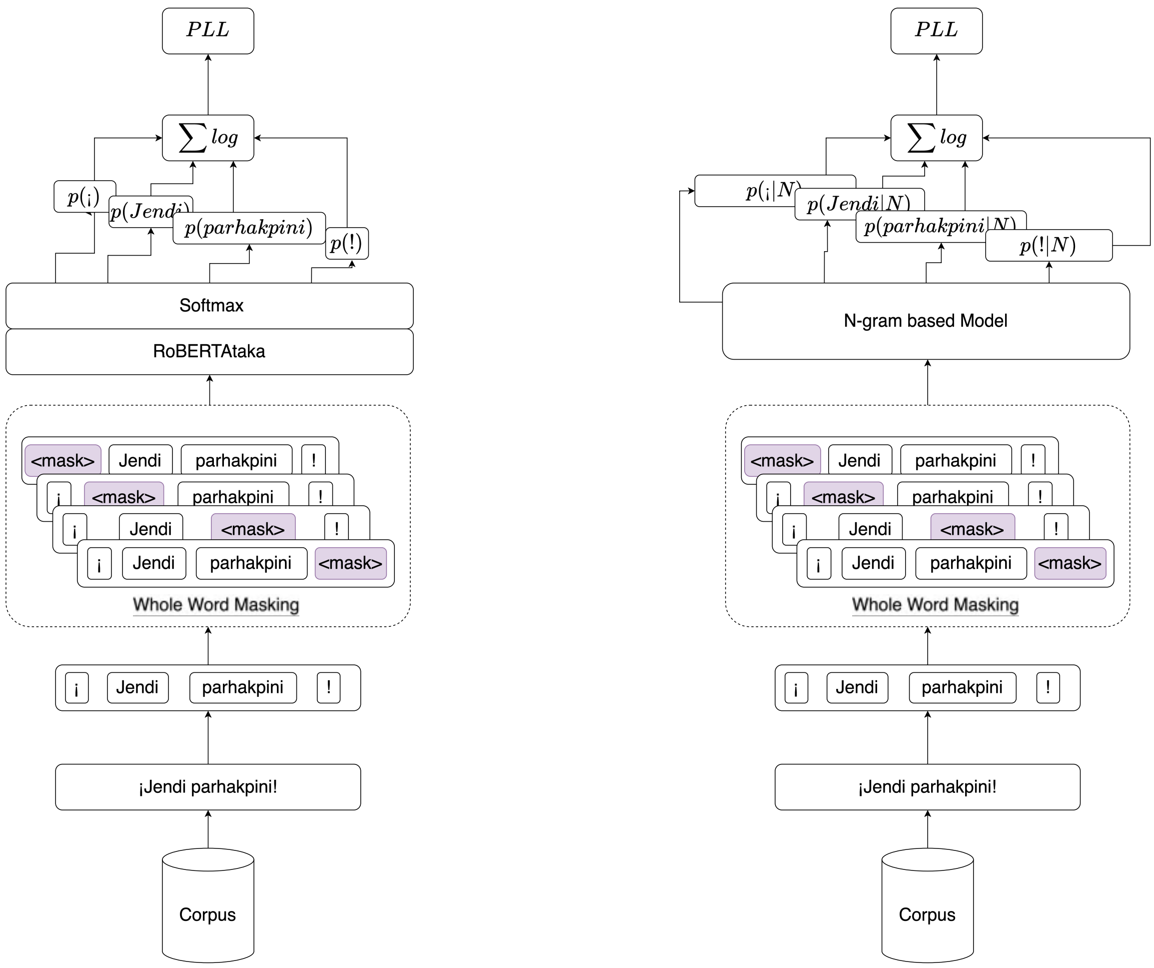

Figure 4.1: Methodology for evaluating performance of text models using PLL. Left the approach for a transformer model (RoBERTa), right for a N-Grams based model

4.3 Data preparation

4.3.1 Data Cleaning

Dirty data can skew results and lead to inaccurate predictions in language models. Jiang et al. [Jiang24] analyzed the performance of different models and highlighted how the limited availability of open-source corpora—such as the one curated in this work—complicates efforts to assess the impact of contamination on model performance.

With this in mind and having identified that OCR software shows problems in properly identifying letters and word combinations when the target language is not present within its training stage, as also reported by Capistran et al. while curating the Axotl corpus [Capistran15]. The decision was made to exclude the documents that had been digitized with OCR software, and duplicates were removed. The composition of the dataset is detailed in Table 4.3

Document Domain	Number of Sentences
Religious	93,513
Education	5,023
Science	43,403
Narrative	2,840
Legal	2,074
Lyric	327
Total	107,180

Table 4.3: New dataset distribution.

4.3.2 Data Splits

Data splitting is the process of dividing the data available in at least two parts in order to compare how different models perform with the same data. Usually it is used a training set and a test set. The training set is the portion of data used to train the model. The model should observe and learn from the training set, optimizing any of its parameters.

It is also fairly common practice to divide the training set into a validation split that helps measure the accuracy of the models between the training steps. Finally, the test split is a portion of the data that is reserved from the beginning and it is used as an evaluation for a fitted or trained model.

Because the text domains have different proportions and to avoid issues regarding uneven distribution of data associated with random sampling, the data splits were sampled using stratified random sampling to ensure that the datasets are correctly distributed in the data splits [Sadaiyandi23]. The composition of the data splits is shown in Table 4.4

Split Name	Sub Splits	Number of sentences	% of data	
Train	Train	68,595	64 %	80%
	Validation	17,149	16 %	
Test		21,436	20%	

Table 4.4: Dataset splits

4.4 N-Grams

N-gram language models are based entirely on statistics, relying in the recurrence of words. The predictions for the next word are based on a fixed context window of size n of the previous words, also known as a context window.

4.4.1 Smoothing

N-gram language models face challenges when processing words and word sequences absent from the model's vocabulary. This issue in counting word occurrences may result in a computational error as elaborated in Equation 4.2.

$$
\begin{aligned}
Counts(\text{uichu ch'anasïndi}) &= 0 \\
P(\text{uichu ch'anasïndi}) &= \frac{Counts(\text{uichu ch'anasïndi})}{N} = \frac{0}{N} = 0 \\
P(\text{ch'anasïndi} \mid \text{uichu}) &= \frac{P(\text{uichu ch'anasïndi})}{P(uichu)} = \frac{0}{N} = 0
\end{aligned}
$$

$$PLL(\text{uichu ch'anasïndi}) = \log(P(\text{uichu} \mid \text{ch'anasïndi})) = \log(0) = -\infty$$

$$PLL(\text{Juchiti uichu ch'anasïndi}) = \sum_{i=0}^{|W|} \log(P_{MLM}(w_i|w_{i-1}) = ... + -\infty + ... = -\infty \quad (4.2)$$

To overcome this issue, two solutions were added: a mechanism to deal with OOV words and missing n-grams of words that were in the vocabulary. These approaches follow the guidelines described by Jurafsky [Jurafsky24].

In order to deal with OOV words, the model will include the *<unk>* token, and words below a given count are converted into the before-mentioned token. This process is also known as the cut-off value. In this work, models were trained with cut-off values of 1, 2 and 3. Although cut-off values were introduced, some rare combinations of both known and *<unk>* tokens can still occur within the testing split. As shown in Table 4.5.

Conditional	Probability	PLL
P(misitu | <s>)	0.00003	-10.26501
P(uarhime | misitu)	0.0	$-\infty$
P(itsïrhu | uarhime)	0.0	$-\infty$
P(. | itsïrhu)	0.10415	-2.26176
P(<\s > | .)	0.98972	-0.01032

Table 4.5: Although all tokens were known in this test sentence *Misitu uarhime itsïrhu* (The cat fell into the water), the combination of two of them *uarhime, itsïrhu* were not present during the training phase resulting in a computational error.

Laplace smoothing was introduced in the n-gram language models to address this issue. The Laplace smoothing algorithm makes a grid of all possible n-grams and then increments with one to the global count. After the counting, the n-gram probabilities are recalculated as described in the Equation 4.3. n stands for the order of n-grams and V refers to the vocabulary size.

$$P_{Laplace}(w_i|w_{i-n+1:i-1}) = \frac{Counts(w_{i-n+1:i-1}w_i) + 1}{Counts(w_{i-n+1:i-1}) + V} \tag{4.3}$$

Conditional	Probability	PLL
P(misitu | s)	2.30534e-5	-10.67769
P(uarhime | misitu)	1.14436e-5	-11.37807
P(itsïrhu | uarhime)	1.14443e-5	-11.37801
P(. | itsïrhu)	6.86286e-5	-9.58679
P(s | .)	0.44589	-0.80766

Table 4.6: Apllying Laplace smoothing to the n-gram model allows accurate PLL calculation without errors from rare n-gram combinations.

4.4.2 Back-off

In order to boost the performance of n-gram language models, two different back-off alternatives were tested. The first relies on an ensemble of n-grams that calculates the trigram probability; this is done by mixing together the probabilities of the bigram and the trigram, each weighted by λ as introduced by Jurafsky and Martin [Jurafsky24] and detailed in Equation 4.4.

$$\hat{P}(w_i|w_{i-2} : w_{i-1}) = \lambda_1 P(w_i|w_{i-1}) + \lambda_2 P(w_i|w_{i-2}) \tag{4.4}$$

The second method involves computing the trigram probability by selecting the maximum value of the trigram probability or, alternatively reverting to the lower-order bigram model when necessary, as shown in Equation 4.5 [Brants07].

$$\bar{P}(w_i|w_{i-2} : w_{i-1}) = \max\{P(w_i|w_{i-1}), P(w_i|w_{i-2})\} \tag{4.5}$$

4.4.3 Lower Casing

Lower Casing is a simple strategy in which the whole vocabulary of the n-gram model is lower cased. This results in an increase in the count and therefore, in the probability of a n-gram. However, this approach can lead to some issues where a token has a different context between its lower casing and the first capital variation, such as proper names. This increment count in the occurrence of n-grams and unigrams was reported by Dressel in his work with US news corpora, Twitter, and blog entries [Dressel16].

An example of this approach is shown in Table 4.7 and it is clear how the count for a specific n-gram increases.

ngram	Counts
"Ni je"	4
"ni je"	22
"ni je" (lower cased vocabulary)	26
"Na xani"	2
"na xani"	40
"na xani" (lower cased vocabulary)	42

Table 4.7: Lower Casing of 2 phrases: "ni je" (come in) and "na xani" (how much)"

4.5 RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa), a transformer model introduced by the Facebook AI team [Liu19], it is a variation of the BERT transformer model introduced by Google in 2019 [Devlin19]. RoBERTa differs from BERT in a few key areas in the pre-training of the model, such as dropping the next-sentence prediction extending the MLM task, it also differs from the tokenizer used, changing from word piece to Byte-level Byte Pair Encoding. The selected RoBERTa model uses a stack of 12 encoder layers, each with 12 multi-attention heads with an input lenght of 512 tokens.

RoBERTa makes use of the bidirectional self-attention mechanism to improve initial word vector representations by adjusting information based on subsequent words in the input sequence between context layers k and $k + 1$. Figure 4.2 graphically represents this mechanism processing the input: *Juchiti iuchu sesi kurhándisïni*(my dog understands well), with thicker lines indicating higher self-attention values between *iuchu*(dog) and *kurhándis-*

ïni(understands).

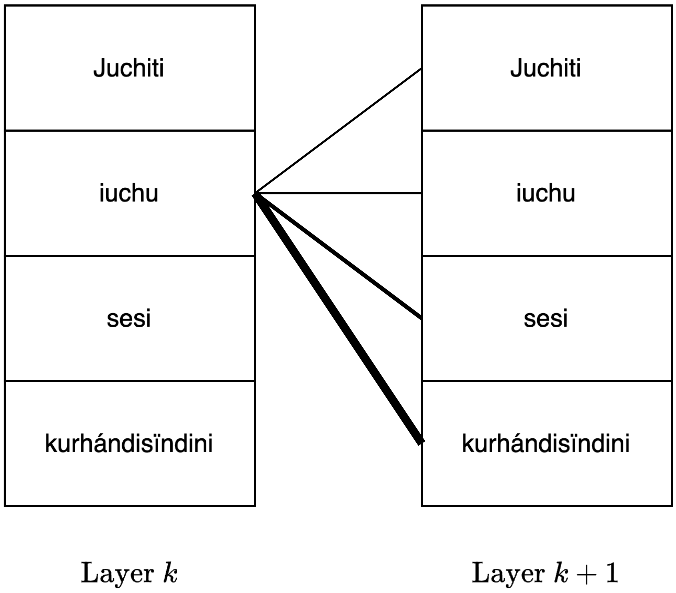

Figure 4.2: Self-attention distribution between layers in RoBERTa.

BBPE tokenizer

Byte-level Byte Pair Encoding (Byte-level BPE) is a variation of Byte Pair Encoding tokenizer introduced in [Sennrich16]. At it core BPE tokenizer first pre-tokenizes sentences, turning them into space-separated tokens, then are split into subword units counting the frequency of the adjacent tokens and updating them in the vocabulary. The process is described in Algorithm 2.

BPE shows great performance in dealing with agglutinative languages due to its ability to effectively split words into known subword units like the ones present in morphemes, suffixes, and prefixes [Sennrich16].

In Byte-level Byte Pair Encoding, the vocabulary is composed of bytes rather than characters. To achieve this, each character is broken down into its corresponding sequence of bytes according to the Unicode standard. This method offers the advantage of eliminating unknown characters and also conserves space when listing individual elements (i.e., bytes) within the vocabulary [Wang19].

df ← data

vocabulary ← all unique characters in df

k ← set number of merges

for i = 0 **to** k **do**

    leftToken, rightToken ← most frequent adjacent tokens

    newToken ← leftToken + rightToken

    vocabulary ← vocabulary + newToken

    replace leftToken, rightToken in Vocabulary

end for

Algorithm 2: BPE tokenization algorithm

An example of how the Byte-level BPE properly encodes some common Purépecha words is shown in Table 4.8. It is worth noticing how some words are properly split into smaller subword units and how a special character is added between different words. Certain variants of the BPE tokenizer choose to represent subword units with a special character. However, this is merely a design decision with no effect on performance.

Input	BBPE tokens
Kurhikuaeri	'Kurhi', 'kuaeri'
Tata Jurhiata	'Tata', 'ĠJurhiata'
Nana Kutsï	'Nana', 'ĠKutsÃ¯'
misitu uarhime itsïrhu	'misi', 'tu', 'Ġuarhi', 'me', 'ĠitsÃ¯rhu',

Table 4.8: Byte-level BPE tokenization for some Purépecha words.

Vocabulary

The different RoBERTa models in this work were trained from scratch without relying on pre-trained models and transfer learning following the same approach as Lou et al. in their Mayan models [Lou24]. The models were fitted with different vocabulary sizes. Evenly distributed from 80% to 90% of the accumulated frequency between the complete vocabulary present in the corpora, as shown in Figure 4.3

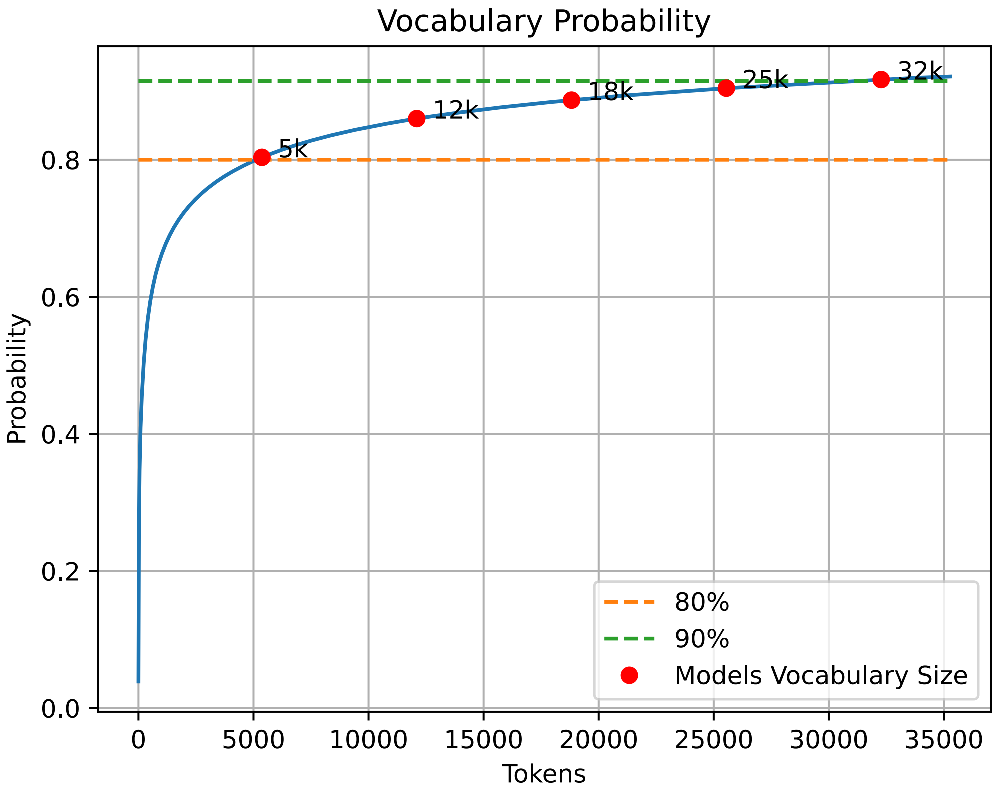

Figure 4.3: Models Vocabulary Target Size

Training Hyperparameters

The RoBERTa models were trained with different vocabulary lengths ranging from 5-32 thousand tokens as their vocabulary size, other hyperparameters are present in Table 4.9. The models are named RoBERTataka which is a pun between the name of the transformer model and *Tataka* (kid) in Purépecha.

Model Name	Vocabulary Size	Hyperparameters
RoBERTataka5	5,367	Learning rate: 1e-4 Seed: 42 Train epochs: 15 Max length: 512 Padding: max length Truncation: true
RoBERTataka12	12,095	
RoBERTataka18	18,822	
RoBERTataka25	25,550	
RoBERTataka32	32,278	

Table 4.9: Training Hyperparameters

4.6 Results

Both phases of the experiment, training and testing, were performed in a virtual server environment using tmux and slurm to queue and run the tests. Although the physical server has more hardware resources, the virtual server environment specs are displayed in Table 4.10.

Name	Description
Processor	1 Intel Xeon Gold 6330
Processing Cores	12
RAM	64 GB
GPU	1 NVIDIA GA100 [A100 PCIe 80GB]
Kernel	6.1.0-9 x64
Distro	Debian GNU/Linux 12
Python Version	3.11.2

Table 4.10: Computer Hardware Specifications

4.6.1 Model Performance Comparison

N-gram Models

The performance of the 2-gram models and their enhancement strategies are shown in Figure 4.4. The best results for 2-grams was a combination of strategies between lower casing and cut off with a value of two.

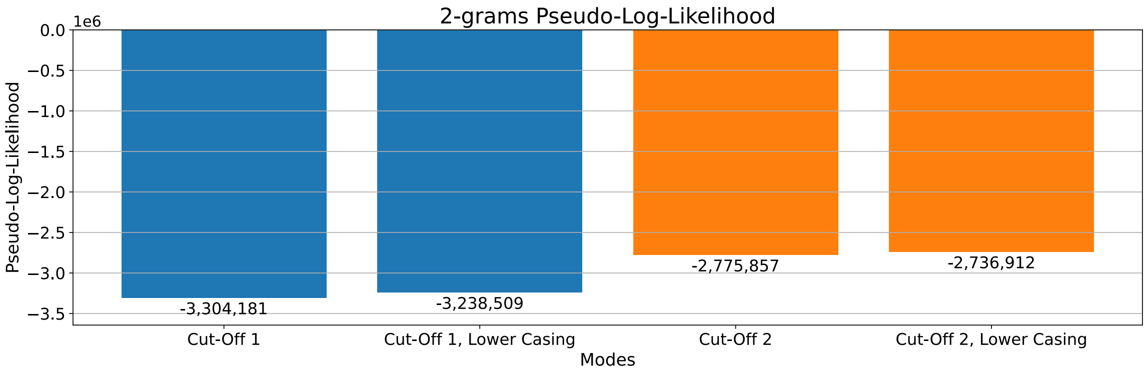

Figure 4.4: 2-gram Models

Figure 4.5 shows the performance of the 3-gram models, as well as their respective boosting alternatives. The 3-gram model with a combination of lower casing and cut off with a value of two shows the best performance.

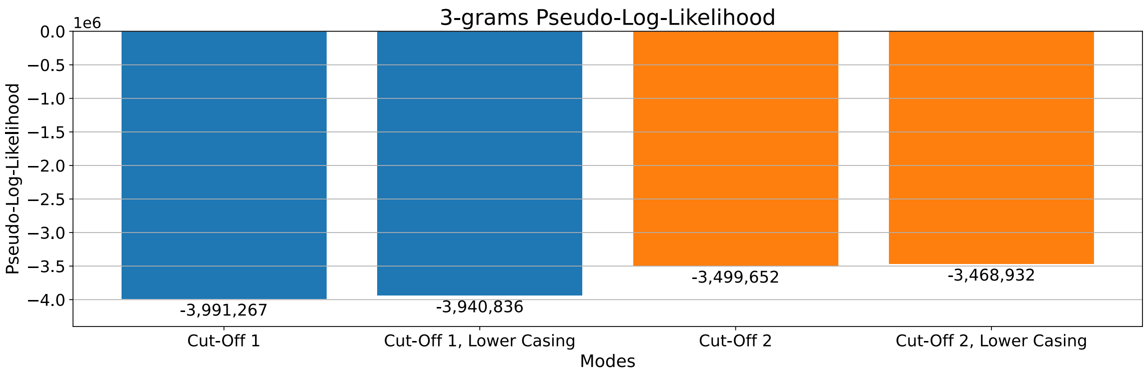

Figure 4.5: 3-gram Models

Figure 4.6 illustrates the performance of the n-gram models with backoff strategies.

The model showing best performance employs a cutoff value of 2, applies lower casing, and utilizes a backoff strategy that selects the maximum probability between the trigram or reverting to the lower-order bigram.

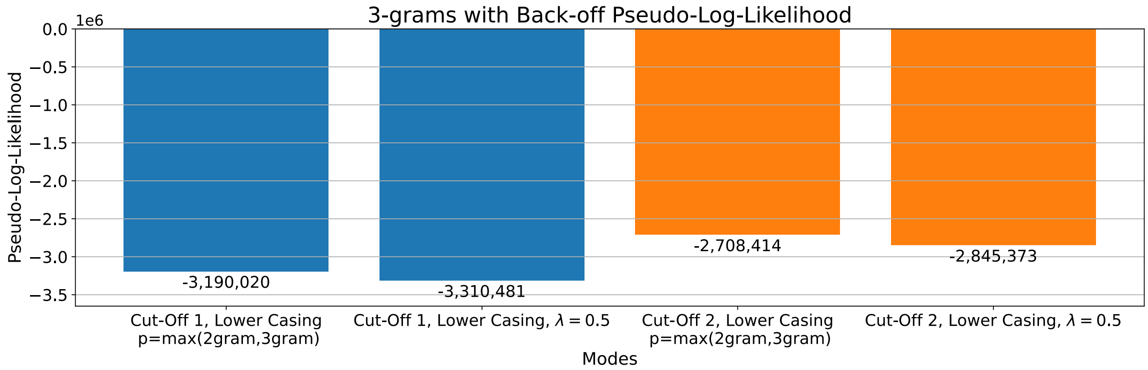

Figure 4.6: 3gram Models with Back-off strategies

Table 4.11 shows a comparison between the n-gram models that make use of back-off strategies that can revert to lower-order n-grams or take a weighted average defined by λ, n-gram models different cut-off values in order to deal with OOV, n-gram models with lower casing strategies and finally models combining different strategies. The best performing n-gram model makes use of cut-off, lower casing and a back-off strategy of reverting to the highest probability model.

N-gram	Strategies	PLL
2-gram	Cut-Off = 1	-3,304,181
2-gram	Cut-Off = 1 Lower Casing	-3,238,509
2-gram	Cut-Off = 2	-2,775,857
2-gram	Cut-Off = 2 Lower Casing	-2,736,912
3-gram	Cut-Off = 1 Lower Casing Back-Off : max(2gram,3gram)	-3,190,020
3-gram	Cut-Off = 1 Lower Casing Back-Off : $\lambda = 0.5$	-3,310,481
3-gram	**Cut-Off = 2** **Lower Casing** **Back-Off : max(2gram,3gram)**	**-2,708,414**
3-gram	Cut-Off = 2 Lower Casing Back-Off : $\lambda = 0.5$	-2,845,373

-

Table 4.11: N-gram models comparison

RoBERTa Models

The bar chart presented in Figure 4.7 shows the performance of the different RoBERTa models. These models share the same hyperparameters and only vary among their vocabulary sizes between 80% and 90%. The models demonstrating optimal performance possess vocabularies ranging from 12,095 to 25,550 tokens, with the very best model containing a vocabulary of 18,822 tokens.

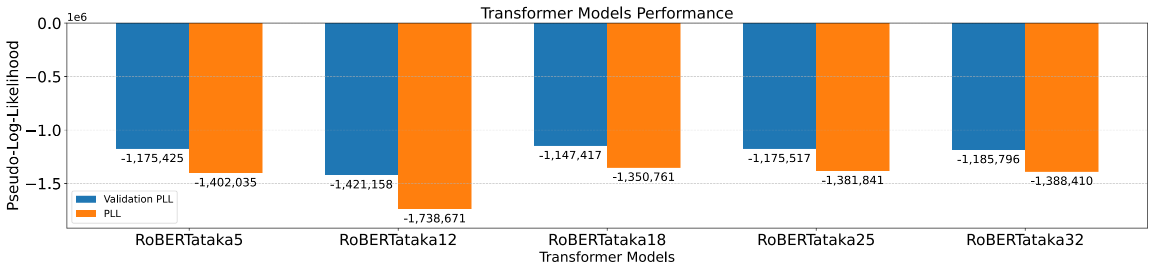

Figure 4.7: Trained RoBERTa Models Vocabulary Target Size Comparison

Final Comparison

A final comparison between the best performing n-gram and RoBERTa models is shown in Figure 4.8. The best performing n-gram models share the lower-casing strategy as well as a cut-off value of two. In addition to the previous settings, the best performing n-gram model makes use of the $max(bigram, trigram)$ back-off strategy. On the right side, the performance of the 3 best RoBERTa models is shown, going from 18 thousand to 32 thousand tokens in their vocabulary. The overall best-performing language model is a RoBERTa based model with slightly above 18 thousand tokens within its vocabulary.

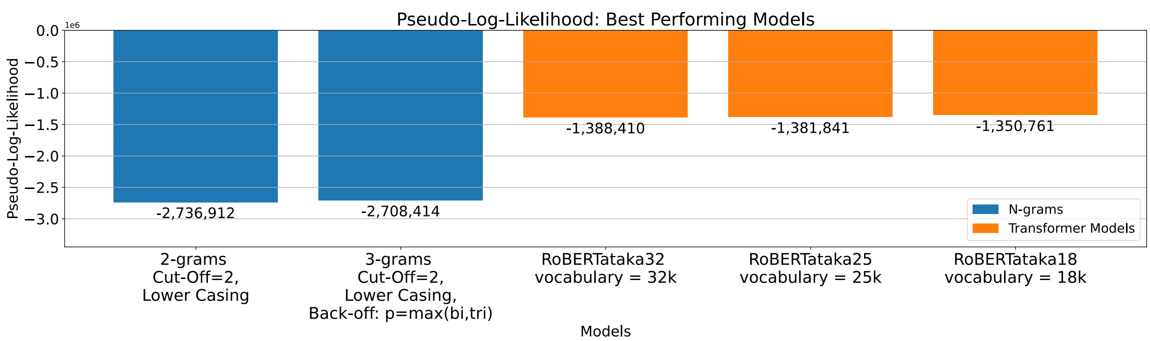

Figure 4.8: Trained Models Performance Comparison

4.7 Final Remarks

In this chapter different evaluation metrics for language models were presented. These metrics assess model skills for classification, text alignment or the models ability to response within a provided context. The discussion of the existence of benchmarks that evaluate models performance across the previous mentioned domains like GLUE or BLUE, which are not applicable to Purépecha due to the lack of annotated corpora.

It was also discussed the main evaluation metrics for TS like PPPL and PLL and how these metrics offer an alternative for task-based evaluations while being able to measure the models' ability to capture probability distributions, offering insights into their linguistic comprehension. Later on the discussion focused on how the PLL metric can be directly applied to both n-gram based models and BERTs based models while using whole-word-masking instead of subword masking to avoid compatibility issues with n-grams.

The evaluation compared two distinct modeling approaches: n-gram models and transformer-based RoBERTa models. For n-gram models, techniques such as Laplace smoothing, lower-casing, and back-off strategies (e.g., maximum probability selection between bigram and trigram) addressed challenges like OOV and rare n-gram combinations. RoBERTa based models were trained from scratch with vocabulary sizes ranging from 5,367 to 32,278 tokens. Although the n-grams used a variety of techniques their performance was greatly surpassed by those based on RoBERTa. It is also worth noticing how the Byte-level BPE tokenizer, proved effective in handling Purépecha's agglutinative morphology.

Chapter 5

Conclusions

5.1 General Conclusions

The recent growth and advance in artificial intelligence models demands large amount of information, this has led to models developing great performance in language tasks. However, models are data dependent. Datasets and corpora are essential for training models. The information for training models can be gathered from different sources and is often an automated task that benefits languages with a larger speaker base or digital presence while eventually leaving other languages underrepresented.

Despite Mexico's rich linguistic and cultural heritage, preservation efforts are needed. A few data sets exist for America's indigenous languages, and in Michoacan, home to a unique language Purépecha with great oral tradition and over 120,000 speakers yet lacks an available dataset. These examples illustrate several issues shared between indigenous languages in México where limited documentation and linguistic erosion are common.

In response to these challenges, this thesis presents the development of a multi-domain corpus for the Purépecha language. A methodology was proposed to gather and process text documents. The dataset preparation process underscored the importance of data quality, with careful cleaning to mitigate issues from OCR errors and duplicates.

This work contributes to the broader effort of developing language technologies for indigenous languages, addressing the gap in resources for Purépecha. The trained RoBER-

Tataka models, their tokenizers and both the raw and curated datasets will be publicly accessible. By making these resources openly available, this thesis supports not only the advancement of computational tools for Purépecha but also its preservation.

5.2 Future Work

Future work for the Purépecha corpus should focus on expanding corpora in a way which reflects a more common use case for the majority of native speakers. Documents inside the corpus could be tagged depending the geographic region or town in order to track regionalisms, slang or new words. Create a web platform to crowd source more documents in order to increment the overall size.

Manually correct the OCR corpus and annotate its documents in order so OCR models can be fine tuned to enlarge the corpus.

Create a "Purepecha GLUE" benchmark by tagging the corpus. This involves adding labels for tasks like sentence similarity, named entity recognition, question answering, and sentiment analysis.

Recording Audio for Text-to-Speech (TTS) and Speech-to-Text (STT) Models to Aid Pronunciation by recording the corpus text with help of native speakers it would create an audio-aligned dataset for training TTS and STT models, directly supporting students and revitalization programs.

Compose a multilingual or parallel corpus for Neural Machine Translation tasks, in order to later train multilingual models that can help with everyday tasks such as translation of government bulletins or official documents or even books from Spanish to Purépecha or even create multilingual models between indigenous languages.

Finally, the corpus can also be used in NLP competitions for training, developing, testing different models and tokenizers in order to perform a given task promoting interest in LRLs, fostering innovation and collaboration.

Bibliography

[Aarts93] Aarts, J., de Haan, P., y Oostdijk, N. *English language corpora: design, analysis and exploitation.* Rodopi Amsterdam, 1993.

[Alberto Medina20] Alberto Medina, A. M. La evolución del alfabeto p'urhépecha, 2020. Data retrived from : `https://www.idiomas.umich.mx/cadi/talleres/docs/alfabeto_purepecha.pdf`.

[Atkins92] Atkins, S., Clear, J., y Ostler, N. Corpus design criteria. *Literary and Linguistic Computing*, 7(1):1–16, ene. 1992. doi:10.1093/LLC/7.1.1. URL `http://www.natcorp.ox.ac.uk/archive/vault/tgaw02.pdf`

[Bagla21] Bagla, K., Kumar, A., Gupta, S., y Gupta, A. Noisy text data: Achilles' heel of popular transformer based nlp models, 2021. URL `https://arxiv.org/abs/2110.03353`

[Blázquez15] Blázquez, D. P. La labor científica de fray jerónimo de alcalá, ofm: Una etnografía misionera del siglo xvi. *Mutatis Mutandis. Revista Latinoamericana de Traducción*, 2015. URL `https://www.redalyc.org/articulo.oa?id=499267768003`

[Brants07] Brants, T., Popat, A. C., Xu, P., Och, F. J., y Dean, J. Large language models in machine translation. *En* J. Eisner, ed., *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, págs. 858–867. Association for Computational Linguistics,

Prague, Czech Republic, jun. 2007.
URL https://aclanthology.org/D07-1090

[Campbell86] Campbell, L., Kaufman, T., y Smith-Stark, T. C. Mesoamerica as a linguistic area. *Language*, 62(3), Sep 1986. doi: 10.1353/LAN.1986.0105.
URL https://posgrado.unam.mx/mesoamericanos/uploads/docs/02.%20Campbell,%20Kaufman%20&%20Smith-Stark,%20T_Mesoamerica%20as%20a%20Linguistic%20Area%20(1986).pdf

[Capistran15] Capistran, A. *Multiple object constructions in P'orhépecha*. Brill's Studies in the Indigenous Languages of the Americas. Brill, feb. 2015.

[Cer17] Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., y Specia, L. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *En Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, 2017. doi:10.18653/v1/s17-2001.
URL http://dx.doi.org/10.18653/v1/S17-2001

[Chai23] Chai, C. P. Comparison of text preprocessing methods. *Natural Language Engineering*, 29(3):509–553, 2023. doi:10.1017/S1351324922000213.

[Chakravarthi23] Chakravarthi, B. R., Sripriya, N., Bharathi, B., Nandhini, K., Navaneethakrishnan, S. C., Durairaj, T., Ponnusamy, R., Kumaresan, P. K., Ponnusamy, K. K., y Rajkumar, C. Overview of the shared task on sarcasm identification of dravidian languages (malayalam and tamil) in dravidiancodemix. *En Forum of Information Retrieval and Evaluation FIRE-2023*. 2023.

[Challa20] Challa, H., Niu, N., y Johnson, R. Faulty requirements made valuable: On the role of data quality in deep learning. *En 2020 IEEE*

Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), págs. 61–69. 2020. doi:10.1109/AIRE51212.2020.00016.

[Chamoreau23] Chamoreau, C. Purepecha: An isolate non-Mesoamerican language in Mesoamerica. *En* S. Wichmann, ed., *The Languages of Middle America: A Comprehensive Guide*. Mouton de Gruyter, 2023.
URL `https://shs.hal.science/halshs-03880946`

[DataCanary17] DataCanary, hilfialkaff, Jiang, L., Risdal, M., Dandekar, N., y tomtung. Quora question pairs. `https://kaggle.com/competitions/quora-question-pairs`, 2017. Kaggle.

[Devlin19] Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[Dressel16] Dressel, F. RPubs - Distribution of n-grams in english text corpus — rpubs.com. `https://rpubs.com/fdd/187848`, 2016. [Accessed 04-02-2025].

[Dubey22] Dubey, S. R., Singh, S. K., y Chaudhuri, B. B. Activation functions in deep learning: A comprehensive survey and benchmark, 2022.
URL `https://arxiv.org/abs/2109.14545`

[FILAC20] FILAC, ed. *Revitalización de Lenguas Indígenas*. Fondo para el Desarrollo de los Pueblos Indígenas de América Latina y El Caribe, La Paz, Bolivia, Dec 2020.
URL `https://www.ordpi.org/wp-content/uploads/2021/03/Informe-Regional-Lenguas-Indigenas-Derechos-Intercult-2.pdf`

[Francis64] Francis, W. N. y Kucera, H. The standard corpus of present-day edited american english (the brown corpus). Inf. téc., Department of

Linguistics, Brown University, Providence, Rhode Island, US, 1964.
URL http://icame.uib.no/brown/bcm.html

[Fruehwald22a] Fruehwald, J. Lin517: Natural language processing. https://jofrhwld.github.io/teaching/courses/2022_lin517/, 2022. Accessed: 2025–02-14.

[Fruehwald22b] Fruehwald, J. Natural Language Processing Course, 2022. University of Kentucky.
URL https://lin611-2024.github.io/

[Gutierrez-Vasques15] Gutierrez-Vasques, X. Bilingual lexicon extraction for a distant language pair using a small parallel corpus. *En* D. Inkpen, S. Muresan, S. Lahiri, K. Mazidi, y A. Zhila, eds., *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, págs. 154–160. Association for Computational Linguistics, Denver, Colorado, jun. 2015. doi:10.3115/v1/N15-2021.
URL https://aclanthology.org/N15-2021

[Gutierrez15] Gutierrez, X., Vilchis Vargas, E. C., y Cerbón Ynclán, R. Recopilación de un corpus paralelo electrónico para una lengua minoritaria: el caso del español-náhuatl. *Congreso Internacional el Patrimonio Cultural y las Nuevas Tecnologías*, 1, 2015.
URL http://www.pcnt.inah.gob.mx/pdf/14290504602.pdf

[Hu24] Hu, T. y Zhou, X.-H. Unveiling llm evaluation focused on metrics: Challenges and solutions, 2024.
URL https://arxiv.org/abs/2404.09135

[Hug24a] Hugging Face. *Natural Language Processing Tasks*, 2024. Available at https://huggingface.co/tasks.

[Hug24b] HuggingFace. *Building a Neural Network Classifier from the Ground Up: A Step-by-Step Guide*, 2024.

[IBM21] IBM. *Best practices on OCR with Content Analyzer*, 2021. Available at https://www.ibm.com/docs/en/bacaoc?topic=solutions-best-practices-ocr-content-analyzer.

[IBM24] IBM. *What is gradient descent?*, 2024. Available at https://www.ibm.com/think/topics/gradient-descent.

[INALI09] INALI. Catálogo de las lenguas indígenas nacionales: variantes lingüísticas de méxico con sus autodenominaciones y referencias geoestadísticas, 2009. Data retrived from : https://www.inali.gob.mx/pdf/CLIN_completo.pdf.

[INEGI09] INEGI. Perfil sociodemográfico de la población que habla lengua indígena, 2009. Data retrived from : https://en.www.inegi.org.mx/contenidos/productos/prod_serv/contenidos/espanol/bvinegi/productos/censos/poblacion/poblacion_indigena/leng_indi/PHLI.pdf.

[INEGI20] INEGI. Lenguas indígenas y hablantes de 3 años y más, 2020. Data retrived from INEGI, https://cuentame.inegi.org.mx/hipertexto/todas_lenguas.htm.

[INJ24] INJOSOFT. *Reference to ASCII Table*, 2024. Available at https://www.ascii-code.com/, Accessed: 2024–10-14.

[Jiang24] Jiang, M., Liu, K. Z., Zhong, M., Schaeffer, R., Ouyang, S., Han, J., y Koyejo, S. Investigating data contamination for pre-training language models, 2024.
URL https://arxiv.org/abs/2401.06059

[Joshi20] Joshi, P., Santy, S., Budhiraja, A., Bali, K., y Choudhury, M. The state and fate of linguistic diversity and inclusion in the NLP world.

En D. Jurafsky, J. Chai, N. Schluter, y J. Tetreault, eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, págs. 6282–6293. Association for Computational Linguistics, Online, jul. 2020. doi:10.18653/v1/2020.acl-main.560.
URL `https://aclanthology.org/2020.acl-main.560/`

[Jurafsky24] Jurafsky, D. y Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models.* Prentice Hall PTR, 3ª edón., 2024. Online manuscript released August 20, 2024.
URL `https://web.stanford.edu/~jurafsky/slp3/`

[KAGGLE24] KAGGLE. Kaggle, 2024. No results found.
URL `https://www.kaggle.com/datasets?search=purepecha`

[Kasper Welbers17] Kasper Welbers, W. V. A. y Benoit, K. Text analysis in r. *Communication Methods and Measures*, 11(4):245–265, 2017. doi:10.1080/19312458.2017.1387238.

[Kauf23] Kauf, C. y Ivanova, A. A better way to do masked language model scoring, 2023. Url available at `https://arxiv.org/abs/2305.10588`.

[Khatter22] Khatter, K. Natural language processing: state of the art, current trends and challenges. *springer*, 82(3), Jul 2022. doi:10.1007/s11042-022-13428-4.

[Kornai07] Kornai, A. *Mathematical linguistics.* Springer Science & Business Media, 2007.

[Kouzis-Loukas16] Kouzis-Loukas, D. *Learning Scrapy.* Packt Publishing Ltd, 2016.

[Liedtke01] Liedtke, S. Comparación de lÉxico tarasco-quechua. *Universidad Nacional Mayor de San Marcos*, 2001. ISSN 1609-9036.
URL `https://sisbib.unmsm.edu.pe/BibVirtualData/`

publicaciones/Alma_Mater/N%C2%BA20_2001/pdf/comparacion_lexico.pdf

[Liu19] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., y Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[Lou24] Lou, A., Pérez-Ortiz, J. A., Sánchez-Martínez, F., y Sánchez-Cartagena, V. M. Curated datasets and neural models for machine translation of informal registers between mayan and spanish vernaculars, 2024.
URL https://arxiv.org/abs/2404.07673

[Magueresse20] Magueresse, A., Carles, V., y Heetderks, E. Low-resource languages: A review of past work and future challenges, 2020.
URL https://arxiv.org/abs/2006.07264

[Manning99] Manning, C. D. y Schutze, H. *Foundations of statistical natural language processing.* The MIT Press. MIT Press, London, England, mayo 1999.

[Manning08] Manning, C. D., Raghavan, P., y Schutze, H. *Introduction to Information Retrieval.* Cambridge University Press, Cambridge, England, jul. 2008.

[Marcus00] Marcus, J. Los calendarios prehispánicos. *Arqueología Mexicana*, 7(41):12–19, Jan 2000.
URL https://arqueologiamexicana.mx/calendarios/los-numerales

[Medina Perez70] Medina Perez, A. *Juchari Uandákuecha -Nueva cartilla.* Morevallado, 1ª edón., 1970.

[Medina Perez06] Medina Perez, A. *Guía Lingüistica del Idioma P'urhépecha*. Morevallado, 2ª edón., 2006.

[Mitchell15] Mitchell, R. *Web Scraping with Python: Collecting Data from the Modern Web*. O'Reilly Media, Inc., 1ª edón., 2015. ISBN 1491910291.

[Newman05] Newman, M. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351, sep. 2005. ISSN 1366-5812. doi: 10.1080/00107510500052444.
URL `http://dx.doi.org/10.1080/00107510500052444`

[OSCAR23] OSCAR. Open super-large crawled aggregated corpus, 2023. No results found.
URL `https://oscar-project.org/`

[Peng19] Peng, Y., Yan, S., y Lu, Z. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets, 2019.
URL `https://arxiv.org/abs/1906.05474`

[Polshcha24] Polshcha, M. pdfreader: A pythonic api for accessing and extracting data from pdf documents, 2024.
URL `https://github.com/maxpmaxp/pdfreader`

[Project24] Project, E. L. Purépecha, 2024.
URL `https://www.endangeredlanguages.com/lang/7569`

[Raschka15] Raschka, S. *Python Machine Learning*. Pakt, sep. 2015.
URL `https://github.com/rasbt/python-machine-learning-book-3rd-edition`

[Ravichandiran21] Ravichandiran, S. *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*. Packt Publishing, Birmingham, England, 2021.

[Reppen10] Reppen, R. *Using Corpora in the Language Classroom*. UNIVERSITY OF MICHIGAN PRESS, Mar 2010.

[Richardson07] Richardson, L. Beautiful soup documentation. *April*, 2007.

[Russell22] Russell, S. y Norvig, P. *Artificial Intelligence: A Modern Approach*. Pearson, ene. 2022.
URL https://aima.cs.berkeley.edu/global-index.html

[Sadaiyandi23] Sadaiyandi, J., Arumugam, P., Sangaiah, A., y Zhang, C. Stratified sampling-based deep learning approach to increase prediction accuracy of unbalanced dataset. *Electronics*, 12:4423, 10 2023. doi:10.3390/electronics12214423.

[Salazar20] Salazar, J., Liang, D., Nguyen, T. Q., y Kirchhoff, K. Masked language model scoring. *En Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. doi:10.18653/v1/2020.acl-main.240.
URL http://dx.doi.org/10.18653/v1/2020.acl-main.240

[Sennrich16] Sennrich, R., Haddow, B., y Birch, A. Neural machine translation of rare words with subword units, 2016.
URL https://arxiv.org/abs/1508.07909

[Sharma14] Sharma, M. Selenium tool: A web based automation testing framework. *En International Journal of Emerging Technologies in Computational and Applied Sciences*. 2014.
URL https://api.semanticscholar.org/CorpusID:18049611

[Sin24] Single Store. *Understanding the Softmax Activation Function: A Comprehensive Guide*, 2024.

[Stap23] Stap, D. y Araabi, A. Chatgpt is not a good indigenous translator. *Proceedings of the Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP)*, 2023.
URL https://api.semanticscholar.org/CorpusID:259833859

[Tiedemann12] Tiedemann, J. Parallel data, tools and interfaces in OPUS. *En* N. Calzolari, K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, y S. Piperidis, eds., *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, págs. 2214–2218. European Language Resources Association (ELRA), Istanbul, Turkey, mayo 2012.
URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf

[Tonja23a] Tonja, A., Destaw Belay, T., Azime, I. A., Ayele, A. A., Mehamed, M. A., Kolesnikova, O., y Yimam, S. Natural language processing in ethiopian languages: Current state, challenges, and opportunities. *ACL*, abs/2303.14406, Mar 2023. doi:10.48550/arXiv.2303.14406.

[Tonja23b] Tonja, A. L., Maldonado-sifuentes, C., Mendoza Castillo, D. A., Kolesnikova, O., Castro-Sánchez, N., Sidorov, G., y Gelbukh, A. Parallel corpus for indigenous language translation: Spanish-mazatec and Spanish-Mixtec. *En* M. Mager, A. Ebrahimi, A. Oncevay, E. Rice, S. Rijhwani, A. Palmer, y K. Kann, eds., *Proceedings of the Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP)*, págs. 94–102. Association for Computational Linguistics, Toronto, Canada, jul. 2023. doi:10.18653/v1/2023.americasnlp-1.11.
URL https://aclanthology.org/2023.americasnlp-1.11

[Tonja24] Tonja, A. L., Balouchzahi, F., Butt, S., Kolesnikova, O., Ceballos, H., Gelbukh, A., y Solorio, T. Nlp progress in indigenous latin american languages, 2024.

[Torfi21] Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavaf, N., y Fox, E. A. Natural language processing advancements by deep learning: A survey, 2021.

[Torruella99] Torruella, J. y Llisterri, J. Diseño de corpus textuales y orales. *Filología e informática. Nuevas tecnologías en los estudios filológicos*, págs. 45–77, 1999.

[Tsvetkov17] Tsvetkov, Y. Opportunities and challenges in working with low-resource languages, June 2017.

[Tyers23] Tyers, F., Pugh, R., y Berthoud F., V. Codex to corpus: Exploring annotation and processing for an open and extensible machine-readable edition of the florentine codex. *En* M. Mager, A. Ebrahimi, A. Oncevay, E. Rice, S. Rijhwani, A. Palmer, y K. Kann, eds., *Proceedings of the Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP)*, págs. 19–29. Association for Computational Linguistics, Toronto, Canada, jul. 2023. doi:10.18653/v1/2023.americasnlp-1.4.
URL `https://aclanthology.org/2023.americasnlp-1.4`

[UNAM23] UNAM. Corpus paralelo de lenguas mexicanas, 2023. No results found.
URL `http://www.corpus.unam.mx/geco/portal/index/cplm`

[Uni24] University of Illinois Urbana-Champaign. *Introduction to OCR and Searchable PDFs*, 2024. Available at `https://guides.library.illinois.edu/OCR/bestpractices`.

[V721] V7. *Activation Functions in Neural Networks*, 2021.

[Vaswani23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I. Attention is all you need, 2023.

[Villena20] Villena, F., Eisenmann, U., Knaup, P., Dunstan, J., y Ganzinger, M. On the construction of multilingual corpora for clinical text mining.

Studies in health technology and informatics, 270:347–351, 06 2020. doi:10.3233/SHTI200180.

[Wang18] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., y Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *En* T. Linzen, G. Chrupała, y A. Alishahi, eds., *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, págs. 353–355. Association for Computational Linguistics, Brussels, Belgium, nov. 2018. doi:10.18653/v1/W18-5446.
URL `https://aclanthology.org/W18-5446`

[Wang19] Wang, C., Cho, K., y Gu, J. Neural machine translation with byte-level subwords, 2019.
URL `https://arxiv.org/abs/1909.03341`

[Ward18] Ward, M. Qualitative research in less commonly taught and endangered language call. *Language Learning and Technology*, 22(2), Jun 2018.

[Warren00] Warren, J. B. *Fray Jerónimo de Alcalá, autor de la Relación de Michoacán*. Colegio de Michoacán-Gobierno del Estado de Michoacán, 2000.

[Warstadt20] Warstadt, A., Parrish, A., Liu, H., Mohananey, A., Peng, W., Wang, S.-F., y Bowman, S. R. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020. doi:10.1162/tacl_a_00321.
URL `https://doi.org/10.1162/tacl_a_00321`

[Werbos82] Werbos, P. J. Applications of advances in nonlinear sensitivity analysis. *En* R. F. Drenick y F. Kozin, eds., *System Modeling and Optimization*, págs. 762–770. Springer Berlin Heidelberg, Berlin, Heidelberg, 1982. ISBN 978-3-540-39459-4.

[Wu16] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., y Dean, J. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[Zhang13] Zhang, C., Baldwin, T., Ho, H., Kimelfeld, B., y Li, Y. Adaptive parser-centric text normalization. *En* H. Schuetze, P. Fung, y M. Poesio, eds., *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, págs. 1159–1168. Association for Computational Linguistics, Sofia, Bulgaria, ago. 2013.
URL `https://aclanthology.org/P13-1114`

Daniel Castro Pineda

A dissertation submitted for the degree of.pdf

Universidad Michoacana de San Nicolás de Hidalgo

Detalles del documento

Identificador de la entrega

trn:oid:::3117:487528416

Fecha de entrega

27 ago 2025, 10:16 a.m. GMT-6

Fecha de descarga

27 ago 2025, 10:45 a.m. GMT-6

Nombre del archivo

A dissertation submitted for the degree of.pdf

Tamaño del archivo

2.6 MB

114 páginas

19.862 palabras

108.051 caracteres

13% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 10 palabras)

Grupos de coincidencias

148 Sin cita o referencia 13%
Coincidencias sin una citación ni comillas en el texto

0 Faltan citas 0%
Coincidencias que siguen siendo muy similar al material fuente

0 Falta referencia 0%
Las coincidencias tienen comillas, pero no una citación correcta en el texto

0 Con comillas y referencia 0%
Coincidencias de citación en el texto, pero sin comillas

Fuentes principales

12% Fuentes de Internet
10% Publicaciones
0% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Formato de Declaración de Originalidad y Uso de Inteligencia Artificial

Coordinación General de Estudios de Posgrado
Universidad Michoacana de San Nicolás de Hidalgo

A quien corresponda,

Por este medio, quien abajo firma, bajo protesta de decir verdad, declara lo siguiente:

- Que presenta para revisión de originalidad el manuscrito cuyos detalles se especifican abajo.
- Que todas las fuentes consultadas para la elaboración del manuscrito están debidamente identificadas dentro del cuerpo del texto, e incluidas en la lista de referencias.
- Que, en caso de haber usado un sistema de inteligencia artificial, en cualquier etapa del desarrollo de su trabajo, lo ha especificado en la tabla que se encuentra en este documento.
- Que conoce la normativa de la Universidad Michoacana de San Nicolás de Hidalgo, en particular los Incisos IX y XII del artículo 85, y los artículos 88 y 101 del Estatuto Universitario de la UMSNH, además del transitorio tercero del Reglamento General para los Estudios de Posgrado de la UMSNH.

Datos del manuscrito que se presenta a revisión		
Programa educativo	Maestría en Ciencias en Ingeniería Eléctrica	
Título del trabajo	CORPUS COMPILATION AND NATURAL LANGUAGE MODEL EVALUATION FOR THE PURÉPECHA LANGUAGE	
	Nombre	**Correo electrónico**
Autor/es	Daniel Castro Pineda	2220220h@umich.mx
Director	Jaime Cerda Jacobo	jaime.cerda@umich.mx
Codirector	José Ortiz Béjar	jose.ortiz@umich.mx
Coordinador del programa	Norberto Garcia Barriga	norberto.garcia@umich.mx

Uso de Inteligencia Artificial		
Rubro	**Uso (sí/no)**	**Descripción**
Asistencia en la redacción	SI	Se realizaron consultas a LLM para la creación de plantillas para tablas , listas, figuras, corrección de errores en ecuaciones de LATEX y posicionamiento de elementos.
Traducción al español	NO	-

Formato de Declaración de Originalidad y Uso de Inteligencia Artificial

Coordinación General de Estudios de Posgrado
Universidad Michoacana de San Nicolás de Hidalgo

Uso de Inteligencia Artificial		
Rubro	**Uso (sí/no)**	**Descripción**
Traducción a otra lengua	NO	-
Revisión y corrección de estilo	SI	Se utilizaron las herramientas incrustadas de overleaf y writefull para sugerencias ortográficas y gramaticales.
Análisis de datos	NO	-
Búsqueda y organización de información	SI	Se utilizaron motores de búsqueda como Google Scholar y ResearchGate para encontrar artículos en base a palabras clave y temas específicos.
Formateo de las referencias bibliográficas	SI	Se utilizó la herramienta de typeset.io para automatizar y generar las entradas en formato bibtex de artículos y libros consultados.
Generación de contenido multimedia	NO	-
Otro	NO	-

Datos del solicitante	
Nombre y firma	Daniel Castro Pineda
Lugar y fecha	Morelia, Michoacán a 26 Agosto 2025