UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Etiquetador automático de la morfología del otomí usando predicción estructurada

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Diego Alberto Barriga Martínez

DIRECTOR DE TESIS

Dr. Victor German Mijangos de la Cruz

Ciudad Universitaria, Cd. Mx., 2022

A mis abuelos Francis y Ángel.
A mi madre Ángeles y a mi hermana Karen.
Su apoyo, cariño y amor forjaron los cimientos de lo que soy.

Agradecimientos

A mi familia, a mis amigas y amigos, por ser parte de mi vida y de lo que soy.

A mi novia, Karen, por todo el amor y por acompañarme en este emocionante camino.

A la Universidad Nacional Autónoma de México, mi alma mater. A la Facultad de Ingeniería, por el conocimiento y las personas con las que compartí charlas e ideas. Fueron fundamentales para mi desarrollo como profesional y persona.

Al Laboratorio de Investigación y Desarrollo de Software Libre, donde conocí excepcionales personas siempre ávidas por compartir y empujar las fronteras del conocimiento libre.

A mi director y directora de tesis, Dr. Victor Mijangos de la Cruz y Dra. Ximena Gutierrez-Vasques, por brindarme la oportunidad de incursionar en el apasionante mundo del Procesamiento del Lenguaje Natural, por la infinita paciencia y el cálido apoyo para el desarrollo y culminación de esta tesis.

Índice general

Índice de figuras

Índice de tablas

Talk, it's only talk
Arguments, agreements
Advice, answers
Articulate announcements
It's only talk

Elephant Talk - Discipline, King Crimson

1 Introducción

El Procesamiento del Lenguaje Natural (*Natural Language Processing, NLP*) es un área de la computación que permite reconocer, procesar e interpretar el lenguaje humano dentro de un sistema computacional. El objetivo de esta área es hacer que las computadoras realicen tareas que involucran el lenguaje humano. Algunas tareas consisten en permitir la comunicación humano-máquina, o en general hacer un procesamiento de texto o voz en lenguaje natural (Jurafsky y Martin, 2008). Ejemplos de aplicación actuales son los traductores automáticos, asistentes personales que reconocen voz, motores de búsquedas en Internet, análisis de sentimientos en textos, síntesis de voz, etiquetado de textos y muchas más aplicaciones. Abundaremos sobre el tema en la sección 2.1.

Para llevar a cabo estas tareas, los métodos de *NLP* actuales aplican técnicas de Aprendizaje de Máquina (*Machine Learning, ML*). El *ML* es una pieza fundamental de la Inteligencia Artificial (*Artificial Intelligence, AI*) y la Ciencia de Datos (*Data Science*), donde, una computadora estima soluciones a problemas a partir de cierta experiencia (Mitchell et al., 1997). La experiencia aquí se refiere a datos que permiten inferir un modelo estadístico de aprendizaje.

El *ML* consta de métodos generales que buscan encontrar patrones relevantes en los datos. Por ejemplo, si se analiza una biblioteca, se podrían

determinar los tópicos importantes presenten en los libros utilizando métodos de aprendizaje de máquina. Entre los métodos de *ML* más ampliamente utilizados se pueden mencionar las *Support Vector Machines (SVM)*, árboles de decisión, o bien los modelos basados en grafos, como las redes neuronales o los métodos generativos (modelos bayesianos), solo por mencionar algunos. Se profundizará sobre el tema en la sección 2.2.

Una de las tareas más populares del *NLP* es el etiquetado automático de textos. Esta tarea consta de asignar etiquetas a palabras dotándolas de información lingüística. Generalmente, para esta tarea se utilizan modelos secuenciales probabilísticos, por ejemplo, modelos ocultos de Markov (*Hidden Markov Models, HMM*) o redes recurrentes (*Recurrent Neural Networks, RNN).* Este etiquetado puede realizarse a diferentes niveles lingüísticos, por ejemplo, a nivel morfosintáctico (*Part Of Speech tagging, POS*), sintáctico (*parsing*), morfológico, entre otros. En la sección 2.3 detallaremos sobre los etiquetadores automáticos.

Hay un tipo de etiquetado, de gran importancia para el análisis lingüístico, llamado glosado o glosa interlineal. Para realizar el etiquetado, lingüistas analizan sentencias en un determinado lenguaje y asignan etiquetas manualmente a las unidades que conforman cada palabra de esa sentencia. Esto permite el análisis morfosintáctico de una lengua. Un ejemplo de un texto en otomí glosado[1] puede verse en la tabla 1.1.

Sentencia	hí	tó=tsogí
Glosa	NEG	3.PRF=dejar
Traducción	'No lo he dejado'	

Tabla 1.1: Ejemplo de glosa morfema a morfema de la frase "hí tótsogí" del otomí

1.1. Definición del problema

Este tipo de datos lingüísticos anotados son recursos valiosos no solo para la documentación lingüística, sino, también, para habilitar tecnologías de *NLP*, por ejemplo, proveyendo datos para analizadores morfológicos automáticos, etiquetadores, segmentación morfológica, etc. Sin embargo, no todas las

[1]El listado completo de etiquetas POS y etiquetas de glosa con sus significados puede consultarse en los apéndices A y B respectivamente.

lenguas tienen este tipo de anotaciones fácilmente disponible. El glosado manual es una tarea que requiere mucho tiempo y el expertiz de lingüistas. Esto es especialmente cierto con lenguas de bajos recursos digitales. Los bajos recursos digitales son un termino que denota aquellas lenguas que carecen de documentación y tecnologías del lenguaje (Mager et al., 2018). En este trabajo nos enfocamos en el otomí, una lengua indígena de bajos recursos digitales hablada en México (familia Oto-Mangue).

Lograr aplicar técnicas de *NLP* a lenguas indígenas mexicanas puede ser un paso para reducir la escasez de recursos digitales. Sin embargo, el lenguaje natural es dinámico y presenta fenómenos complejos. Adicionalmente, los bajos recursos digitales son un escenario particularmente retador para los métodos tradicionales de *NLP* que en general requieren de grandes cantidades de datos para funcionar correctamente. Los bajos recursos implican pocos datos iniciales para el entrenamiento de modelos de aprendizaje. Esto, a su vez, ocasiona que las predicciones sean poco precisas o equivocadas. En la sección 2.4.1 profundizaremos sobre las limitaciones de los métodos tradicionales.

Los bajos recursos digitales son un escenario común en México donde, a pesar de que existe una rica diversidad lingüística, gran parte de las lenguas indígenas mexicanas no poseen contenido web ni publicaciones digitales y, por tanto, carecen también de tecnologías del lenguaje. El desarrollo de tecnologías para el manejo computacional del lenguaje, que funcionen para lenguas de bajos recursos digitales, es una oportunidad y reto de investigación sumamente importante.

En este trabajo nos enfocamos en la construcción de un glosador automático para el otomí, en particular para el otomí de Toluca, una lengua de gran riqueza morfológica y que presenta escasez de recursos digitales. Nuestro recurso inicial es un pequeño corpus transcrito a un alfabeto fonético, pre-procesado y glosado manualmente. Una vez que tenemos este conjunto de datos lo utilizamos para entrenar un sistema automático de glosado para el otomí de Toluca. Implementamos algunas variaciones de un *framework* de aprendizaje estructurado llamados Campos Aleatorios Condicionales (*Conditional Random Fields, CRF*).

Los *CRFs* necesitan una representación vectorial por cada letra de las sentencias de entrada. Esto se realiza por medio de las *feature functions*. Para que las *feature functions* hagan un mapeo de las letras a vectores se debe considerar la información relevante de la lengua. Esta información está codificada en una lista de características (*feature lists*) como la letra actual,

las etiquetas *POS*, posición de la letra en la palabra, entre otras.

Este desarrollo puede ser una herramienta útil para reducir la carga de trabajo cuando se realiza glosado manual y, a su vez, podría tener un impacto en la documentación lingüística. Además, esto puede llevar a un incremento de recursos anotados para el otomí, los cuales pueden ser un punto de partida para el desarrollo de tecnologías de *NLP* que hoy en día todavía no están disponibles para esta lengua. Más aún, el glosado puede ser un primer paso para el desarrollo de más tecnologías del lenguaje; no solo para el otomí, que presenta un grado de extinción acelerada (Comisión Nacional para el Desarrollo Indígena)[2], sino para las 68 agrupaciones lingüísticas que se hablan en México.

1.2. Objetivo

El objetivo de este trabajo es producir glosa automática en un escenario de bajos recursos digitales. Nos enfocaremos en una variante del otomí[3] hablada en la región de San Andrés Cuexcontitlán, Toluca, Estado de México, a la que nos referimos como otomí de Toluca.

Para esto se diseñará e implementará un etiquetador morfológico basado en técnicas de *NLP* y *ML*. En particular, se hará énfasis en métodos de aprendizaje estructurado y supervisado. En concreto, se aplicarán *Conditional Random Fields (CRF)* para etiquetado morfológico automático del otomí.

1.2.1. Objetivos particulares

- Obtener del corpus en otomí glosado manualmente
- Normalizar y preprocesar del corpus para entrenamiento
- Crear una lista de características (*feature lists*) que a su vez servirán para la construcción de las *feature functions*, piezas fundamentales para la correcta generación de modelos por parte de los *CRFs*
- Utilizar e implementar el método *Linear-Chain CRF* para entrenamiento

[2]https://www.gob.mx/inpi/

[3]Variante autonombrada en otomí como hñatho

- Entrenar los CRF con diferentes parámetros y configuraciones para generar los modelos de aprendizaje
- Verificar el desempeño utilizando la técnica de *k-fold cross validation* considerando el *accuracy* como principal medida de desempeño, entre otras.
- Utilizar el modelo generado para, a su vez, generar etiquetas para oraciones en otomí nuevas.

1.2.2. Hipótesis

Para la tarea de etiquetado de textos, como el glosado automático, son ampliamente utilizados modelos bayesianos, como los *HMM* o de alta complejidad computacional, como las redes neuronales, entre otros.

En esta tesis comprobaremos la hipótesis que supone que, en la tarea de glosado automático, para una lengua de bajos recursos como el otomí de Toluca, es posible obtener un mejor desempeño utilizando métodos de aprendizaje estructurado como los *Linear-Chain CRFs* en contraste con modelos tradicionales como los *HMM*. Creemos que los *CRFs* aprovecharán mejor la información lingüística codificada en las *feature functions*, una pieza fundamental para el funcionamiento de los *CRFs*.

Talk, it's only talk
Babble, burble, banter
Bicker, bicker, bicker
Brouhaha, balderdash, ballyhoo
It's only talk
Back talk

Elephant Talk - Discipline, King Crimson

2 Marco teórico

En este capítulo se tratarán conceptos fundamentales del *Natural Language Processing (NLP)*, haciendo énfasis en los etiquetadores, y conceptos del *Machine Learning (ML)* con enfoque en los métodos de aprendizaje basados en grafos. También, se abordarán elementos de los *CRFs* que serán de utilidad para la comprensión de la arquitectura propuesta en este trabajo. Por último, se hará una breve revisión de los avances en los etiquetadores automáticos.

2.1. Natural Language Processing (NLP)

Dentro de las ciencias de la computación existe una subrama llamada Inteligencia Artificial (IA). Para este trabajo podemos decir que la IA[1] se enfoca en la construcción de sistemas o agentes que perciben su entorno, operan de forma autónoma, se adaptan al cambio y crean o persiguen objetivos para obtener el mejor resultado, y cuando hay incertidumbre, el mejor resultado

[1]Es preciso señalar que la definición de los objetivos y alcances de la IA es un tema del que aún no hay consenso dada la amplitud del área, la ambigüedad de conceptos como inteligencia y los diversos enfoques existentes como los centrados en el comportamiento o en el pensamiento.

esperado (Russell y Norvig, 2010).

Dentro de la IA existe una disciplina llamada Procesamiento de Lenguaje Natural, o en inglés *Natural Language Processing (NLP)*, que busca hacer que las computadoras puedan realizar una maniputación significativa del lenguaje natural en sus distintos niveles lingüísticos[2](Mijangos de la Cruz, 2020). En ese sentido, dependiendo de la aplicación o tarea a resolver, las técnicas de *NLP* se enfocan en uno o varios niveles lingüísticos. Por ejemplo, pude ir desde algo relativamente simple, como que la computadora pueda proveer al usuario información acerca del conteo de palabras en un texto, hasta niveles más abstractos como el conocimiento semántico del lenguaje, que involucra las connotaciones y los significados que los hablantes de una sociedad le dan a las palabras (Mijangos de la Cruz, 2020).

En otras palabras, el *NLP* se encarga del procesamiento de los fenómenos del lenguaje natural a través del tratamiento y análisis de textos o voz, permitiendo que las computadoras realicen tareas que involucren elementos propios del lenguaje (Jurafsky y Martin, 2008). *NLP* tiene dos grandes enfoques; el orientado a voz y el orientado a texto.

Debido a que el tratamiento del lenguaje es complejo para las computadoras, estas requieren la intervención de múltiples disciplinas, por lo que el *NLP* se convierte en un área multidisciplinaria. Ciencias de la computación, psicología, ingeniería, matemáticas, filosofía, ciencias cognitivas y lingüística, por mencionar algunas, son áreas del conocimiento relevantes para este campo de estudio.

Esto obliga a recordar el trabajo de Turing (1950) dónde se plantea la pregunta *¿Pueden las máquinas pensar?* y se propone un ejercicio práctico, que después sería conocido como la prueba de Turing. Hoy en día aún es difícil saber si las computadoras pueden emular el procesamiento del lenguaje de la mente, por lo que estos conceptos son abordados por las ciencias cognitivas y la filosofía. Comprender cómo es la representación de los elementos del lenguaje y cómo sucede su procesamiento en la mente humana podría suponer avances para que las computadoras hagan lo propio.

Para el desarrollo de métodos de *NLP* se necesita de una relación estrecha entre la lingüística y la ingeniería en computación. La lingüística se encarga del estudio de la lengua en todas sus manifestaciones. Tradicionalmente, se habla de diferentes niveles de la lengua que ayudan a su estudio, por ejemplo, la fonología, la morfología, la sintaxis, la semántica, la prag-

[2]1. fonético y fonológico, 2. morfológico, 3. morfosintáctico, 4. sintáctico, 5. semántico

mática. La lingüística debe proveer los conocimientos fundamentales sobre la lengua, de tal forma que, permita una representación formal de la misma. Esta representación formal posibilita el procesamiento del lenguaje por una computadora.

La relación entre lingüística e ingeniería en computación cobran un papel importante para el *NLP* (por ello el área también es conocida como lingüística computacional)[3]. En la figura 2.1 muestra la intersección de disciplinas que hacen posible el desarrollo del *NLP* o lingüística computacional. Es importante recordar que mucho de los avances recientes han sido impulsados por el creciente poder computacional para el manejo de grandes cantidades de información.

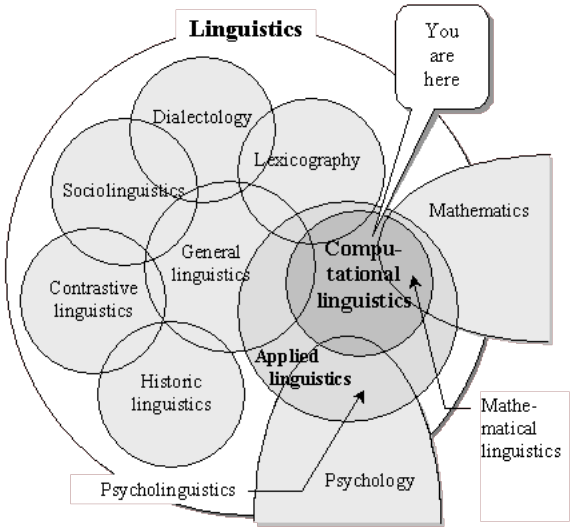

Figura 2.1: Estructura de la ciencia lingüística (Bolshakov y Gelbukh, 2004)

Los primeros métodos de *NLP* que trataron de lidiar con el lenguaje se basaron en modelos formales como máquinas de estados finitos, modelos basados en reglas y sistemas apoyados de lógica formal. Posteriormente, comenzaron a aplicarse los modelos estadísticos, en particular los modelos de aprendizaje de máquina.

[3]Para ciertos autores existe una distinción entre lingüística computacional y *NLP*. En este trabajo los consideramos sinónimos. Para profundizar en esta distinción ver: (Jurafsky y Martin, 2008) y (Manning y Schutze, 1999)

Dentro de los enfoques basados en reglas, uno de los trabajos fundacionales pueden atribuirse a Chomsky (1956), donde considera por primera vez a los autómatas finitos como una forma de caracterizar una gramática, y define un lenguaje finito como un lenguaje generado por una gramática finita (finite-state grammar). Estos primeros avances y los de otros fueron enfocados en la generación manual de reglas lingüísticas que intentaban modelar el lenguaje humano.

Por otra parte, se comenzó a implementar un enfoque estadístico que tiene raíces en diversos trabajos, uno de los más importantes, el de Shannon (1948) basado en la teoría de la información. En estos modelos se recogen conceptos como canal ruidoso y decodificación para la transmisión del lenguaje a través de un medio. Se introduce el concepto de entropía como forma de medir la capacidad de un canal para transmitir información, o la información contenida por un lenguaje (Jurafsky y Martin, 2008).

Con el paso de los años, el poder de cómputo y el acceso a grandes cantidades de datos propiciaron que los métodos estadísticos, específicamente el uso de métodos de *ML*, cobraran gran relevancia. En este trabajo nos enfocaremos en el uso de estos métodos estadísticos. En la sección 2.2 se abundará más sobre el *ML* y sus diversos enfoques.

La adopción del enfoque estadístico ha permitido el desarrollo de diversas aplicaciones de *NLP*. Parafraseando a Jurafsky y Martin (2008) una de estas aplicaciones es la traducción automática de textos o *machine traslation*. En esta tarea se busca traducir un documento completo o fragmento de un idioma a otro de forma automatizada.[4].

Otra aplicación muy utilizada está relacionada con los motores de búsqueda en Internet y son los sistemas que responden preguntas. Esta es una generalización de búsquedas simples en donde, en lugar de escribir palabras clave, se pueden hacer preguntas, más o menos complejas, expresadas en lenguaje natural tales como: ¿Qué significa la palabra "afable"?, ¿Cuántos estados tiene México? y ¿Qué opina la comunidad científica sobre la eugenesia?

Algunas de estas preguntas son más fáciles de responder que otras. Las preguntas sobre definiciones o hechos ya pueden ser respondidas actualmente por motores de búsqueda (Raghavan, 2021). Otras preguntas son más complejas de responder porque requieren que una computadora pueda inferir,

[4]Actualmente la aplicación más popular de traducción automática llamada Google Translate cuenta con un billón de instalaciones solo en dispositivos móviles (Pitman, 2021)

resumir y sintetizar información de múltiples fuentes. Los sistemas de *NLP* requieren diferentes capacidades para hacer frente a estos retos que incluyen: extracción de información, desambiguación del significado de palabras, reconocimiento de entidades, entre otras.

Aunque actualmente utilizamos ampliamente aplicaciones impulsadas por técnicas de *NLP*, hay muchos retos en la resolución de este tipo de tareas. La mayoría de aplicaciones son muy complicadas, ya que, a diferencia de otros sistemas de procesamiento de datos, las aplicaciones de *NLP* utilizan conocimientos del lenguaje. Por mencionar un caso, se necesita aportar cierto conocimiento lingüístico para resolver la ambigüedad que pueda presentarse en las sentencias de un texto. La ambigüedad se refiere a que una frase puede tener múltiples significados dependiendo de factores como el contexto. Por ejemplo, en la sentencia "El hombre ve a la mujer con el telescopio" no queda claro quien tiene el telescopio ¿El hombre o la mujer?

A pesar del gran reto que representa que una computadora logre un correcto procesamiento del lenguaje y sus fenómenos, gracias a los avances en investigación, es innegable que hoy en día muchas herramientas de *NLP* están presentes y facilitan la vida diaria, por ejemplo, la síntesis y reconocimiento automático de voz, lectores de pantallas, digitalizadores de documentos, clasificadores de documentos, generadores de textos y etiquetadores automático de textos. Este trabajo se centrará en el etiquetado automático de textos.

El etiquetado automático de textos consiste en asignar etiquetas o categorías que deberían llevar las partes de un texto, sean estas palabras, partes de palabras o frases completas. Se utiliza para realizar análisis lingüísticos y, en *NLP*, agregar información no explicita en el texto que puede aporta conocimiento lingüístico a la computadora. Esto sería útil para facilitar la desambiguación de una oración, por mencionar un ejemplo. En concreto, en este trabajo se realizará un etiquetador automático a nivel morfológico para textos en otomí. Sobre la tarea de etiquetado automático se profundizará en la sección 2.3.

2.2. Machine Learning (ML)

El Aprendizaje de Máquina, o en inglés *Machine Learning (ML)*, es otra de las subramas de la *AI*. Para entender el *ML* se pueden destacar tres conceptos: datos, modelos y aprendizaje.

El primer concepto, los **datos** (también conocidos como conjunto de da-

tos), es la parte fundamental de un sistema de *ML*. El objetivo de estos sistemas es extraer del conjunto de datos patrones valiosos o información útil que es relativamente difícil de vislumbrar o de extraer de forma manual. En ese sentido, parafraseando a Shalev-Shwartz y Ben-David (2014): una característica de este tipo de sistemas computacionales es que, en contraste con soluciones más tradicionales, dada la complejidad de los patrones que se necesitan detectar, un programador o programadora no podría proporcionar una especificación detallada y explicita de como extraer dichos patrones.

El segundo concepto, los **modelos**, son diseñados, construidos e implementados. El modelo tiene como objetivo la predicción de datos nunca antes vistos basándose en el conjunto de datos de donde se pretende extraer la información. En ese sentido, un modelo de aprendizaje podría describirse como un proceso de generación de datos, similares a los del conjunto de datos. Un buen modelo puede ser usado para predecir que pasaría en el mundo real sin la necesidad de hacer experimentos reales (Deisenroth et al., 2020).

Para que los modelos sean capaces de generar estos datos de forma acertada, deben tomar en cuenta aspectos del conjunto de datos al momento del modelado y extracción. Por ejemplo, no todos los datos con los que se trabaja son numéricos y esto puede ser problemático, ya que las computadoras solo pueden procesar números. Es por esto que es indispensable que los datos tengan una representación numérica. Por ejemplo, si el conjunto de datos consiste de un conjunto de palabras, es conveniente convertir esas palabras a vectores para su correcto procesamiento.

El último concepto, el **aprendizaje** (en inglés *learning* y que es parte del nombre *Machine Learning*), relaciona los datos y el modelo. Puede entenderse como la automatización que permite encontrar patrones en los **datos** por la optimización de parámetros de un **modelo**. A este proceso de automatización se le llama entrenamiento. Suponiendo que se tiene un conjunto de datos y un modelo adecuados, entrenar el modelo quiere decir que utilizaremos los datos disponibles para optimizar los parámetros del modelo con respecto a una función objetivo. Esta función evaluará que tan bien (o mal) el modelo predijo los datos del entrenamiento.

En *ML*, la función objetivo permite una notación matemática formal para cuantificar que tanto las máquinas están aprendiendo. En general, entre más bajo sea su valor es mejor, por lo que se busca minimizar la función. Es por ello que a esta función se le suele llamar también función de perdida (*loss*

function)[5]. Para obtener el valor deseado de la función objetivo se utilizan métodos de optimización numérica. La función objetivo será definida con respecto a los parámetros del modelo y depende del conjunto de datos (Zhang et al., 2020).

Para que el aprendizaje sea exitoso se necesita que la máquina logre generalizar y hacer predicciones correctas con datos que no ha visto. Si el modelo tiene un buen desempeño solo con los datos de entrenamiento, significa que la máquina memorizó los datos por lo que sus predicciones no son significativas. Para verificar que la máquina ha generalizado es común que el conjunto de datos sea dividido en dos partes. Una parte es utilizada en la fase de entrenamiento y se le llama conjunto de entrenamiento (*train set*). La segunda parte es utilizada para evaluar el desempeño del modelo y se le llama conjunto de pruebas (*test set*).

Formalmente, Mitchell et al. (1997) dice que *una máquina "aprende" de una experiencia E con respecto a una tarea T y una medida de rendimiento P, si el rendimiento de las tareas T, medidas por el rendimiento P, mejora con la experiencia E.* En otras palabras, un **modelo aprende** a partir de los **datos** si su rendimiento en una determinada tarea mejora cuando se toman en cuenta los datos.

En resumen, un sistema de *ML* se compone principalmente de tres elementos: datos, los cuales deben tener una representación adecuada para ser procesados por la computadora (generalmente como vectores); un modelo apropiado, dada la tarea a resolver, por ejemplo, un modelo que usa un enfoque probabilístico; por último, el aprendizaje a partir de los datos utilizando métodos de optimización numérica. El objetivo es que el modelo tenga un buen desempeño con los datos que no ha visto antes para lograr una generalización. En la figura 2.2 se muestra una vista general de los elementos que componen un sistema de aprendizaje.

Como se mencionó antes, las tareas que trata el *ML* tienen la peculiaridad de no tener una solución o forma de automatización bien definida, debido a su complejidad, a diferencia de tareas convencionales donde efectivamente hay un algoritmo bien definido para su solución. Para ilustrar esto proponemos un ejemplo.

Programar un filtro que distinga entre correo electrónico deseado (*ham*) y no deseado (*spam*) supone un reto, ya que no hay reglas claras para su

[5]Esta es una mera convención, bien se pueden utilizar funciones donde entre mayor sea su valor es mejor con lo que se buscaría maximizar la función

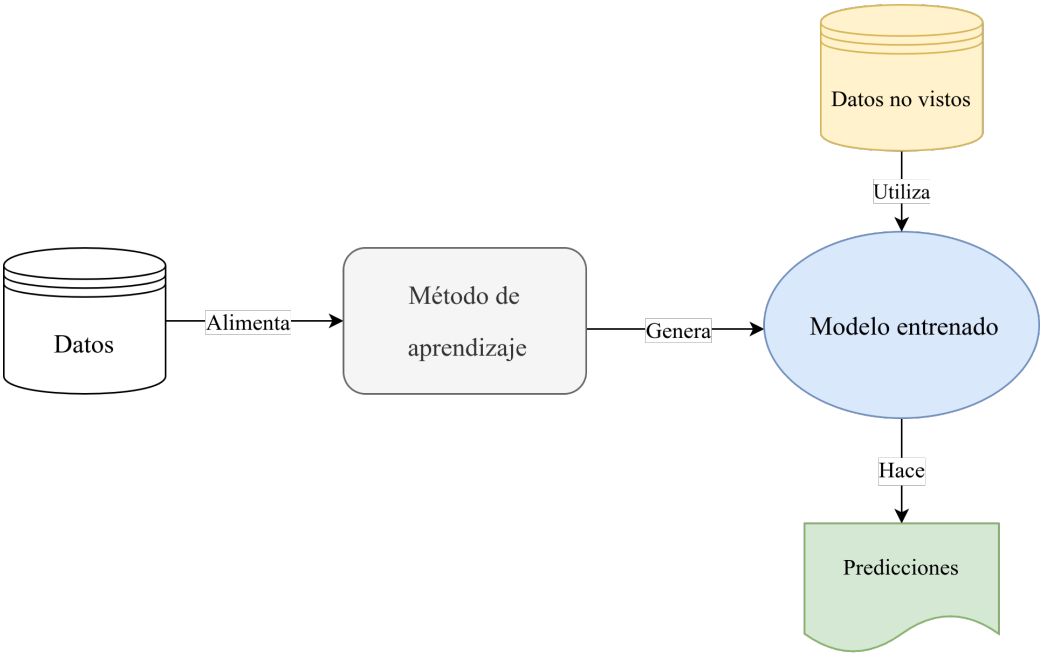

Figura 2.2: Sistema de aprendizaje

solución. Si el sistema solo se consideran ciertos elementos por separado como el remitente, la extensión del cuerpo del correo, o la aparición de ciertas palabras, podría fácilmente equivocarse y clasificar correos deseados como *spam* y viceversa. Lo que se necesita entonces es que la computadora infiera patrones, relaciones y elementos que no son claros a simple vista.

Para aplicar métodos de *ML* necesitamos tener un problema de aprendizaje bien definido y para ello debemos identificar ciertos elementos como la tarea a resolver, la medida de desempeño a mejorar y la fuente de experiencia. Retomando la definición de Mitchell et al. (1997) definiremos nuestro ejemplo en términos de una tarea *ML* a continuación; un detector de *spam*, que es la tarea, deberá mejorar su rendimiento, medido por el número de correos que clasificó correctamente, a través de la experiencia obtenida del análisis de los correos recibidos previamente.

En general, los métodos de *ML* y *NLP* se ven beneficiados de grandes cantidades de datos y del poder de cómputo creciente de los últimos años. Comúnmente, entre más grandes son los conjuntos de datos, mejores son los resultados. Es importante mencionar que, los modelos, tienen que entrenarse

con una muestra representativa que permita generalizar adecuadamente.

Sin embargo, hay escenarios retadores para los métodos tradicionales. En *NLP* uno de estos escenarios es el de los bajos recursos digitales, o en inglés *low resources*, donde la cantidad de datos de una lengua son extremadamente limitados e insuficientes para que los métodos funcionen correctamente. A diferencia de lenguas como el inglés o el español, que cuentan con una gran cantidad de textos (también conocidos como corpus) en medios físicos y digitales, las lenguas de bajos recursos digitales carecen de estos datos. Aunado a lo anterior, se muestra un escaso o nulo desarrollo de tecnologías para estas lenguas.

Un caso de lenguas con bajos recursos digitales son las lenguas indígenas de México, conformadas por 68 agrupaciones lingüísticas con 364 variantes [6]. Esta escasez de recursos puede deberse a factores como la baja cantidad de hablantes de las lenguas y a fenómenos políticos y sociales. En este trabajo se utilizará un corpus pequeño en lengua otomí, una lengua de bajos recursos digitales. En la sección 3.1 se describirán ciertas características de la lengua y del corpus utilizados para este trabajo.

2.2.1. Tipos de problemas en ML

Existe una gran diversidad de tareas que el *ML* puede resolver y la mayoría pueden categorizarse en tres enfoques: Aprendizaje supervisado (en inglés *Supervised Learning*), Aprendizaje no supervisado (en inglés *Unsupervised learning*) y Aprendizaje por refuerzo (en inglés *Reinforcement Learning*). Dependiendo que tarea se desea resolver, es conveniente elegir un enfoque u otro. Por ejemplo, el detector de correos *spam* que se mencionó con anterioridad es relativamente sencillo de modelar desde el *Supervised Learning*. A continuación una breve revisión de estos enfoques.

Supervised Learning

Los problemas que aborda el *supervised Learning* son aquellos en los que se predicen elementos llamados **etiquetas**. Para poder predecir estos elementos, el conjunto de datos debe componerse de una serie de pares de características y etiquetas. El conjunto de pares características-etiqueta reciben el nombre de **ejemplos** y se denotan como $\{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i=1}}^{\mathbf{N}}$. A las características $\mathbf{x_i}$ sobre N

[6] https://www.inali.gob.mx/clin-inali/

se les llama vectores de características y cada dimensión $j = 1, 2, ..., D$ de este vector contiene un valor que describe el ejemplo. Ese valor recibe el nombre de *feature* y se denota $x^{(j)}$. Las **etiquetas** son denotadas como y_i y pueden ser un conjunto finito de **clases** $\{1, 2, ..., E\}$, valores numéricos o elementos más complejos. El objetivo es producir un modelo que logre mapear vectores de características que nunca ha visto con su etiqueta correcta correspondiente.

Como se mencionó anteriormente, en el detector de *spam* se pretenden predecir dos clases $\{spam, no\ spam\}$ para un *email*. Los vectores de características de entrada podrían estar compuestos por $x^{(1)} = longitud\ del\ correo$, $x^{(2)} = asunto$, $x^{(3)} = palabras\ clave$ y tantas como se deseen tomar en cuenta.

Como puede verse, en la etapa de entrenamiento el supervisor, usualmente una persona, debe proporcionar el conjunto pares de *features* y su etiqueta correcta. Es por esto que se le llama *supervised learning*. En términos de probabilidad, esto puede verse como la estimación de una probabilidad condicional de una etiqueta dado un vector de características $P(y_i|\mathbf{x_i})$ (Zhang et al., 2020).

Algunos otros ejemplos de tareas comúnmente resueltas por este paradigma podrían ser:

- Detectar si alguna imagen contiene un gato o un perro
- Predecir si una transacción bancaria es o no fraudulenta
- Determinar el idioma en el que se encuentra una oración

Informalmente, el proceso de aprendizaje supervisado se constituye de los siguientes pasos: tomar un conjunto de ejemplos (conjunto de datos) de los cuales conocemos las *features* y sus etiquetas asociadas. Puede que estas etiquetas hayan sido recolectadas y asociadas previamente o que las *features* tengan que ser etiquetadas manualmente por personas. Seleccionar un subconjunto de estos pares de *features*-etiquetas de forma aleatoria para la etapa de entrenamiento (*train set*). Tomar el *train set* como la entrada de una función que devuelve como salida un modelo aprendido. A este modelo aprendido se le pasarán vectores de características que no haya visto con anterioridad y deberá predecir sus etiquetas correspondientes. La figura 2.3 muestra un esquema de un sistema de aprendizaje supervisado.

Existen casos más complejos que los descritos hasta ahora, por ejemplo, cuando los vectores de características y las etiquetas de salida pueden variar

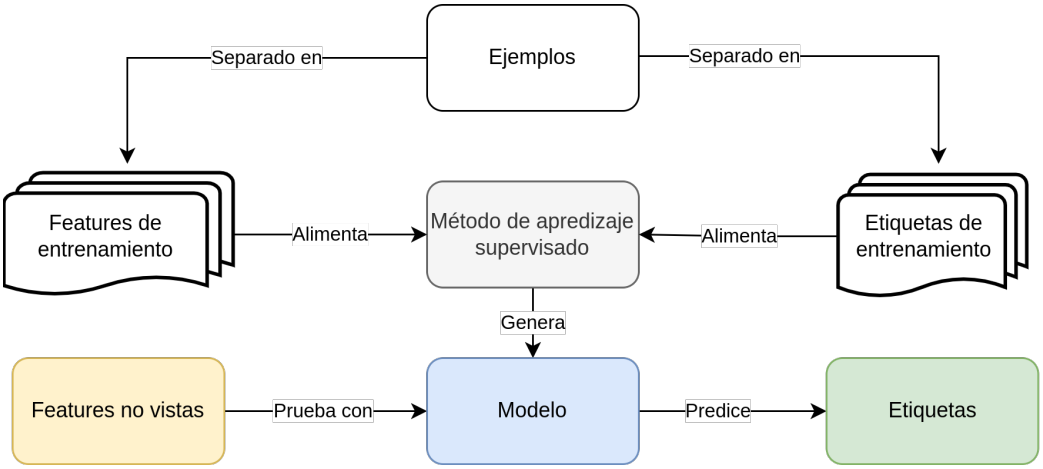

Figura 2.3: Sistema de aprendizaje supervisado

en longitud. Esto es el caso del procesamiento del lenguaje, ya que para hacer mejores predicciones es conveniente tener en cuenta elementos previos o siguientes. Por ejemplo, para comprender el significado de una palabra en una oración es útil conocer las palabras previas y subsecuentes.

Este tipo de problemas se encuentran dentro del llamado *sequence learning*. En estos se requiere procesar secuencias entradas del estilo $c^{(1)}, ..., c^{(n)}$ y emitir secuencias de salidas. Un caso particular son las tareas *sequence to sequence, Seq2Seq* en las que se tratan problemas con entradas y salidas de longitud variable (Zhang et al., 2020). Este trabajo aborda una tarea *Seq2Seq* que esencialmente inicia con el ingreso secuencias de texto en otomí y termina generando secuencias de etiquetas, o glosa, asociadas a las secuencias de entrada.

Unsupervised Learning

Existe otro enfoque llamado *unsupervised learning*. En contraste con el *supervised learning*, los datos de entrada no disponen de etiquetas asignadas previamente ni se tiene un valor esperado como salida. El conjunto de datos está constituido por **ejemplos no etiquetados** $\{\mathbf{x_i}\}_{\mathbf{i=1}}^{\mathbf{N}}$. Nuevamente, $\mathbf{x_i}$ será un vector de características.

En este enfoque predomina principalmente el reconocimiento de patrones por parte del modelo sin ninguna intervención previa. Estas técnicas son adecuadas para el agrupamiento (*clustering*) de datos donde los métodos

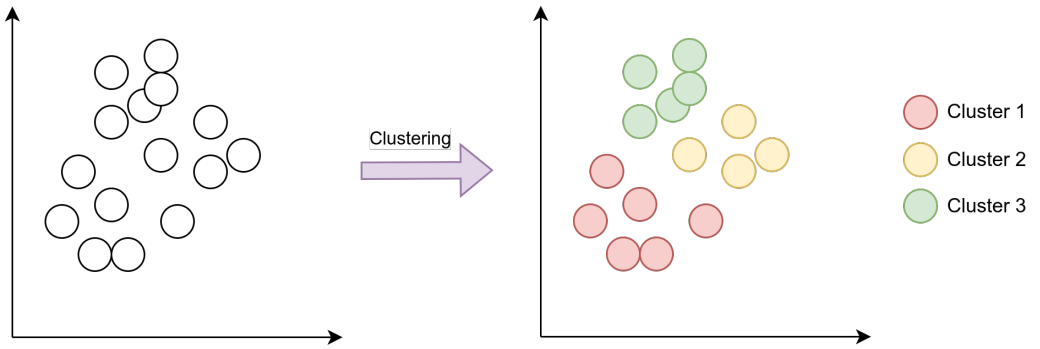

Figura 2.4: Datos antes y después de aplicar *clustering*

encuentran relaciones difíciles de ver o donde no se tiene una estructura u orden aparente. Por ejemplo, dado un conjunto de puntos en un plano, un método de aprendizaje no supervisado agruparía los puntos dependiendo de la distancia que hay entre ellos, asignando una etiqueta a cada conjunto de puntos. La figura 2.4 muestra como luce la técnica de *clustering*.

Otro caso es la reducción de dimensionalidad (*dimensionality reduction*) dónde la salida del modelo es un vector de características con menos características que el vector $\mathbf{x}$ de entrada y conservando la información mínima necesaria.

Pueden apreciarse los avances, relativamente recientes, del *unsupervised learning* en la aplicación de redes generativas adversariales (*Generative Adversarial Networks (GAN)*) para la generación de imágenes realistas de alta resolución (Goodfellow et al., 2020).

Reinforcement Learning

En el *reinforcement learning*, a diferencia de los dos enfoques previos donde solo se realizan predicciones, el sistema de aprendizaje interactúa con el ambiente a través de un agente que ejecuta acciones o toma decisiones. Posteriormente, las acciones tomadas por el agente impactarán en el ambiente.

Zhang et al. (2020) explica el *reinforcement learning* como un agente que interactúa con el ambiente en una serie de pasos sobre el tiempo. En cada paso de tiempo, el agente recibe una observación del ambiente, con lo que deberá elegir una acción que afectará al ambiente por medio de algún mecanismo (a veces llamado un actuador). Finalmente, el agente recibe una

recompensa del ambiente. El agente recibirá la siguiente observación, después elegirá la siguiente acción y así sucesivamente. El comportamiento de un agente en el *reinforcement learning* es gobernado por una política (*policy*). En general, una política es una función que mapea observaciones en el ambiente a acciones. El objetivo del *reinforcement learning* es producir una buena política.

Las aplicaciones principales se pueden encontrar en robótica o inteligencia artificial para videojuegos. Un ejemplo reciente es AlphaGo, que logró ganarle al campeón mundial de Go (Mind, 2021).

2.3. Etiquetadores

Cuando se trabaja con textos dentro del *NLP* surge el concepto de corpus, del latín "cuerpo", que no es más que una colección de textos. Al conjunto de varios corpus se les conoce como corpora. Un corpus cumple ciertas características en investigaciones modernas de *NLP*. Según (McEnery y Wilson, 2003) estas características son las que se listan a continuación:

- **Muestreo y representatividad**: es de interés tener variedad lingüística en lugar de un texto individual o textos de un solo autor. El corpus se analiza a partir de muestras de esa variedad
- **Tamaño finito**: un corpus tiende también a ser de un tamaño determinado, por ejemplo, 10,000 palabras
- **Legible por computadoras**: por mucho tiempo el término corpus se refería exclusivamente a textos impresos. Actualmente, esto ha cambiado, por lo que en general un corpus está en un formato que permite a las máquinas leerlos, procesarlos y manipularlos
- **Referencia estándar**: No es una parte esencial de la definición de corpus, pero a menudo hay un entendimiento tácito de que un corpus es una referencia estándar para la variante lingüística que representa. Esto puede verse, por ejemplo, cuando un corpus es ampliamente conocido y usado, como es el caso del corpus de Francis y Kucera (1979). Esto permite comparar fácilmente resultados de nuevos desarrollos y evitar que las diferencias entre investigaciones sean atribuidas al uso de datos exclusivamente y más a las presuposiciones o metodologías utilizadas.

Se puede encontrar cierto tipo de información en los textos. Pueden ir desde el nombre del archivo, qué autores tiene, lugares y fechas en que fueron escritos, etc. Estas serian cuestiones generales y externas al texto. En términos particulares están también las anotaciones lingüísticas. Estas anotaciones consisten en adjuntar códigos especiales a las palabras (o incluso partes de palabras como veremos más adelante) indicando características detalladas. Los mencionados códigos especiales son conocidas como etiquetas (en ingles *tags*).

Los corpus pueden encontrarse en dos formas: anotados, donde el texto original es enriquecido con información lingüística, y no anotados, donde el texto se encuentra tal cual fue obtenido. Si bien un corpus no anotado es valioso para el estudio del lenguaje, su valor incrementa considerablemente al ser anotado.

Un aspecto notable de un corpus anotado es que ya no es solo un texto con presencia de información implícita, sino que, dicha información se hace explícita con una anotación concreta (puede verse un ejemplo más adelante en la tabla 2.2). Esta información explícita permite obtener y analizar fenómenos acerca de un lenguaje más rápida y fácilmente. Esto es particularmente cierto cuando los textos que son procesados por máquinas.

2.3.1. *Part of Speech*

El tipo de anotación más básica es la *POS* (*Part Of Speech*). En este etiquetado se asigna un *tag* indicando la categoría morfosintáctica a las palabras (Por ejemplo, si la palabra es un adjetivo, sustantivo, participio pasado, etc.). Una muestra de etiquetas se encuentra en la Tabla[7] 2.1. Sin embargo, en sistemas de *NLP* se utilizan listas más completas como las del corpus, de Francis y Kucera (1979), donde se pueden encontrar hasta 87 categorías. Además de determinar la categoría morfológica, las etiquetas *POS*, brindan una gran cantidad de información lingüística sobre las palabras y sus vecinas.

El etiquetado *POS* es uno de los más comunes, en parte, porque puede delegarse a una computadora y porque se obtiene un alto grado de precisión en la predicción de etiquetas con una baja intervención humana. Para que las etiquetas sean altamente predecibles se toma en cuenta el contexto cercano de las palabras y se agrega información básica del lenguaje (McEnery y Wilson,

[7]Las etiquetas están en inglés. Por ejemplo, sustantivo en inglés es *Noun* por ello la etiqueta NN

Categoría	Etiqueta
Sustantivo	NN
Verbo	VB
Adjetivo	JJ
Adverbio	RB
Conjunción	CC
Determinante	DT

Tabla 2.1: Ejemplos de etiquetas *POS*

2003). Sin embargo, no debe pensarse que esta tarea es trivial. Para hacer que una computadora obtenga etiquetas *POS* hay una serie de consideraciones importantes que quedan fuera del alcance de este trabajo.

La información lingüística que aportan las etiquetas *POS* es una base fundamental para incrementar la especificidad de la extracción de datos de un corpus y es, también, parte esencial para otras formas de análisis. Por mencionar algunos ejemplos; dado un sustantivo, se puede inferir que palabras son más probables de ocurrir a continuación; ciertas etiquetas *POS* puede índica como es que se pronuncia una palabra; facilita la obtención de la raíz de palabras[8]; las etiquetas juegan un papel importante en el *morphological parsing* y algoritmos para la desambiguación de palabras.

2.3.2. Parsing

Otro etiquetado, que es opera a un nivel lingüístico distinto, es llamado análisis sintáctico o *parsing*. En el proceso de *parsing* se toma una entrada y se produce algún tipo de estructura lingüística que puede tener varias formas. Ya sean estas cadenas de texto, árboles o una red (Jurafsky y Martin, 2008).

Corpora que se ha sometido a un proceso de *parsing* es también llamada *treebanks* ya que la estructura de las sentencias son a menudo representadas como un árbol. Un ejemplo puede verse en la figura 2.5. La información gráfica presente en uno de estos diagramas puede ser representada también usando etiquetas. Por ejemplo, la equivalencia con etiquetas del árbol de la figura 2.5 se puede ver en figura 2.6.

[8]Esta tarea es conocida como *stemming* y es muy utilizada en la extracción de información

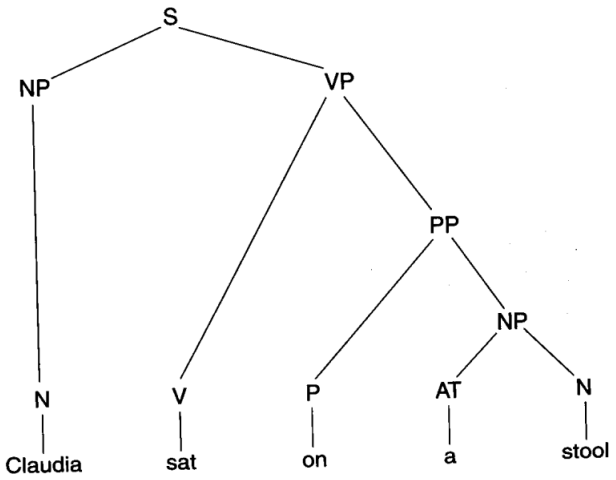

Figura 2.5: *Parsing* de la frase *Claudia sat on a stool* tomada de McEnery y Wilson (2003)

```
[S
  [NP Claudia_NP1 NP]
  [VP sat_VVD
    [PP on_II
      [NP a_AT1 stool_NN1 NP]
     PP]
   VP]
S]
```

Figura 2.6: Representación con etiquetas del árbol de la figura 2.5

2.3.3. Parseo morfológico

La **morfología** es el estudio de como unidades mínimas de significado se estructuran para formar palabras. Estas unidades suelen ser llamadas **morfemas**. De este modo, la palabra *cama* consta de un solo morfema, mientras la palabra *niñas* consiste de dos: *niña* y *-s*. Con esto en mente se pueden distinguir dos grandes tipos de morfemas: **raíces** (*stems*) y **afijos**. Las reglas morfológicas permiten reconocer las partes que conforman una palabra o sus morfemas.

Los afijos agregan significado adicional al *stem* y se dividen a su vez en prefijos (preceden al *stem*), sufijos (van después del *stem*), infijos (se insertan dentro del *stem*), entre otros. Por ejemplo, la palabra *Amorfo* se descompone en el prefijo *A-*, en el *stem morf* y en el posfijo *-o* (denotando género).

Con este preámbulo en mente, introduciremos el concepto de parseo morfológico (o en inglés *morphological parsing*). Jurafsky y Martin (2008) lo describe como una tarea donde se reconoce en qué morfemas se puede descomponer una palabra y cómo construir una representación estructurada de este hecho. Cabe destacar que algunas palabras pueden ser ambiguas entre diferentes *parsings*. Por ejemplo, las palabras *vino* o *canto* pueden tener varios posibles resultados como puede verse en la tabla 2.2.

Entrada	**Salida del *morphological Parsing***
vino	venir + V + PRF + 3 + SG
vino	vino + N + MASC + SG
canto	cantar + V + P.IND + 1 + SG
canto	canto + N + MASC + SG

Tabla 2.2: Resultados del *morphological parsing* de la palabra "vino" y "canto" (Jurafsky y Martin, 2008)

El *morphological parsing* es un paso importante para el procesamiento del lenguaje y el habla, ya que, encuentra los morfemas constitutivos en las palabras (Jurafsky y Martin, 2008). De en la tabla 2.2 se destacan dos cosas. En primer lugar, la segunda columna muestra el *stem* de la palabra de entrada, por lo que hubo un proceso de lematización[9]. En segundo lugar, las etiquetas hacen explícita cierta información del texto. Con esto el sistema

[9]La lematización consiste en reducir palabras hasta obtener la parte que da el mayor significado. Estas partes son también conocidas como *stems*.

puede lidiar con la ambigüedad de las palabras, lo cual, no es una tarea trivial.

2.3.4. Glosa

Un paso importante en la documentación lingüística es la descripción de la gramática de un lenguaje. El análisis morfológico constituye un paso para la construcción de esta descripción. Tradicionalmente, esto se hace mediante glosa interlineal.

La glosa es un tipo de etiquetado dónde lingüistas analizan oraciones de un lenguaje y segmentan cada palabra con el objetivo de anotar las categorías morfosintácticas de los morfemas de la palabra. Esto es describir la estructura morfológica de una oración y asociar a cada morfema una etiqueta morfológica que será la glosa. En el ejemplo (1) se muestra una oración en otomí glosada.

En lingüística usualmente hay tres niveles de análisis: 1) la segmentación por morfemas, 2) La glosa que describe esos morfemas y 3) la traducción o correspondencia léxica en el lenguaje de referencia. Este trabajo se enfoca a los primeros dos niveles de análisis.

Estas etiquetas son datos lingüísticos valiosos, ya que permiten la documentación de lenguas y posibilitan el desarrollo de tecnologías de *NLP*. Sin embargo, no todas las lenguas tienen disponible este tipo de corpora anotada. Comúnmente, el glosado se hace a mano y requiere del expertiz de lingüistas, lo que la hace una tarea costosa en tiempo y esfuerzo. En particular, las lenguas de bajos recursos digitales carecen de documentación y tecnologías del lenguaje (Mager et al., 2018).

Este trabajo busca producir un glosador automático para una lengua de bajos recursos digitales utilizando métodos de *ML* y *NLP*, buscando, por un lado, automatizar parte del glosado manual y, por otro lado, contribuir a las pocas herramientas de computacionales disponibles para estas lenguas.

(1) hí tó=tsogí
NEG 3.PRF=stem

Nuestro sistema sería capaz de segmentar morfológicamente cada palabra y asignar glosa a los morfemas, como se muestra en el ejemplo (1). La traducción implica un nivel diferente de análisis y, dada la escasez de recursos digitales, no se aborda en este trabajo.

Etiquetas BIO

Para el corpus de entrenamiento se utilizaron etiquetas BIO o *BIO-labels*. Las etiquetas BIO consiste en asociar cada etiqueta original con una etiqueta *Beginning-Inside-Outside (BIO)*. Esto quiere decir que cada posición de las letras de un morfema es declarada como inicio (B) o interna (I). Descartamos las etiquetas externas (O). Las etiquetas BIO incluyen las categorías de los morfemas, por ejemplo: B-STEM, o afijos de glosa, por ejemplo: B-PST para tiempo pasado. Una representación de la palabra *tótsogí* podría ser la siguiente:

(2)

t	ó	t	s	o	g	í
B-3.PRF	I-3.PRF	B-stem	I-stem	I-stem	I-stem	I-stem

Como puede verse en el ejemplo (2) las etiquetas BIO ayudan a delimitar las fronteras de morfemas dentro de una palabra y asignan una etiqueta de glosa a cada morfema. Esto fue parte del preprocesamiento del corpus y se abundará en la subsección 3.2.1.

El etiquetado de textos tradicionalmente se hace de forma manual, haciendo a esta una tarea costosa y repetitiva. Es por esto que es deseable encontrar procesos que la automaticen.

La mayoría de etiquetadores automáticos entran en una de dos categorías; los **basados en reglas**, donde se toma en cuenta una base de datos con reglas de desambiguación escritas a mano, y los **etiquetadores estocásticos**, que resuelven las ambigüedades usando un corpus de entrenamiento con el que computan la probabilidad de que cierta etiqueta sea asignada a cierta palabra dado cierto contexto (Jurafsky y Martin, 2008). Este trabajo se desarrolla en el marco de los etiquetadores estocásticos.

En resumen, el etiquetado automático determina y asigna etiquetas a las unidades lingüísticas. La entrada de un sistema de etiquetado puede ser una cadena de texto y una lista de etiquetas. La salida será la asociación cada unidad lingüística con la mejor etiqueta dado el contexto. El etiquetado varía dependiendo del tipo de etiquetas y unidades a la que se le asignan. Obtener etiquetas automáticamente es un proceso sumamente importante para una amplia variedad de aplicaciones de *NLP*.

2.4. Modelos gráficos probabilísticos convencionales basados en grafos

Hasta ahora hemos mencionado una amplia variedad de tareas que son abordadas con técnicas de *ML* y *NLP*. Diferentes enfoques pueden ser considerados a la hora de modelar y representar estas tareas dentro de una computadora. Modelar una tarea consiste en codificar el conocimiento en un formato legible por una máquina. A estos modelos se les pueden aplicar algoritmos que permiten obtener respuestas a preguntas basadas en el mismo modelo. Es común que modelos de *NLP* contengan un alto grado de incertidumbre y que sean difíciles de computar.

La teoría de la probabilidad provee un conjunto de herramientas que permite reducir la incertidumbre determinando los estados posibles y la probabilidad de que estos ocurran en el sistema. Estos estados pueden ser caracterizados en términos de un conjunto variables aleatorias χ. El valor de cada variable define una propiedad importante del sistema. Por ejemplo, en un sistema de etiquetado, una variable a considerar es la probabilidad de que a una palabra le pertenezca cierta categoría morfológica.

El objetivo es obtener, de forma probabilística, los valores de una o más variables dada la observación de otras variables previas. Para ello se construye una distribución de probabilidad conjunta $P(X, Y)$ sobre el espacio de todos los posibles valores que puede tomar un conjunto de variables aleatorias χ. Lamentablemente, considerar todas y cada una de las características de un sistema puede fácilmente llevar problemas intratables computacionalmente.

Es en este contexto que son relevantes los modelos probabilísticos basados en grafos[10]. Este es un *framework* que, según Koller y Friedman (2009), permite describir, construir y utilizar la estructura de distribuciones de probabilidad complejas de forma compacta. Estos modelos permiten una representación gráfica para codificar de forma compacta distribuciones complejas de alta dimensionalidad.

En estas representaciones basadas en grafos los vértices corresponden a variables de nuestro dominio y las aristas corresponden a las interacciones probabilísticas entre ellas. Por ejemplo, en la figura 2.7, las variables objetivo están representadas en color blanco. También, se asume que las variables X_i son siempre visibles, por lo que son denotadas en color gris.

Este tipo de modelos tiene una doble perspectiva. Por un lado, otorga

[10]También llamadas gráficas

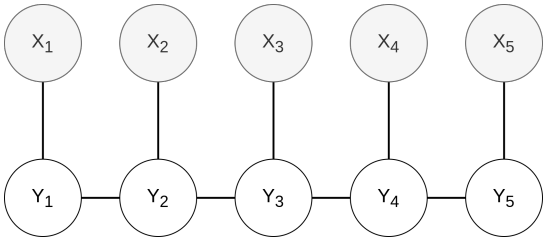

Figura 2.7: Estructura de un *linear-chain CRF* (Koller y Friedman, 2009).

información sobre la independencia que se encuentra en la distribución de variables. Por otro lado, la estructura del grafo define de forma compacta la representación de una distribución de alta dimensionalidad. Esto permite redefinir la distribución conjunta en el producto de factores más pequeños cada uno con un espacio de posibilidades reducido.

Hay dos grandes familias de grafos, los bayesianos, que usan grafos dirigidos, donde las aristas surgen de un vértice hacia otro con cierta dirección, y las redes markovianas, donde se usan grafos no dirigidos. Estas últimas son útiles en casos donde no es posible asignar naturalmente una direccionalidad que indique la interacción entre variables.

En resumen, los modelos basados en grafos en general permiten la representación de una distribución de probabilidad conjunta o condicional sobre χ.

2.4.1. Los límites de los modelos gráficos para bajos recursos

Como se mencionó, en *NLP* una tarea de gran interés es el etiquetado y segmentación de secuencias de textos. Para abordar esta tarea comúnmente se usan un par de modelos basados en grafos; los **modelos generativos**, cómo los *Hidden Markov Models (HMMs)*, y **modelos condicionales**, como los *Maximum Entropy Markov Models (MEMMs)* (Koller y Friedman, 2009).

Los modelos generativos intentan modelar una probabilidad conjunta $P(X, Y)$ sobre observaciones y etiquetas y, como mencionamos anteriormente, definir este tipo de distribución de probabilidad requiere enumerar todas las observaciones posibles. Las limitantes de este enfoque, además de la mencionada intratabilidad computacional, son las grandes dimensionalidades en el vector de entrada X y la dificultad de representar múltiples características

que interactúan unas con otras, por mencionar algunas.

En respuesta a las limitantes de los modelos generativos se utilizan modelos condicionales. Estos modelos no son tan estrictos como los primeros al momento de asumir independencias en las observaciones. Los modelos condicionales especifican la probabilidad de posibles etiquetas dada una secuencia de observación.

La probabilidad condicional puede depender de características arbitrarias y no dependientes de la secuencia de observación sin forzar al modelo a tomar en cuenta la distribución de estas características, permitiendo que el modelo sea tratable (Lafferty et al., 2001). Esto quiere decir que no se tratan de modelar todas las observaciones, dado que en al momento de realizar pruebas ciertas observaciones son fijas.

Un ejemplo son los *MEMMs* que, a diferencia de los *HMMs*, son modelos secuenciales de probabilidad condicional. Los modelos condicionales, y otros no generativos, presentan claras ventajas frente a los modelos generativos de estados finitos y que son clasificadores basados en el estado siguiente. Sin embargo, ambos tipos de modelos comparten una debilidad llamada *label bias problem*.

Lafferty et al. (2001) define que existe el *label bias problem* cuando "las transiciones que dejan un estado compiten solo entre sí, en lugar de entre todas las demás transiciones en el modelo". Esto puede entenderse como: dado que las transiciones son las probabilidades condicionales de los siguientes posibles estados, una observación puede afectar cuál será el estado siguiente sin tomar en cuenta que tan adecuado será este. Por tanto, se tendrá un sesgo en los estados con menos transiciones de salida o que tengan menos frecuencia.

Como menciona Sutton et al. (2012), modelar las dependencias entre las entradas puede conducir a modelos intratables, pero ignorar estas dependencias puede reducir el rendimiento. Para el etiquetado de secuencias en entornos de bajos recursos digitales se requiere una solución más conveniente. A continuación presentaremos los *Conditional Random Fields* como alternativa.

2.4.2. Conditional Random Fields

Los *Conditional Random Fields (CRFs)* son modelos basados en grafos que aprovechan técnicas de aprendizaje estructurado. Este tipo de modelos permite capturar una distribución de probabilidad condicional $P(Y|X)$, dón-

de Y es un conjunto de variables objetivo y X son un conjunto (disjunto) de variables observadas. Los *CRFs* poseen las ventajas de modelos como los *MEMMs* y, en principio, solucionan el *label bias problem*.

Koller y Friedman (2009) menciona que los *CRFs* son grafos no dirigidos (aunque hay versiones híbridas), que pueden ser parametrizados en el mismo sentido que una red markoviana ordinaria, con un conjunto de *feature functions* representadas como $\phi(\boldsymbol{D}_1)_1, ..., \phi(\boldsymbol{D}_m)_m$ dónde $\boldsymbol{D}$ es un subgrafo llamado *subclique*[11]. Estas *feature functions* pueden ser codificadas de forma compacta parametrizándolas como un modelo *log-linear*[12].

Formalmente, Koller y Friedman (2009) define los *CRFs* como se muestra a continuación:

Definición 1. Un *Conditional Random Field* es un grafo no dirigido $\mathcal{H}$ cuyos vértices corresponden con $Y \cup X$; la red es anotada con un conjunto de *feature functions* $\phi_1(\boldsymbol{D}_1), ..., \phi_m(\boldsymbol{D}_m)$ tal que $\boldsymbol{D}_i \nsubseteq X$. La red codifica una distribución condicional como se muestra a continuación:

$$P(Y|X) = \frac{1}{Z} \prod_{i=1}^{N} exp\{w^T \phi(Y, X, i)\} \tag{2.1}$$

Dos variables en $\mathcal{H}$ son conectadas por una arista (no dirigida) siempre que aparezcan juntas en el alcance de alguna *feature function*.

En la definición 1, $Z = \sum_y \prod exp\{w^T \phi(Y, X, y)\}$ es la función de partición y w es el vector de pesos. Por otra parte, $X = (c_1, ..., c_N)$ es una secuencia de caracteres que representa la entrada de nuestro modelo (una oración), y $Y = (g_1, ..., g_N)$ representa la salida (una secuencia de *BIO-labels*). $\phi(Y, X, i)$ es el vector que representa el i-esimo elemento en la sentencia de entrada. Este vector es extraído por la *feature function* ϕ.

Los *CRFs* necesitan representar cada carácter de la secuencia de entrada como un vector. Esto se realiza por medio de las *feature functions*. Para mapear de la secuencia de entrada a vectores, las *feature functions* necesitan tomar en cuenta información relevante acerca de las secuencias de entrada y salida. A esta información la llamaremos *feature lists*. Las *feature functions* desempeñan un papel importante en el desempeño de los modelos de *CRFs*. En la subsección 3.2.2 detallamos las *features* utilizadas en este trabajo.

[11] Un *clique* es un subconjunto de vértices de un grafo no dirigido tal que dos vértices distintos en el *clique* son adyacentes al grafo

[12] En este tipo de modelos se convierten los factores al espacio logarítmico

El algoritmo de optimización *L-BFGS* busca maximizar el logaritmo de la probabilidad de los datos de entrenamiento con los términos de regularización L_1 y L_2 antes mencionados. En la práctica, este algoritmo mejora los pesos de las *features* lentamente al inicio del proceso de entrenamiento, pero converge rápidamente a los pesos óptimos de las *features* al final (Okazaki, 2007).

2.5. Avances en etiquetadores automáticos

Varios trabajos han abordado la tarea del glosado automático utilizando diversos métodos de *NLP*. A continuación daremos una revisión breve.

En el trabajo de Snoek et al. (2014) se utiliza un enfoque basado en reglas (usando un *Finite State Transducer*) obteniendo glosa para Plains Cree, una lengua algonquina. Ellos se enfocaron en el análisis de sustantivos. En Samardzic et al. (2015) proponen un método para glosar la lengua Chintang; ellos dividen la tarea en glosado gramatical y glosado léxico. El glosado gramatical es abordado como una tarea supervisada de etiquetado *POS*, mientras que para el glosado léxico usaron un diccionario. Una automatización completa no es realizada dado que la segmentación de palabras no es abordada en este trabajo.

Algunos otros trabajos se han acercado a la automatización de un *pipeline* para glosado automático como una tarea de etiquetado supervisado usando modelos secuenciales de *ML* y que se han enfocado, en particular, en lenguas de bajos recursos digitales (Moeller y Hulden, 2018; Anastasopoulos et al., 2018; Zhao et al., 2020). En Anastasopoulos et al. (2018), utilizan modelos neuronales con recursos duales, aprovechando traducciones fáciles de obtener.

En Moeller y Hulden (2018), realizan glosado automático para la lengua Lezgi (de la familia Naj-Daguestaní) bajo difíciles condiciones de bajos recursos digitales. Implementaron diferentes modelos, por ejemplo; *CRF*, *CRF+SVM* y redes neuronales *Seq2Seq*. Los mejores resultados se obtuvieron de los *CRFs* aprovechando las etiquetas *POS*. El glosado se enfocó principalmente en etiquetado gramatical (funcional) de morfemas, mientras que los elementos léxicos fuero etiquetados simplemente como *stems*.

El trabajo de Moeller y Hulden (2018) influye fuertemente esta tesis. De hecho, Moeller y Hulden (2018) resalta la importancia de probar estos modelos en otras lenguas, en partículas lenguas ricas en morfología y polisintéticas con fusión[13]. Nuestro caso de estudio, el otomí, es precisamente una lengua

[13]Fue desarrollado un *paper* en el marco de la conferencia *AmericasNLP* que desarrolla

con patrones morfofonológicos complejos. Abundaremos más sobre el corpus en la sección 3.1.

este punto y amplía este trabajo de tesis. Puede consultarse en `https://aclanthology.org/2021.americasnlp-1.5.pdf`

Talk talk talk, it's only talk
Comments, cliches, commentary, controversy
Chatter, chit-chat, chit-chat, chit-chat
Conversation, contradiction, criticism
It's only talk
Cheap talk

Elephant Talk - Discipline, King Crimson

3

Etiquetador morfológico para el otomí

El objetivo de este capítulo es la presentación de la metodología utilizada en el trabajo que reporta en esta tesis. La metodología consta de un corpus etiquetado, del cual se incluye una descripción detallada, y la arquitectura para la generación de glosa para el idioma otomí. La arquitectura incluye, entre otras cosas, la codificación y preprocesamiento del corpus, el diseño e implementación de los *CRFs* y la determinación de las *feature lists* que contienen información de la lengua.

El punto principal de la metodología fue la implementación de los *CRFs* que mostraron claras ventajas sobre otros métodos de aprendizaje basados en gráficas. Elegimos este *framework* ya que se presenta como una opción para el contexto de los bajos recursos digitales que presenta el otomí, como se describirá más adelante. En ese sentido, la implementación de los *CRFs* fue utilizada para predecir secuencias de etiquetas, con el enfoque del aprendizaje estructurado, que describen las unidades morfológicas dentro de una palabra de una variante del otomí.

3.1. Corpus: otomí de Toluca

La clasificación lingüística introduce al otomí dentro de las lenguas otomianas, las cuales a su vez pertenecen a la rama otopame de la familia otomangue (Barrientos López, 2004). Cada variante muestra particularidades fonológicas, morfológicas, sintácticas y léxicas. En el tratamiento de textos por medio de técnicas de *NLP* se requiere que estos textos estén normalizados y homogéneos. Dicha normalización propicia la obtención del mejor desempeño posible en los diversos métodos de aprendizaje de máquina. Más adelante se describirá el proceso de preprocesamiento aplicado al corpus que tuvo el propósito de adecuar el corpus a la arquitectura y normalizar el texto.

Esta tesis utilizó un corpus en otomí que, además, cumple la característica de estar glosado. Se trabajó con la variante del otomí de Toluca de la región de San Andrés Cuexcontitlan.

Se recogió un corpus basado en el trabajo de Lastra (1992) titulado *El otomí de Toluca* y que a su vez fue etiquetado y glosado manualmente por el lingüista Víctor Germán Mijangos de la Cruz[1]. Este corpus es un subconjunto del corpus paralelo español-otomí que se encuentra en la plataforma web Tsunkua[2].

Además, se agregaron 81 líneas de casos poco usuales que son fenómenos poco frecuentes y, por tanto, particularmente difíciles de predecir. El corpus utilizado en la sección experimental está descrito en las tablas 3.1, donde se encuentra el tamaño de las etiquetas *POS* y el tamaño de la glosa, y 3.2, donde se pueden ver los tipos de textos presentes en el corpus.

Los textos que componen el corpus fueron construidos a partir de las aportaciones de diez hablantes distintos de entre diez y setenta y tres años, de los cuales, siete son de sexo femenino y tres masculinos (Lastra, 1992).

Los tipos de etiquetas *POS* presentes en el corpus se pueden observar en la tabla 3.3. Dentro del corpus se encontró glosas que no corresponden a etiquetas *POS*, sino que describen la semántica (traducciones) de las palabras. Como por ejemplo: en el fragmento en otomí `"...ná ra sapahtá pe lo prinsipal..."` ("*...empezado un Zapata, pero lo principal...*") `sapahtá` tendría como glosa la palabra `zapata` que es su traducción. Esta forma de presentar las etiquetas *POS* es común en el uso lingüístico y no vale la pena presentarla en la tabla 3.3, pues, son descriptivos por sí mismos.

[1] El corpus glosado puede encontrarse en: `https://github.com/VMijangos/Glosado_neuronal`

[2] `https://tsunkua.elotl.mx/`

Categoría	**Cuenta**
Tokens (POS)	8550
Tipos (POS)	44
Tokens (Glosa)	14405
Tipos (Glosa)	112
Total de oraciones etiquetadas	**1769**

Tabla 3.1: Tamaño del corpus

Textos	**Número**
Narrativos	32
Dialogados	4
Total de textos	**36**

Tabla 3.2: Textos del corpus

Por otra parte, una descripción de las etiquetas *POS* se puede encontrar en el apéndice A, en la tabla A.1, dónde se detalla el significado de cada etiqueta. Por último, la distribución de las etiquetas *POS*, se encuentra en la figura 3.1. En esta figura se muestra una carga importante hacia unas pocas etiquetas *POS*. Esta característica es considerará por el modelo de aprendizaje estructurado que utilizamos y puede suponer un problema importante en métodos de aprendizaje convencionales.

Para las etiquetas de Glosa nos basamos en las reglas de Comrie et al. (2008) desarrolladas por el departamento de lingüística del Instituto Max Planck y el Departamento de lingüística de la Universidad de Leipzig. El estándar consiste en diez reglas para la sintaxis y la semántica de glosas interlineales y un apéndice con un lexicón propuesto de etiquetas de categorías abreviadas (Comrie et al., 2008). Si bien las reglas cubren parte de las necesi-

v	obl	det	cnj	dem
unkwn	n	neg	p.loc	prt
conj.adv	dim	gen	cond	it
lim	aff	loc	dec	conj
cord	san	cnj.adv	regular/v	adv
adj				

Tabla 3.3: Tipos de etiquetas *POS*

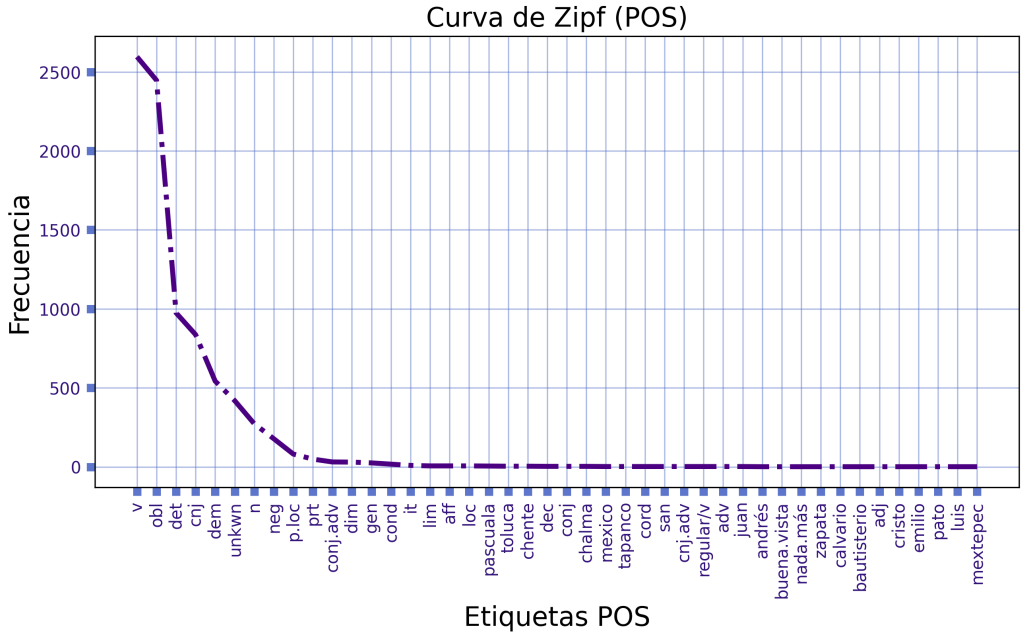

Figura 3.1: Distribución de etiquetas POS

dades lingüísticas en el glosado de textos, también, son flexibles y se pueden agregar o modificar las convenciones dependiendo de las necesidades.

Los tipos de etiquetas de glosa presentes en el corpus se encuentran en la tabla 3.4. En algunos casos aparecen números al inicio de etiquetas que significan las personas gramaticales. Existen combinaciones de varias etiquetas que son separadas por puntos. Por ejemplo, PL.EXC es una combinación de las etiquetas "plural" y "exclusivo".

Nuevamente, el significado para cada etiqueta de glosa se muestra en el apéndice B, en la tabla B.1. Además, la distribución de las etiquetas presentes en el corpus se muestran en la figura 3.2. Igual que en la figura 3.1 se observa que hay una carga pronunciada en la distribución hacia unas pocas etiquetas como STEM. Este fenómeno es propiciado, en parte, porque son escasos los textos digitales disponibles para el idioma otomí.

En las tablas 3.5 se muestran las cuentas de los diez tokens más comunes para las etiquetas *POS* y para la glosa presentes en el corpus. Es destacable tanto la cantidad de oraciones etiquetadas, presentadas en la tabla 3.1, como la cantidad de tokens de la tabla 3.5 ya que representan un conjunto de

1.cnt	1.cpl	1.cpl.irr	1.enf	1.icp
1.icp.irr	1.irr	1.obj	1.pls	1.pot
1.prf	1.pss	1.sg	2	2.cnt
2.cpl	2.enf	2.icp	2.icp.irr	2.obj
2.pot	2.prf	2.pss	3.cnt	3.cpl
3.cpl.irr	3.icp	3.icp.irr	3.imp	3.irr
3.obj	3.pls	3.pot	3.prf	3.pss
3.pss.pl	3.sg	adj	agujerear/v	animal.de.dios
aqui	aum	caus	chente	chico
com	como	comp	con	cond
conj.adv	coraje	ctrf	cuando	dcl
dem	det	det.dem	det.pl	dim
dios	dist	dual	dual.exc	dónde
eh	encl	gen	hasta	ila
int	it	lig	lim	loc
loco	lugar/v	maria	med	mexico
mientras	mod	mov	mucho	mujer/v
muy	neg	nom	p.loc	para
pascuala	pl	pl.exc	por.que	prag
prf	prt	psd	pueblo	pues
que	rapido	si	solo	spt
stem	tal.vez	tiempo	toluca	uno
vez	y			

Tabla 3.4: Glosas usadas en el modelo

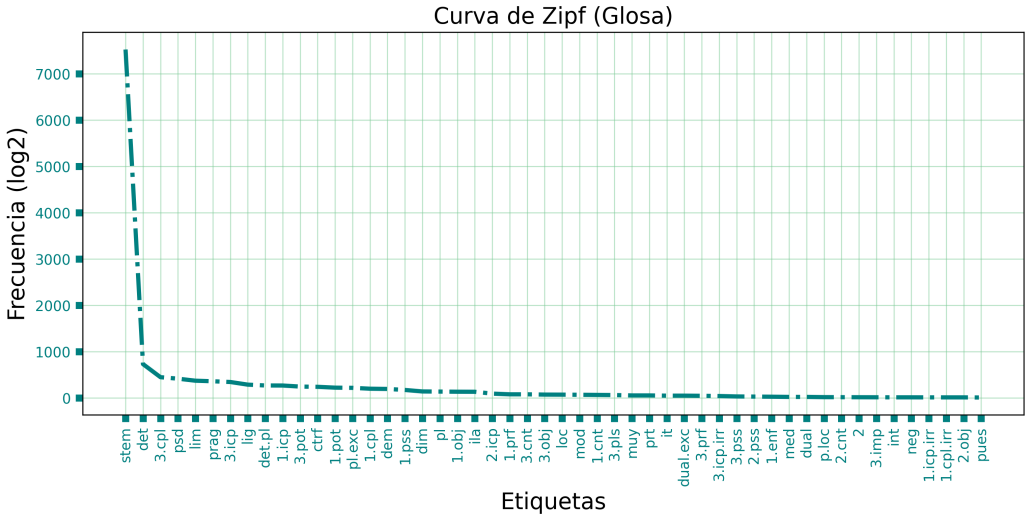

Figura 3.2: Distribución de glosa (primeras 50)

textos ínfimo en comparación con los corpus utilizados en métodos de *NLP* convencionales, donde, en general, su desempeño es altamente dependiente de grandes cantidades de textos.

Esta escasez en el corpus es consecuencia de que el idioma otomí es una lengua con bajos recursos digitales. Como menciona Vasques (2018), estas lenguas son aquellas que tienen una cantidad limitada de recursos digitales a consecuencia de una baja densidad de hablantes, cuestiones relacionadas con la brecha digital o motivos de otra índole. Gran parte de los métodos tradicionales de *NLP* tienen dificultades importantes para trabajar en entornos de bajos recursos digitales.

Sin embargo, existen métodos que buscan solucionar el problema de los bajos recursos digitales. Los trabajos de Moeller y Hulden (2018) y Anastasopoulos et al. (2018), y en particular aquellos trabajos que utilizan *CRFs*, tienen buen desempeño en estas condiciones experimentales.

Esto puede deberse a que los *CRFs*, a diferencia de otros modelos gráficos, toman las virtudes de los modelos generativos y de los modelos condicionales, evitando, a su vez, el *label bias problem* del que habla Lafferty et al. (2001).

Etiqueta *POS*	**Tokens**
v	2579
obl	2443
det	973
cnj	835
dem	543
unkwn	419
n	272
neg	176
p.loc	81
prt	49

Glosa	**Tokens**
stem	7501
det	733
3.cpl	444
psd	413
lim	370
prag	357
3.icp	341
lig	287
1.icp	270
det.pl	269

Tabla 3.5: Tokens de etiquetas (Primeras 10)

3.2. Arquitectura

Para esta tesis proponemos una arquitectura de aprendizaje estructurado y supervisado utilizando el método gráfico *Conditional Random Fields (CRF)*. La arquitectura permitirá la predicción de secuencias que describen las unidades morfológicas (glosa) dentro de una palabra en otomí y, ya que el resultado esperado es la generación de etiquetas *BIO* que, en principio, son secuenciales, un método que construye modelos de aprendizaje basados en grafos nos pareció adecuado.

Una vez obtenido el corpus, previamente glosado, realizamos una etapa de preprocesamiento del corpus, luego de esta etapa fueron creadas las *feature lists*, que a su vez, son utilizadas internamente por los *CRFs* para construir las *feature functions*. Las *feature functions* las definimos como X. Tales funciones contienen información sobre la estructura de la lengua y permiten que la máquina aprenda un modelo para etiquetar secuencias. Entonces X, que contiene las *feature functions*, serán la entrada de los *CRFs*.

Debido al enfoque de aprendizaje de máquina que estamos utilizando, consideramos dos conjuntos; un conjunto de entrenamiento y otro de pruebas. Los conjuntos de entrenamiento y pruebas fueron obtenidos del corpus que fue previamente glosados a mano por un experto lingüista.

La etapa de pruebas consistió en aplicar el modelo de aprendizaje al conjunto de evaluación. En ese sentido, aplicar el modelo se refiere a que con el modelo se generaron etiquetas *BIO* para textos del conjunto de pruebas. Con las etiquetas generadas y en comparación con las etiquetas reales se ob-

tuvieron, entre otras medidas de rendimiento, el *accuracy* del modelo. Las secuencias de etiquetas o salidas generadas por el modelo, dadas las observaciones de X, fueron definidas como Y.

En general, el glosado se realiza manualmente y requiere del expertiz de un lingüista y una cercana cooperación con hablantes nativos, quienes reciben capacitación elemental en lingüística y software (Moeller y Hulden, 2018). Estas condiciones convierten al glosado manual en una tarea altamente costosa en tiempo y recursos. El objetivo de esta arquitectura es realizar un correcto etiquetado automático de glosa para la lengua otomí, en particular la variante del otomí de Toluca, utilizando técnicas de aprendizaje estructurado y supervisado. Con ello se pretende asistir a personas especializadas, como lingüistas, en el glosado manual.

Parte de la definición de la estructura de la lengua se encuentra codificada en el conjunto de *feature lists*. Estas listas describen características de la lengua, considerando, detalladamente, el contexto, la posición y etiquetas gramaticales que brindan información útil para la fase de entrenamiento.

Puntualmente, el proceso de entrenamiento consistió en aprender los pesos contenidos en w. Siguiendo a Moeller y Hulden (2018), utilizamos el software de Okazaki (2007) llamado *CRFSuite* para implementar un modelo de tipo *Linear-Chain Conditional Random Field*. Esta implementación utiliza el algoritmo de optimización *Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)* para aprender los parámetros de los *CRFs*. Para evitar *overfitting* se incorporó la regularización *Elastic Net* en el proceso de optimización. *Elastic Net* consiste en la combinación de dos métodos de regularización; el de *Ridge*, que utiliza la norma L_2, y el de *Lasso* que utiliza la norma L_1(Giba, 2021) donde $L_1 = \sum_i |\theta_i|$ y $L2 = \sim_i \theta_i^2$.

Además de los coeficientes de regularización L_1 y L_2, fueron tomados en cuenta un conjunto de hiperparámetros que se mencionarán a continuación: El número máximo de iteraciones, el número máximo de memorias utilizadas para aproximar la matriz hessiana inversa, epsilon que determina la condición de convergencia, el número de iteraciones para probar el umbral de detención, delta que define el umbral de detención de cada iteración, el método de búsqueda de línea usado por el algoritmo *L-BFGS* y el número máximo de intentos para el algoritmo de línea. El modelo fue tratado a detalle en la subsección 2.4.2.

La arquitectura propuesta abarca desde la obtención del corpus etiquetado, pasando por el preprocesamiento del mismo, llegando a la creación de las *feature functions*, seguido de la separación de los datos en conjun-

```
[                    POS
                      |
  [                   V
    ['hi', 'stem'], 'neg'               POS
  ], # Palabra: hi                       |
  [                                      V
    ['tó', '1.prf'], ['tsogí', 'stem'], 'v',
  ], # Palabra: tótsogí
]
```

Figura 3.3: Ejemplo de texto etiquetado presente en el corpus

tos de entrenamiento y pruebas, continuando con la fase de entrenamiento y construcción del modelo de aprendizaje, terminando en la etapa de pruebas compuesta por la generación de etiquetas y posterior comparación con la glosa real. Las subsecciones que componen esta arquitectura son **codificación y preprocesamiento**, **implementación de los CRFs** y **feature functions**. En el siguiente capítulo se abordarán las etapas de generación de glosa y pruebas.

La arquitectura de aprendizaje supervisado para la generación de glosa para el otomí de Toluca se describe a continuación.

3.2.1. Codificación y preprocesamiento

Se obtuvo el corpus de un archivo de texto plano. Cada renglón de este archivo es una oración en otomí con glosa. Además, se tiene una etiqueta *POS* por cada palabra. Las frases están estructuradas en forma de listas que contienen otras listas válidas para el lenguaje `Python`.

La glosa está presente por cada fragmento de las posibles palabras en otomí. Ejemplificando, la frase *hi tótsogí* ("No lo he dejado") se representa en el corpus como se muestra en la figura 3.3.

Por último, la lengua otomí tiene una vasta variedad interna, lo cual implica diferentes ortografías dependiendo de la región. Tomar en cuenta esta variación ortográfica es de suma importancia para el preprocesamiento. Como menciona Elotl (2019) todas las lenguas otomíes son tonales y distinguen entre vocales orales y vocales nasales, existen fonemas que pueden presentarse en unas variantes, mientras que en otras están ausentes. En general, las vocales orales incluyen a las mismas vocales que también se presentan en el español: a, e, i, o, u; pero su inventario de vocales orales no se limita a estas

cinco.

IPA	ɨ	ɛ	ɔ	ʌ	ɵ
Ortografía práctica	u̲	e̲	a̲	i̲	o̲
Convención para este trabajo	μ	ϵ	α	ι	

Tabla 3.6: Representación de cada vocal en IPA (alfabeto fonético internacional)

Ya que el otomí presenta una amplia variedad de vocales, como se puede ver en la tabla 3.6, su representación digital puede suponer un reto por la falta de normalización o por la codificación. En nuestro caso, cuando se codificaban algunas vocales para ser representadas como cadenas, eran separadas por el lenguaje de programación, ocasionando un etiquetando que por si solo no tiene sentido. Entonces, fue necesario modificar la representación de algunas de las vocales del otomí. Las equivalencias de tales modificaciones pueden observarse también en la tabla 3.6

Entonces, las listas, por renglón, tiene la estructura `[[[letras, glosa], [letras, glosa], ..., POS],...]`. Una vez obtenido el corpus de entrenamiento se le aplicó cierto preprocesamiento. Este preprocesamiento consistió en asociar, a cada letra de cada palabra, dos elementos; la etiqueta *POS* y una *BIO-label* correspondiente a esa letra.

Retomando la figura 3.3, y después de aplicar el preprocesamiento, el resultado sería el que se muestra en la figura 3.4. Cabe señalar que las *BIO-labels* asignadas dependieron de la posición de la letra, como se explicó en la sección 2.

Con las palabras etiquetadas a nivel de letra se obtuvo un conjunto de entrenamiento y un conjunto de pruebas. Por una parte, el conjunto de entrenamiento permitió la observación de ejemplos y la posterior generación de un modelo de aprendizaje. Por otro lado, con el conjunto de pruebas obtuvimos el *accuracy* de los modelos que etiquetaron frases no vistas. Detallaremos esta y otras medidas de desempeño en la sección 4.1. Es importante destacar que estos dos conjuntos estuvieron completamente separados, ya que mezclar el conjunto de entrenamiento y de pruebas podría conducir a resultados erróneos y sesgados.

Previo al entrenamiento se construyeron las *feature lists* usadas a su vez por las *feature functions*. En ese sentido, por cada letra etiquetada de las palabras en otomí se tuvo una *feature list* y por cada *feature list* se tiene

```
[
  [
    ['h', 'neg', 'B-stem'],
    ['i', 'neg', 'I-stem']],
  [
    ['t', 'v', 'B-1.prf'],
    ['ó', 'v', 'I-1.prf'],
    ['t', 'v', 'B-stem'],
    ['s', 'v', 'I-stem'],
    ['o', 'v', 'I-stem'],
    ['g', 'v', 'I-stem'],
    ['í', 'v', 'I-stem']],
]
```

Figura 3.4: Sentencia preprocesada con *BIO-labels* por cada letra.

una *BIO-label* asociada. La construcción de estas funciones será descrita a profundidad en la subsección 3.2.2.

Los *CRFs* recibieron como entrada vectores, construidos con base en las *feature lists*, representados como X y sus respectivas *BIO-labels*, representadas por un vector Y. Estos vectores fueron asociados con cada *feature function* con concordancia uno a uno y respetando la posición.

3.2.2. Feature functions

La extracción de estas características es importante porque capturar fenómenos lingüísticos necesarios para que la estructura de la lengua se pueda plasmar en el modelo de aprendizaje. Estas características están capturando, entres otras cosas, el contexto de la palabra que es importante para predecir la morfología. Para construir las *feature lists* se necesita el corpus glosado y etiquetado a nivel de letra. Se extrajeron las siguientes características para cada letra en el corpus:

`Bias:` Esta característica captura la proporción de una etiqueta dada en el conjunto de entrenamiento. El modelo aprende los pesos asociados con las etiquetas como si las etiquetas se generaran independientemente de una distribución de probabilidad dada. Ayuda a tomar en cuenta que algunas etiquetas son poco usuales y otras muy usuales.

`letterLowercase:` Toma la letra y la convierte a minúsculas. Es importante para la creación del modelo tener en cuenta la letra que se está viendo

en un momento determinado para las predicciones posteriores.

postag: Toma la etiqueta *POS* de la palabra actual. Recordemos que es conveniente y muy útil tomar en cuenta las etiquetas *POS* ya que brindan información gramatical de la palabra que se observa en ese momento. Tal información se basa en la morfología y algunas veces en la sintaxis de la lengua.

prevpostag y nxtpostag: `prevpostag` toma la etiqueta *POS* de la palabra previa y `nxtpostag` toma la etiqueta *POS* de la palabra siguiente. En ambos casos si se encuentra la palabra.

BOS, EOS, BOW y EOW: Indicar el inicio y fin de frase y palabras. Otorga la capacidad de ver qué tipo de palabras se está considerando en ese momento. Por ejemplo, transiciones que propician estados iniciales y finales brindan información contextual.

`BOS` indica el inicio de la frase, `EOS` indica el fin de la frase, `BOW` indica el inicio de la palabra y `EOW` indica el final de la palabra.

letterposition: Indica la posición de la letra en la palabra. Otorga más información sobre el contexto de la palabra en la frase que se observa en un determinado momento.

prev<n>letters y nxt<n>letters: La recuperación de ventanas de contexto son convenientes para del otomí, ya que se trata de una lengua aglutinante donde, en general, las palabras son largas. En particular, la longitud promedio de las palabras en el corpus es de 4.89[3]. Esta característica da la pauta para que la observación del contexto en un determinado momento sea relevante para la construcción del modelo.

`prev<n>letters` toma las n letras previas y `nxt<n>letters` toma las n letras siguientes, si existen en ambos casos. La variable n determina el nombre de la *feature* que indica el tamaño de la ventana a partir de la letra que se está viendo en ese momento y va desde el número uno, caso donde no se incluye número en el nombre, hasta el número cuatro. Por ejemplo, `prev4letters` toma las cuatro letras previas y `nxtletter` toma la letra siguiente.

[3]Promedio de la distribución de palabras basada en en tokens. Para una distribución por tipos se obtuvo como promedio 7.2

Las características mencionadas son información relevante para poder construir un modelo más preciso dadas las estructuras de la lengua. Dichas características pueden o no estar presentes dependiendo de la letra que se esté viendo en ese momento. Por ejemplo, si es la primera letra de una palabra la que se observa, no estarán presentes las características `prevletter`, `prev2letters` o `EOW` por mencionar algunas. Otras características mencionadas como `letterLowercase` o `bias` siempre estuvieron presentes.

La construcción de las *feature lists* dependió del experimento en turno. Por ejemplo, en algunos casos fueron modificadas reduciéndolas e ignorando las etiquetas que brindaban información gramatical o fueron construidas para solo observar la letra actual y la anterior simulando un *Hidden Markov Model (HMM)* utilizando *CRFs*.

3.2.3. Implementación de CRFs y entrenamiento

Como mencionamos, los *CRFs* son una familia de métodos de aprendizaje basados en gráficas y discriminativos, en contraste con los modelos del tipo *HMM* que son generativos. Para la generación de secuencias de etiquetas utilizamos un tipo específico de *CRF* llamado *Linear-Chain CRF*. También se utilizó el algoritmo de optimización *L-BFGS* requerido por el modelo. Este algoritmo mejora los pesos de las funciones muy lentamente al comienzo de un proceso de entrenamiento, pero al final converge rápidamente en los pesos óptimos de las funciones.

El entrenamiento consistió en la búsqueda de un modelo que maximiza el logaritmo de verosimilitud con el método de aprendizaje *L-BFGS*. Es necesario entonces definir los parámetros del algoritmo de maximización. Los hiperparámetros modificados fueron los de *Elastic Net* con valores de regularización L_1 y L_2. También, fue configurado el número máximo de iteraciones para el algoritmo, pero fue fijado en 50 para todos los modelos. El resto de hiperparámetros se dejaron con un valor predeterminado. En la tabla 3.7 se muestran los valores que no se modificaron para los hiperparámetros.

Además de los hiperparámetros se configuraron diferentes entornos de experimentación dónde se construyeron las *feature lists* con más o menos información. En total fueron tres entornos y se describen a continuación. Cada entorno corresponde con el nombre inicial de los modelos.

linearCRF: Entorno que utiliza todas las *features* disponibles para la construcción de las *feature lists*. Las *features* pueden encontrarse en la des-

Hiperparámetro	Valor
`max_iterarions`	50
`num_memories`	6
`epsilon`	$1 * 10^{-5}$
`stop`	10
`delta`	$1 * 10^{-5}$
`linesearch`	"MoreThuente"
`max_linesearch`	20

Tabla 3.7: Valores de los hiperparámetros

cripción 3.2.2

POSLess: Entorno que solo ignora las etiquetas *POS* para la construcción de las *feature lists*

HMMLike: Entorno de información mínima para la construcción de las *feature lists*. Solo se toma en cuenta la letra actual y la letra anterior. Este entorno es una simulación un *HMM* ya que es construido utilizando *CRFs*.

Para encontrar el mejor modelo, adicional a los entornos de experimentación que describimos en con anterioridad, variamos los hiperparametros L_1 y L_2 en tres sentidos. Primero, fijamos $L_1 = 0.1$ y $L_2 = 1*10^{-3}$. Estas variantes corresponden a los modelos cuyo nombre tiene el sufijo `_reg`. A continuación, eliminamos la norma L_1 igualándola a cero. Estas variantes corresponden a los modelos con el sufijo `_l1_zero`. Repetimos el proceso con la norma L_2 igualándola a cero, dando como resultado los modelos con el sufijo `_l2_zero`. Por último, eliminamos ambos hiperparametros de regularización igualándolos a cero y teniendo como resultado los modelos con el sufijo `_no_reg`. En la tabla 3.8 se listan las diferentes configuraciones implementadas en esta tesis.

Terminada la fase de entrenamiento y construido el modelo de aprendizaje se puso a prueba con el conjunto destinado a este propósito y que fue previamente obtenido. Similar a la fase de entrenamiento, las entradas fueron las *feature lists* y la *BIO-labels* asociada a cada *feature list*. Utilizando el modelo de aprendizaje se etiquetó cada letra con base en la estructura de la lengua codificada en las *feature lists*. Las etiquetas generadas fueron comparadas con las reales y con ello se obtuvo el *accuracy* del modelo, entre otras medidas de desempeño.

Modelo	L1	L2	Features
`linearCRF_reg`	0.1	$1 * 10^{-3}$	Todas las disponibles
`linearCRF_l1_zero`	0	$1 * 10^{-3}$	Todas las disponibles
`linearCRF_l2_zero`	0.1	0	Todas las disponibles
`linearCRF_noreg`	0	0	Todas las disponibles
`POSLess_reg`	0.1	$1 * 10^{-3}$	Sin etiquetas *POS*
`POSLess_l1_zero`	0	$1 * 10^{-3}$	Sin etiquetas *POS*
`POSLess_l2_zero`	0.1	0	Sin etiquetas *POS*
`POSLess_noreg`	0	0	Sin etiquetas *POS*
`HMMLike_reg`	0.1	$1 * 10^{-3}$	Mínimas
`HMMLike_l1_zero`	0	$1 * 10^{-3}$	Mínimas
`HMMLike_l2_zero`	0.1	0	Mínimas
`baseline_HMMLike`	0	0	Mínimas

Tabla 3.8: Modelos de aprendizaje

Recapitulando, se obtuvo un corpus en otomí previamente glosado por un experto lingüista, después, en la etapa de codificación estructuramos la información de la lengua y en el preprocesamiento se crearon las *feature lists* que serán las entradas de los *CRFs*, continuamos con la fase de entrenamiento, creación del modelo de aprendizaje. Por último, se ejecutó la fase de evaluación comparando las *BIO-labels* generadas por el modelo y las reales presentes en el conjunto de pruebas, recuperando el *accuracy* del modelo. La arquitectura final se puede observar en la figura 3.5.

Software y hardware utilizado

En la fase de experimentación fue utilizado el paquete `python-crfsuite`, que utiliza como base la implementación de Okazaki (2007), para implementar *CRFs* en lenguaje de programación *Python* en su versión *3.7*. Esta fase corrió en una máquina con un procesador `Intel i7-7700HQ @ 3.800GHz` con `16 GB` de memoria principal. En promedio, una corrida de entrenamiento y evaluación tomó aproximadamente 5 minutos[4].

[4]El código utilizado para construir los modelos se encuentra publicado en el repositorio `https://github.com/umoqnier/otomi-morph-segmenter/`

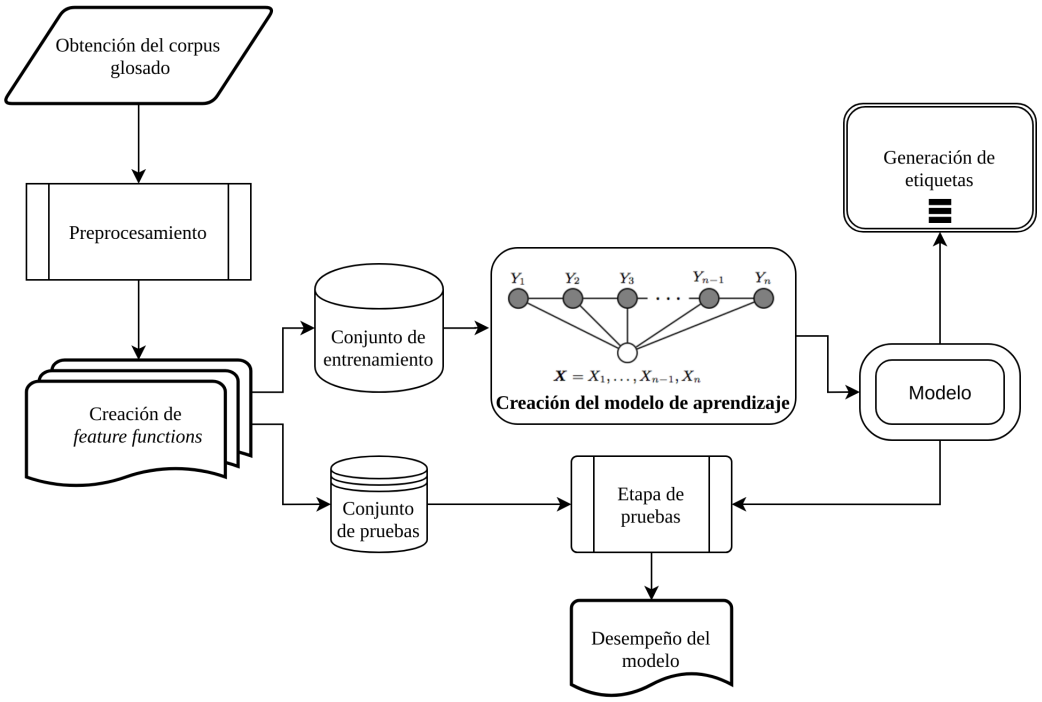

Figura 3.5: Arquitectura de aprendizaje

Talk, talk, it's only talk
Debates, discussions
These are words with a D this time
Dialog, duologue, diatribe
Dissention, declamation
Double talk, double talk

Elephant Talk - Discipline, King Crimson

4 Resultados

En este capítulo se examinará el método de evaluación utilizado para la construcción de los modelos de aprendizaje. Además, se detallará el desempeño de los modelos que fueron mencionados en el capítulo 3 y se profundizará en los resultados obtenidos en los diferentes entornos de experimentación.

4.1. Evaluación

Propusimos diferentes entornos de experimentación dónde fueron modificadas las *feature lists* y los parámetros de regularización *Elastic Net*. Como ya mencionamos en la subsección 3.2.3, se presentaron tres entornos de evaluación; en el primero se redujeron las *features* al mínimo, utilizando la información de la letra actual en la palabra y la letra anterior relativa a la letra actual, con lo que se simulaba un *Hidden Markov Model (HMM)*; en el segundo, se modificaron las *features* para ignorar las etiquetas *POS*, y en el último, se utilizaron todas las *features* disponibles. Las *features* fueron abordadas en la subsección 3.2.2.

Con el propósito de comparar el desempeño entre los modelos construidos, presentes en la tabla 3.8, seleccionamos como referencia el entorno dónde se simula un *HMM* que corresponde con el modelo `baseline_HMMLike_zero`.

Si se quiere saber que tan buenas son las predicciones de un modelo, es necesario validar su desempeño. La forma más sencilla de realizar esta validación es separar el corpus en dos conjuntos; el primer conjunto será destinado al entrenamiento del modelo y el segundo conjunto, independiente del conjunto de entrenamiento, tendrá el propósito de evaluar el desempeño del modelo. Realizar esta partición equivale a dividir aleatoriamente nuestro corpus en dos partes. Esta técnica de validación es conocida como *Hold Out*.

Sin embargo, este método de validación supone inconvenientes. Por un lado, es obligatorio sacrificar una parte del corpus para la realización de pruebas y, por otro lado, el desempeño del modelo podría mostrar valores muy optimistas o muy pesimistas dependiendo solo de cómo se realiza la partición del corpus.

Dado que en este trabajo nos enfrentamos a un entorno de *low resources* el método *Hold out* no es el más conveniente. Un método de validación que parece ser más adecuado es el de *K-fold cross-validation*. Esta técnica está diseñada para medir el desempeño de modelos sin sacrificar muchos datos.

En *K-fold* el corpus es dividido en K subconjuntos (*folds*), a diferencia de *Hold Out*, donde se divide solo en dos partes. Se toma un subconjunto para pruebas y el resto es usado para la etapa de entrenamiento. El modelo es entrenado y después se obtienen varias medidas de desempeño. Estas medidas serán abordadas más adelante.

El proceso se repite K veces hasta que cada subconjunto fue utilizado en la etapa de pruebas. Finalmente, se recolectan las medidas de desempeño de cada K. En la figura 4.1 se muestra una representación gráfica del método de evaluación con $K = 5$. Para todas las etapas de entrenamiento-evaluación utilizamos $K = 3$.

Fold 1	Pruebas	Entrenamiento	Entrenamiento	Entrenamiento	Entrenamiento
Fold 2	Entrenamiento	Pruebas	Entrenamiento	Entrenamiento	Entrenamiento
Fold 3	Entrenamiento	Entrenamiento	Pruebas	Entrenamiento	Entrenamiento
Fold 4	Entrenamiento	Entrenamiento	Entrenamiento	Pruebas	Entrenamiento
Fold 5	Entrenamiento	Entrenamiento	Entrenamiento	Entrenamiento	Pruebas

Figura 4.1: Representación gráfica de *K-fold cross-validation* con $K = 5$

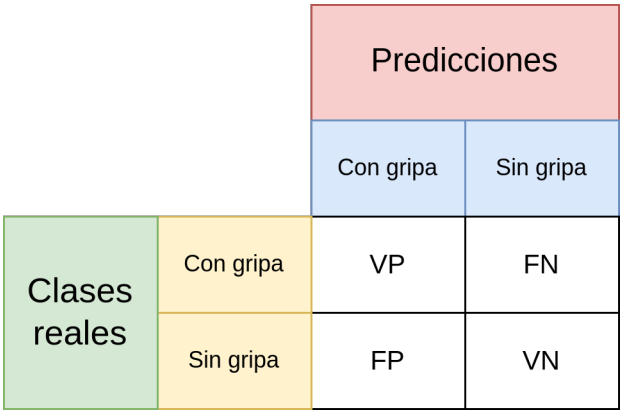

Figura 4.2: Ejemplo de matriz de confusión para un clasificador que indica si una persona tiene o no gripa

Consideramos exitosa la predicción si se logra maximizar la correcta clasificación de las etiquetas de salida. La correcta clasificación puede ser determinada con base en una amplia variedad de medidas de desempeño. Nosotros consideramos las siguientes: *accuracy*, *recall*, *precision* y *F1-score*. En este trabajo reportamos principalmente el *accuracy*, pero se utiliza el resto para detallar el análisis de resultados como se verá en la sección 4.2.

Para comprender estas medidas conviene mencionar un recurso llamado matriz de confusión[1], cuyo propósito, es brindar una visualización del desempeño de algún modelo.

La matriz de confusión es una tabla de dos columnas y dos filas. Las columnas representan las clases reales y las filas las clases predichas por el sistema o viceversa. Un ejemplo puede verse en la tabla 4.2. Esta tabla no es, por sí misma, una medida de desempeño, pero sirve como base para formar las medidas que mencionamos y otras más.

Cada celda de la matriz de confusión contiene un factor de evaluación que contabiliza ciertos casos. Los factores que reporta esta tabla son los casos verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos. A continuación describimos estos factores.

Verdadero Positivo (VP): Prueba que muestra correctamente la presencia de una clase. El sistema predijo que la persona tiene gripa y

[1]Se llama matriz de confusión porque puede verse rápidamente cuando el sistema confunde ciertas clases

efectivamente tiene gripa.

Falso Positivo (FP): Prueba que muestra incorrectamente la presencia de una clase. El sistema predijo que la persona tiene gripa cuando en realidad no tiene gripa.

Verdadero Negativo (VN): Prueba que muestra correctamente la ausencia de una clase. El sistema predijo que la persona no tiene gripa y efectivamente no tiene gripa.

Falso Negativo (FN): Prueba que muestra incorrectamente la ausencia de una clase. El sistema predijo que la persona no tiene gripa cuando en realidad si tiene gripa.

La combinación de estos factores permite análisis más detallados. Así pues, con base en estos factores, describiremos las medidas de desempeño utilizadas para evaluar los modelos de este trabajo.

El *accuracy* puede definirse como el número total de predicciones correctas dividido por el total de predicciones realizadas, donde, $0 <= accuracy <= 1$. La ecuación 4.1 muestra como obtenerlo. Para los resultados se realizó el promedio de los K *accuracy* obtenidos en cada prueba de los modelos. Este promedio por modelo es el que reportamos en la sección 4.2.

$$accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.1}$$

La *precision* es una medida que describe la relación entre las predicciones correctas y el total de predicciones positivas realizadas por el modelo, sean estas correctas o incorrectas, donde, $0 <= precision <= 1$. En la ecuación 4.2 se puede ver como obtener la *precision*.

$$precision = \frac{VP}{VP + FP} \tag{4.2}$$

El *recall* indica la porción de elementos positivos que el modelo es capaz de identificar correctamente. Toma en cuenta todas las clases positivas reales disponibles. Relaciona las predicciones positivas correctas y clases positivas incorrectamente clasificadas (los falsos negativos). El *recall* está definido en la ecuación 4.3. Por ejemplo, si se tiene un *recall* de 0.10 quiere decir que el modelo identificó correctamente el 10 % de las clases positivas.

$$recall = \frac{VP}{VP + FN} \tag{4.3}$$

Por último, el *F1-score* es una combinación de las medidas *precision* y *recall*. Esta combinación se da a través de la media armónica y está definida en la ecuación 4.4.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{4.4}$$

A continuación mostramos los modelos resultantes de cada etapa de experimentación propuestas en este trabajo.

En la tabla 4.1, con modelos del entorno *LinearCRF*, se muestran los modelos dónde se incluyó toda la información lingüística disponible en las *feature lists*.

Modelo	**Accuracy**	**Tiempo de entrenamiento**
`linearCRF_reg`	**0.9624**	**6.41[m]**
`linearCRF_l2_zero`	0.9598	5.41[m]
`linearCRF_noreg`	0.9586	5.43[m]
`linearCRF_l1_zero`	0.9586	5.39[m]

Tabla 4.1: Modelos con todas las *feature lists* disponibles

En la tabla 4.2, con modelos del entorno *POSLess*, se encuentran los modelos que pertenecen al entorno de experimentación donde fueron ignoradas las etiquetas *POS*.

Modelo	**Accuracy**	**Tiempo de entrenamiento**
`POS_Less_reg`	**0.9482**	**5.48[m]**
`POS_Less_l2_zero`	0.9472	5.21[m]
`POS_Less_l1_zero`	0.9442	5.17[m]
`POS_Less_noreg`	0.9407	5.49[m]

Tabla 4.2: Modelos donde se ignoran las etiquetas *POS*

En la tabla 4.3, con modelos del entorno *HMMLike*, se aprecian los modelos generados en el entorno de experimentación con las *feature functions* reducidas al mínimo, simulando *HMMs*.

Modelo	Accuracy	Tiempo de entrenamiento
HMMLike_l2_zero	**0.8800**	**5.83[m]**
HMMLike_reg	0.8760	6.04[m]
baseline_HMMLike_zero	0.8710	5.68[m]
HMMLike_l1_zero	0.8707	5.48[m]

Tabla 4.3: Modelos que simulan *HMMs*

4.2. Análisis de resultados

Como mencionamos, el entorno de experimentación dónde simulamos *HMMs* y, en concreto, el modelo `baseline_HMMLike_zero`, fue el que tomamos como base para poder comparar el desempeño de los *CRFs*.

Es importante destacar que, si bien este entorno de mínima información simula el método de generación de etiquetas *HMM*, no es, en sentido estricto, un *HMM*[2] ya que fueron utilizados los *CRFs* para su construcción.

El modelo con el *accuracy* más bajo se encuentra en el entorno *HMMLike*. Este es `HMMLike_l1_zero` con un 87.07 % de *accuracy*. Por otra parte, el modelo propuesto como base supera en al modelo `HMMLike_l1_zero` por apenas $3 * 10^{-4}$ unidades. Notamos que variando los hiperparametros L_1 y L_2 para este entorno de experimentación se obtuvo una mejora de apenas $9 * 10^{-3}$ respecto al modelo base.

Continuando con el segundo entorno de experimentación, *POSLess*, el *accuracy* mejoró de forma considerable respecto a los modelos que simulaban *HMMs*. Como se puede ver en la tabla 4.2 todos los modelos de este entorno lograron al menos un *accuracy* del 94 %. Observamos una mejora poco significativa con la variación de los hiperparametros L_1 y L_2. En este entorno el modelo con mayor *accuracy* fue `POS_Less_reg` con 94.82 %.

Por último, el entorno *linearCRF*, fue en el que se obtuvo el mejor desempeño. Los modelos generados en este entorno obtuvieron entre 95 % y 96 % de *accuracy*. Una vez más, la variación de hiperparametros de regularización comprobó mejoras leves en los modelos. Destacamos además que la mejora con respecto al entorno *POSLess* es de poco más del 2 % en los modelos con mejor desempeño del entorno *linearCRF*. En la tabla 4.4 se muestra la com-

[2]Una diferencia entre los *HMM* simulados con el modelo gráfico *CRF* y los *HMM* tradicionales radica en que los *CRFs* son grafos no dirigidos. Además, los *HMM* son modelos generativos, mientras que los *CRFs* son modelos discriminativos

paración entre el modelo base y los modelos con mejor *accuracy* en los otros entornos de experimentación.

Modelo	Accuracy
`linearCRF_l2_zero`	**0.9624**
`POS_Less_reg`	0.9482
`baseline_HMMLike_zero`	0.8710

Tabla 4.4: Comparación de modelos de diferentes entornos con mejor *accuracy*

Comparando el rendimiento de los modelos a través de los entornos de experimentación, observamos lo siguiente: entornos ricos en información, codificada en las *feature lists*, obtienen un desempeño mayor en comparación con los entornos donde se disponía de información mínima.

Con base en la arquitectura propuesta observamos que el modelo ***linearCRF_reg*** obtuvo el mejor *accuracy* y que supera el modelo base propuesto ***baseline_HMMLike_zero*** (Ver tabla 4.4). Del análisis a través de los entornos de experimentación podemos concluir, por un lado, que con la variación de hiperparametros en las tres familias de modelos (*linearCRF*, *POSLess*, *HMMLike*) (Ver tablas 4.1, 4.2, 4.3), se obtuvo una mejora en el *accuracy* poco significativa; por otra parte, el *accuracy* que muestran los modelos `POS_Less_reg` y `linearCRF_reg` es similar.

Entonces, parece ser que la configuración de los hiperparametros *Elastic Net* y las etiquetas *POS* (codificadas en las *feature functions*) mejoran levemente el desempeño de los modelos. Además, podemos concluir que el resto de información de la lengua, codificada también en las *feature lists*, es la que determinan, en gran medida, el desempeño de los *CRFs* para la correcta generación de *BIO-labels* y, por tanto, de glosa para el idioma otomí.

Sin embargo, si bien el entorno que incluyó toda la información fue el de mejor desempeño, éste no dista del desempeño que encontramos en el entorno que no incluye las etiquetas *POS*. Esto resulta prometedor, ya que es difícil encontrar estas etiquetas en un corpus, principalmente, porque el etiquetado manual es sumamente costoso en tiempo y esfuerzo.

Algo destacable es que las etiquetas *POS* no parecen ser tan relevante debido a que existen características morfológicas (patrones presentes en las mismas palabras) o sintácticas (secuencias de palabras) que indican la categoría de la palabra. Por ejemplo, los sustantivos en su mayoría son precedidos

de ‘ri̲’ o ‘yi̲’ (artículos singular y plural, respectivamente). Asimismo, los verbos tienen una morfología particular y existen patrones que muestran que se trata de un verbo: prefijos flexivos, sufijos, formativos temáticos, etc.

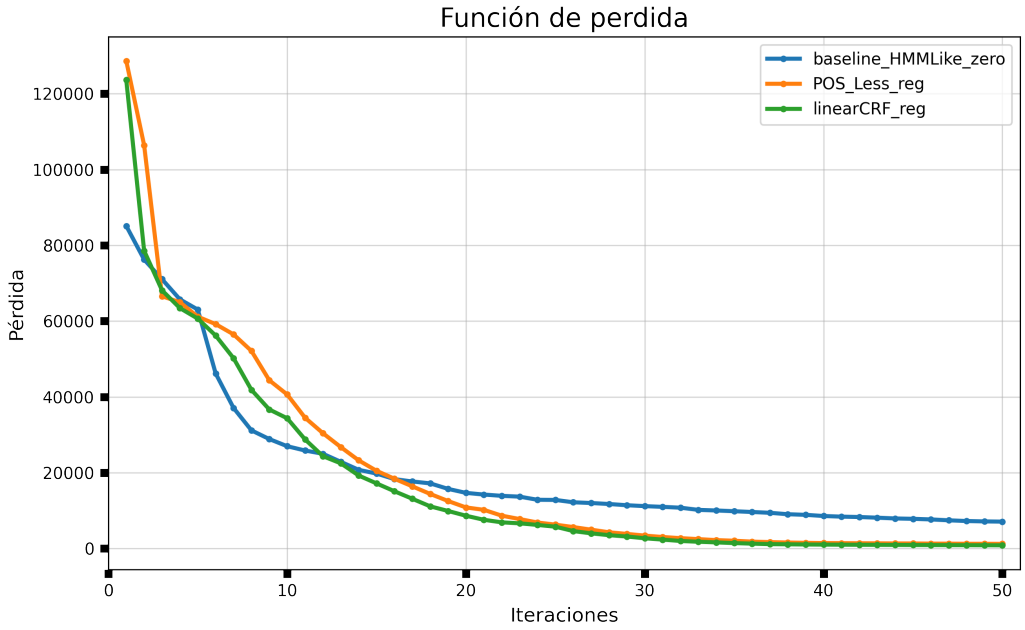

Figura 4.3: Función de perdida de los modelos con mejor desempeño

Mostramos una comparación de la función de perdida en la figura 4.3 de los modelos que se encuentran en la tabla 4.4. En la figura se encuentran las curvas para la iteración $K = 3$ de cada modelo.

En la figura 4.3 vemos como el modelo `baseline_HMMLike_zero` comienza con una pérdida baja en comparación con los otros dos modelos, pero que a medida que avanzan las iteraciones, la minimización del error que logra no es tan buena. Esto puede deberse a la cantidad de información codificada en la *feature lists* de inicio.

Por otro lado, las curvas de los modelos `POS_Less_reg` y `linearCRF_reg` reflejan un mejor desempeño que la del modelo base. Esto reafirma el desempeño observado en la tabla 4.4 que sugiere que a mayor información codificada en las *feature lists* el desempeño es mejor.

La información codificada en ambos modelos es casi la misma, salvo porque en los modelos *POSLess* se omitieron las etiquetas *POS*. Ergo, las curvas de los modelos `POS_Less_reg` y `linearCRF_reg` son similares.

En la tabla[3] 4.5 se observan las medidas *precision*, *recall* y *f1-score* que dan información detallada de la eficiencia del modelo `linearCRF_reg` en el etiquetado del último *fold*. Con base en dicha tabla hacemos las siguientes observaciones.

Etiqueta	**Precisión**	**Recall**	**F1-score**	**Instancias**
PRT	0.50	0.22	0.31	18
PL	0.94	0.74	0.83	39
2.CNT	1	1	1	6
LIG	0.93	0.76	0.84	111
3.CNT	1	1	1	25
1.OBJ	0.98	0.96	0.97	48
CTRF	0.95	0.97	0.96	89
STEM	0.96	0.96	0.96	2396
PRAG	0.97	0.99	0.98	116
3.ICP	0.93	0.94	0.94	118

Tabla 4.5: Métricas por etiqueta del modelo `linearCRF_reg`

Vemos un bajo desempeño en la etiqueta PRT ('partícula') el cual puede deberse a que la etiqueta hace referencia a cosas que no son sistemáticas, que son inusuales o raras y a que PRT puede aparecer en cualquier contexto. La etiqueta PL presenta un bajo *recall* debido a que los plurales tienen variaciones en la forma. Por ejemplo, algunas palabras plurales tendrán '-hí', o bien '-mí', incluso algunos pocas muestran plural '-phi'.

Por otro lado, las etiquetas como STEM o CTRF al ser sistemáticas y muy frecuentes se ven beneficiadas, por lo que tienen una *precisión* y un *recall* altos.

PRAG aparece en los verbos y siempre en la misma posición (al final de la palabra, en la posición del verbo) lo que hace que también sea fácilmente predicha por el modelo. Es interesante que este morfema es frecuente, pero si se quita no aporta mucha información. Podría decirse que es una muletilla que se traduce como 'pues'.

Finalmente, 3.ICP, es un morfema de tiempo muy frecuentemente usado (su equivalencia en español sería el tiempo presente en tercera persona). Siempre aparece antes del verbo y siempre en la misma posición. Por tanto, esta etiqueta es de las más altas en desempeño.

[3]Las etiquetas presentadas y sus significados se encuentran en los apéndices A y B

Cabe destacar que, por ejemplo, las etiquetas STEM, PRAG y 3.ICP, que son parte de las etiquetas con mayor desempeño, no necesariamente comparten la frecuencia. Mientras que STEM es la etiqueta con mayor frecuencia PRAG y 3.ICP muestran una frecuencia considerablemente más baja. Aun así, las tres etiquetas muestran *precision*, *recall* y *f1-score* altos.

En suma, vemos que el modelo tiene dificultades para asignar las etiquetas a frases que no son comunes y que carecen de variaciones contextuales para la desambiguación. Contrario, las que tienen alta *precision* y *recall* son sistemáticas, es decir, son frecuentes y consistentes en su distribución contextual, aunque vemos que la frecuencia no es un factor determinante. Entonces, las variaciones morfológicas y las etiquetas con apariciones no sistemáticas propician un bajo rendimiento en el etiquetado.

En la tabla 4.6 se muestran la *precision*, el *recall* y el *f1-score* que dan información detallada de la eficiencia del modelo `baseline_HMMLike_zero` en el etiquetado del último *fold*.

Etiqueta	**Precisión**	**Recall**	**F1-score**	**Instancias**
3.IMP	0.80	0.80	0.80	5
MED	0.62	0.71	0.67	7
MOD	0.30	0.14	0.19	21
DUAL.EXC	1	0.71	0.83	17
MUY	0.78	0.41	0.54	17
1.PRF	0.86	0.83	0.84	23
2.CNT	0.67	0.29	0.40	7
3.PRF	0.65	0.87	0.74	15
LOC	0.83	0.87	0.85	23
3.PLS	0.67	0.70	0.68	20
STEM	0.85	0.84	0.85	2504

Tabla 4.6: Métricas por etiqueta del modelo `baseline_HMMLike_zero`

Etiquetas como 3.PRF y MED tienen un *recall* alto, pero una *precisión* baja, lo que sugiere que el modelo está teniendo *overfitting*. En particular, la etiqueta MED puede verse afectada también porque aparece con variaciones entre 'n-' y 'm-'. Notamos que en general el bajo desempeño con estas etiquetas puede deberse a la baja frecuencia de instancias.

Destacamos que la *precision* de las etiquetas STEM es 11 % menor que la presente en el modelo, `linearCRF_reg` a pesar de tener una frecuencia alta,

con lo que reafirma que el desempeño de los modelos construidos con *CRFs* se ven beneficiados más por la información de la lengua codificada en las *feature functions* que por la frecuencia de las etiquetas.

Finalmente, el desempeño de etiquetado de este modelo es el más bajo. Dado que tenemos un bajo número de instancias de origen, podemos concluir que `baseline_HMMLike_zero` es más dependiente de la frecuencia que de la estructura. Esto tiene sentido dado que la información estructural dada para la construcción del modelo (a través de las *feature lists*) es mínima.

A continuación mostramos ejemplos de frases etiquetadas en los entornos de experimentación `linearCRF_reg` y `baseline_HMMLike_zero`

4.2.1. Ejemplos de etiquetado del modelo `linearCRF_reg`

(3) bu̲ m-bi-'u̲n-gí ya dó-ráhi-i̲-'wi
STEM MED-3.ICP-STEM-1.OBJ STEM 1.CPL-STEM
'Cuando me pegaban pues me quitaba'

En el etiquetado 3 vemos la aparición de etiquetas sistemáticas como STEM y 3.ICP, por mencionar algunas, por lo que la frase es correctamente etiquetada.

(4) a. g-i-né
psd-3.icp-STEM
'*Quería'

b. gi-né
2.POT-STEM
'Vas a querer'

En el ejemplo 4a se ve como el modelo hace una mala segmentación. Esto puede deberse, por un lado, a que la etiqueta 3.ICP es muy frecuente en el corpus y que dado que la etiqueta previa es PSD el modelo se inclina por estas etiquetas frecuentes. El patrón PSD-3.ICP es muy frecuente, por lo que el modelo tiende a utilizarlo causando un mal etiquetado. Por otra parte, la combinación correcta de etiquetas (2.POT-STEM) es poco frecuente. La versión del etiquetado esperado se puede ver en el ejemplo 4b

Curiosamente, la letra 'g' por sí misma no coincide con ningún patrón del tiempo pasado. Por otro lado, la letra 'i' que sigue si corresponde a un 3.ICP por lo que el modelo está haciendo una inferencia regresiva. Como la letra 'i' es 3.ICP la letra anterior debería ser etiquetada como PSD.

Es notorio que en múltiples casos en los que hay morfemas alrededor del verbo, el modelo tiende a considerarlos parte del STEM en lugar de separarlos. Esto quiere decir que toma morfemas que no deberían ser parte del STEM. Tiene sentido en tanto que el STEM no tiene una forma determinada (es una clase abierta) y puede tener cualquier combinación de letras. Un ejemplo de esto se puede ver en la glosa 5a.

Las características de las clases abiertas hacen que sean difíciles de ver por completo en el entrenamiento, contrario a las clases cerradas que son finitas. Además, notamos que el modelo `linearCRF_reg` se equivoca cuando no hay suficiente contexto como en frases cortas de una sola palabra.

(5) a. bi-nd̲uzú
3.cpl-stem
(La segmentación no es traducible ,porque, la raíz nd̲uzú, no corresponde con ninguna palabra ya que fue etiquetada erróneamente)

b. bi-nd̲u-zú
3.ICP-MUY-STEM
'Se asustó'

4.2.2. Ejemplo de frase etiquetada por el modelo `baseline_HMMLike_zero`

(6) bi-'̲un-gi yi̲ mb̲uhi nge hín dí-má-né
3.CPL-STEM-1.OBJ DET.PL STEM STEM STEM 1.ICP-CTRF-STEM
gwa-porá nge dí-má-dáhní
1.ICP.IRR-STEM STEM 1.ICP-CTRF-STEM
'Me pegaron los patrones porque no me quería apurar, porque era flojo'

Ya mencionamos que el desempeño de este modelo, que simula un *HMM*, depende de la frecuencia. El ejemplo 6 muestra etiquetas y patrones muy frecuentes como CPL y CTRF que en general son también sistemáticos. A pesar de que el etiquetador pertenece al entorno con menor información y desempeño, el etiquetado es correcto. Además, ayuda que, como se observó, los ejemplos largos son mejores para estos modelos de baja información.

(7) a. má-ndé bi-ni
ctrf-stem 3.CPL-STEM

‘*La tarde lo tuve’

b. mánde bi-ni
STEM 3.CPL-STEM
‘Ayer lo tuve’

En el ejemplo 7a vemos como el modelo está sobre ajustado, ya que el morfema ‘ma’ está siendo asociado con la etiqueta CTRF dado que esta combinación es muy frecuente. Incluso el morfema ‘ndé’ es un STEM existente y válido pero incorrecto en este contexto. Notamos que ocurre el efecto contrario a lo que vimos en el ejemplo 5a. Esto es que, ya que ha visto el morfema ‘ma’ separado con tanta frecuencia en este ejemplo, también lo separa en lugar de integrarlo en uno solo y etiquetarlo como STEM.

Un error característico de este modelo es que está sobre segmentando y encontrando morfemas que no son correctos. Confunde secuencias con patrones frecuentes y no es raro porque a este modelo se le otorgó mínima información, por lo que no está tomando en cuenta el contexto de las oraciones. Con más información esto se reduce.

Por último, notamos que en este entorno de experimentación los modelos no aprenden a formar correctamente las estructuras válidas de las *BIO labels* para generar la glosa. Esto es que la primera etiqueta debe comenzar con una letra `'B'` (*Beginning*) seguida opcionalmente de etiquetas que comienzan con la letra `'I'` (*Inside*). Mostramos una la salida de etiquetas generadas por el modelo en el ejemplo 8.

(8)

i	b	i	t	h	ó	p	a	n
I-it	B-3.CPL	I-3.CPL	**B-ila**	**I-ila**	**I-ila**	B-STEM	I-STEM	**B-stem**
i	r	i	c	h	á	i		
I-stem	B-DET	I-DET	B-STEM	I-STEM	I-STEM	I-STEM		

Como podemos ver, la primera etiqueta comienza con una letra `'I'` cuando debería ser una `'B'`. La salida con las *BIO-labels* bien formadas sería la que se ve en el ejemplo 9.

(9)

i	b	i	t	h	ó	p		
B-STEM	B-3.CPL	I-3.CPL	B-STEM	I-STEM	I-STEM	B-STEM		
a	n	i	r	i	c	h	á	i
I-STEM	B-DET	I-DET	B-DET	I-DET	I-DET	B-STEM	I-STEM	I-STEM

Con las *BIO-labels* que se muestran en el ejemplo 9 se tendría la glosa que se muestra en el ejemplo 10.

(10) i bi-thó pa nı̲ rı̲ chái
STEM 3.CPL-STEM STEM DET DET STEM
'Y paso por una zanja'

En resumen, configuramos tres entornos de experimentación dónde variamos la información con la que construimos las *feature lists* y algunos hiperparámetros del modelo. Usamos como base un modelo simulación de los *HMM*.

Explicamos que para probar el desempeño de los modelos utilizamos la técnica de validación llamada *k-fold cross validation* con un $K = 3$. Adicionalmente, presentamos las medidas de desempeño que utilizamos que son *accuracy*, *recall*, *precision* y *f1-score* siendo el *accuracy* el reportado en este trabajo con el resto de medidas como complementos para el análisis.

Por otro lado, observamos que el modelo con mejor desempeño se encontró en el entorno con máxima información lingüística (*LinearCRF*). Después, los modelos dónde se ignoraron las etiquetas *POS* (*POSLess*) y por último los modelos de mínima información (*HMMLike*).

A pesar de que los modelos no realizan un etiquetado perfecto, el desempeño mostrado, es suficientemente bueno. Estos modelos pueden ser una herramienta auxiliar en el etiquetado automático de textos en otomí. Además, el análisis de los errores cometidos por los modelos pueden ser de ayudar para mejorar el desempeño aquí presentado.

Talk, talk, it's all talk
Too much talk
Small talk
Talk that trash
Expressions, editorials
Explanations, exclamations, exaggerations
It's all talk
Elephant talk (x3)

Elephant Talk - Discipline, King Crimson

5 Conclusiones

Este trabajo se enfocó en la tarea de la creación de un glosador automático para el otomí de Toluca, una lengua indígena mexicana con fenómenos morfológicos complejos. Nos enfrentamos a un escenario de bajos recursos digitales donde se recogió un pequeño corpus previamente anotado por un experto lingüista.

La hipótesis de este trabajo fue que utilizar métodos de aprendizaje estructurado y basados en grafos, como los *Conditional Random Fields*, permitiría el desarrollo de modelos con un mejor desempeño que métodos tradicionales como los *HMM*. En ese sentido, aplicamos etiquetadores basados en *CRFs* con diferentes variaciones con respecto a las características que fueron tomadas en cuenta por el modelo. Además, la variación que simulaba un *HMM* fue utilizada como referencia para comparar los modelos.

Como vimos, usando métodos de aprendizaje como los *CRFs* obtuvimos resultados prometedores para abordar una tarea típica del *NLP*, el etiquetado automático de textos. El etiquetado puede realizarse a diferentes niveles lingüísticos. En este trabajo buscamos que los modelos generaran etiquetas de glosa interlineal. Este es un etiquetado de gran importancia tanto para el análisis lingüístico como para el desarrollo de futuras tecnologías del lenguaje para lenguas de bajos como el otomí.

Para la predicción de etiquetas implementamos una arquitectura que con-

templaba la obtención de corpus previamente glosado, preprocesamiento del texto para construir las *feature functions*, separación del corpus en conjuntos de entrenamiento y pruebas y, la generación y evaluación de modelos comparando la glosa real con la glosa predicha.

En general, los modelos que tomaron en cuenta más información (codificada en las *feature functions*) obtuvieron un mejor desempeño. Esto puede verse en el modelo `linearCRF_reg` que supero en desempeño al modelo de referencia `baseline_HMMLike_zero`.

Podemos concluir que cuando se reduce la información codificada en las *feature lists* la frecuencia de las instancias tiene mayor peso. Por lo tanto, para bajos recursos digitales, donde las frecuencias de las instancias son bajas, es necesario dar un contexto más amplio y agregar información lingüística más detallada.

Sin embargo, notamos que excluir las etiquetas *POS* no perjudica mucho el desempeño del sistema. Esto puede ser una ventaja dado que las etiquetas *POS* son un recurso difícil de obtener y herramientas para el etiquetado automático *POS* no siempre están disponibles para lenguajes de bajos recursos digitales.

Nuestro glosador fue capaz de conseguir una precisión de 96.24 % (y 94.82 % sin etiquetas *POS*) con lo que podemos confirmar que nuestra hipótesis es correcta. Esto suena prometedor para reducir el trabajo del glosado manual y puede representar un paso intermedio no solo para fortalecer la documentación lingüística, si no, también, para facilitar la creación de tecnologías del lenguaje que pueden ser de utilidad para los hablantes del otomí.

5.1. Trabajo Futuro

El desarrollo el glosador automático fue fuertemente apoyado por los *CRFs*. Como mencionamos, para comparar el desempeño tomamos como base un modelo que simulaba un *HMM* con los *CRFs*. Un trabajo a futuro interesante sería comparar el desempeño de los *CRFs* con otras arquitecturas de aprendizaje como *HMM* tradicionales (no simulados como los de este trabajo) o redes neuronales.

Un buen experimento sería aplicar la arquitectura de glosado automático propuesta a otras lenguas de bajos recursos digitales como el maya, wixárika y guaraní, solo por mencionar algunas. Como un paso hacia adelante en la distribución del desarrollo de esta tesis, se pretende publicar la arquitectura a

través de un paquete para el lenguaje python[1] para su fácil uso y distribución.

[1] `https://pypi.org/project/elotl/`

Bibliografía

Antonis Anastasopoulos, Marika Lekakou, Josep Quer, Eleni Zimianiti, Justin DeBenedetto, y David Chiang. 2018. Part-of-speech tagging on an endangered language: a parallel griko-italian resource.

Guadalupe Barrientos López. 2004. *Otomíes del Estado de México*. Comisión Nacional para el Desarrollo de los Pueblos Indígenas.

Igor A Bolshakov y Alexander Gelbukh. 2004. *Computational linguistics: models, resources, applications*. Instituto Politecnico Nacional.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on information theory* 2(3):113–124.

Bernard Comrie, Martin Haspelmath, y Balthasar Bickel. 2008. The leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses. *Department of Linguistics of the Max Planck Institute for Evolutionary Anthropology & the Department of Linguistics of the University of Leipzig. Retrieved January* 28:2010.

Marc Peter Deisenroth, A Aldo Faisal, y Cheng Soon Ong. 2020. *Mathematics for machine learning*. Cambridge University Press.

Comunidad Elotl. 2019. Procesando el otomí (hñähñu) £dónde empezar? Acceso: 2020-05-08.

W Nelson Francis y Henry Kucera. 1979. Brown corpus manual. *Letters to the Editor* 5(2):7.

Boris Giba. 2021. Elastic net regression explained, step by step. Acceso: 2022-06-22.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, y Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM* 63(11):139–144.

Daniel Jurafsky y James H Martin. 2008. Speech and language processing: An introduction to speech recognition, computational linguistics and natural language processing. *Upper Saddle River, NJ: Prentice Hall* .

Daphne Koller y Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

John Lafferty, Andrew McCallum, y Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

Yolanda Lastra. 1992. *El otomí de Toluca*. Instituto de Investigaciones Antropológicas, UNAM.

Manuel Mager, Ximena Gutierrez-Vasques, Gerardo Sierra, y Ivan Meza-Ruiz. 2018. Challenges of language technologies for the indigenous languages of the Americas. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, pages 55–69. https://www.aclweb.org/anthology/C18-1006.

Christopher Manning y Hinrich Schutze. 1999. *Foundations of statistical natural language processing*. MIT press.

Tony McEnery y Andrew Wilson. 2003. Corpus linguistics. *The Oxford handbook of computational linguistics* pages 448–463.

Victor Mijangos de la Cruz. 2020. Notas de procesamiento de lenguaje natural. Acceso: 2020-11-02.

Deep Mind. 2021. Alphago: The story so far. Acceso: 2021-12-04.

Tom M Mitchell et al. 1997. Machine learning.

Sarah Moeller y Mans Hulden. 2018. Automatic glossing in a low-resource setting for language documentation. In *Proceedings of the Workshop on Computational Modeling of Polysynthetic Languages*. pages 84–93.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Jeff Pitman. 2021. Google translate: One billion installs, one billion stories. Acceso: 2021-10-11.

Prabhakar Raghavan. 2021. How ai is making information more useful. Acceso: 2021-10-11.

Stuart J Russell y Peter Norvig. 2010. Artificial intelligence-a modern approach, third international edition.

Tanja Samardzic, Robert Schikowski, y Sabine Stoll. 2015. Automatic interlinear glossing as two-level sequence classification. In *Proceedings of the 9th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*. pages 68–72.

Shai Shalev-Shwartz y Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27(3):379–423.

Conor Snoek, Dorothy Thunder, Kaidi Loo, Antti Arppe, Jordan Lachler, Sjur Moshagen, y Trond Trosterud. 2014. Modeling the noun morphology of plains cree. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*. pages 34–42.

Charles Sutton, Andrew McCallum, et al. 2012. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning* 4(4):267–373.

Alan Turing. 1950. Computing machinery and intelligence-am turing. *Mind* 59(236):433.

María Ximena Gutiérrez Vasques. 2018. Extracción léxica bilingüe automática para lenguas de bajos recursos digitales .

Aston Zhang, Zachary C. Lipton, Mu Li, y Alexander J. Smola. 2020. *Dive into Deep Learning*. `https://d2l.ai`.

Xingyuan Zhao, Satoru Ozaki, Antonios Anastasopoulos, Graham Neubig, y Lori Levin. 2020. Automatic interlinear glossing for under-resourced languages leveraging translations. In *Proceedings of the 28th International Conference on Computational Linguistics*. pages 5397–5408.

A

Etiquetas *POS*

Tabla con las etiquetas *POS* utilizadas en este trabajo y sus significados. Dentro del corpus se encuentra glosa que no corresponde con etiquetas *POS*, si no, con una descripción semántica (traducción). Estas etiquetas no son presentadas en este apéndice por ser descriptivas en sí mismas.

Etiqueta	**Significado**	**Etiqueta**	**Significado**
v	verbo	obl	oblicuo
det	determinante	cnj	conjunción
dem	demostrativo	unkwn	desconocido
n	sustantivo	neg	negativo
p.loc	partícula locativa	prt	partícula
conj.adv	conjunción adversativa	dim	diminutivo
gen	genitivo	cond	condicional
it	iterativo	lim	limitativo
aff	afirmativo	loc	locativo
dec	decimal	conj	conjunción
cord	coordinación	cnj.adv	conjunción adversativa
regular/v	verbo regular		

Tabla A.1: Descripción de etiquetas *POS*

B
Glosa

Tabla con las etiquetas de glosa utilizadas en este trabajo y su descripción. Estas etiquetas están basadas en el estándar de (Comrie et al., 2008) desarrolladas por el departamento de lingüística del Instituto Max Planck y el Departamento de lingüística de la Universidad de Leipzig. El estándar consiste en diez reglas para la sintaxis y la semántica de glosas interlineales y un apéndice con un lexicón propuesto de etiquetas de categorías abreviadas (Comrie et al., 2008). En esta tabla se omitieron las variaciones de etiquetas con personas gramaticales para compactarla.

Glosa	**Significado**	**Glosa**	**Significado**
stem	base	ctrf	contrafactual
cpl	completivo	dem	demostrativo
icp	incompletivo	dim	diminutivo
pot	potencial	ila	ilativo
ctn	continuativo	mod	modo
prf	perfecto	loc	locativo
pls	pluscuamperfecto	prt	partícula
irr	irrealis	it	iterativo
imp	imperativo	enf	enfático
psd	pasado	neg	negativo
pl	plural	int	interrogativo
sg	singular	aum	aumentativo
ex	exclusivo	gen	genitivo
pss	posesivo	com	comitativo
obj	objeto	adj	adjetivo
med	voz media	encl	enclítico
dual	número dual	enf	enfático
det	determinante	caus	causativo
lim	limitativo	comp	comparativo
lig	ligadura	dcl	declarativo
prag	partícula pragmática		

Tabla B.1: Descripción de Glosa