Etiquetador automático de la morfología del otomí usando predicción estructurada

Diego Alberto Barriga Martínez

25 de agosto de 2020

Índice general

Capítulo 1

Introducción

1.1. Lengua otomí

En esta sección se menciona los lugares donde se describe el idioma otomí de forma somera, se mencionan algunos lugares donde es hablado el otomí y características fundamentales de la lengua.

1.1.1. Origen

La palabra otomí es de origen náhuatl (singular: *otomitl*, plural: *otomí*). Por otra parte, los otomíes se nombran a sí mismos *ñähñu*[1], que significa "los que hablan otomí".

Los grupos indígenas que hablan el idioma otomí se encuentran en diversas partes del territorio mexicano como: Estado de México, Querétaro, Hidalgo, Puebla y Veracruz (Barrientos López, 2004). El otomí es una lengua indígena una gran variación dialectal que depende de su distribución geográfica.

En el Estado de México el pueblo *ñähñu* está disperso por varios municipios tales como: Toluca, Lerma, Chapa de Mota, Aculco, Amanalco, Atizapán de Zaragoza, por mencionar algunos. En otros municipios como Naucalpan, Ecatepec, Nezahualcóyotl y Tlalnepantla se pueden encontrar hablantes por efectos de la migración. Según Barrientos López (2004) la población total de hablantes otomíes en el Estado de México supera los cien mil, sin embargo,

[1]Existen organizaciones indígenas, como el Consejo de la Nacionalidad Otomí, que escriben la auto-denominación como hñätho hñähñu y también ñätho ñähño. Sin embargo, esta auto denominación puede variar.

datos actuales .

En concreto existen **nueve** variantes del otomí y cabe recalcar que dicha variación puede presentarse incluso dentro del mismo estado. Tan solo el Estado de México presenta tres variantes del otomí: El otomí de Tilapa, hablado en el municipio de Santiago Tianguistenco; el Otomí de Acazulco, del municipio de San Jerónimo Acazulco; y el Otomí de Toluca, de San Andrés Cuexcontitlán.

1.2. Problemática

El 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 generales consisten en permitir la comunicación humano-máquina, o simplemente hacer un exitoso procesamiento de texto o voz humanos (**?**). 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.

Una de las tareas más populares de NLP es el etiquetado automático de textos. Este etiquetado puede realizarse a diferentes niveles lingüísticos, por ejemplo, morfosintáctico (*Part Of Speech tagging, POS*), sintáctico (*parsing*), morfológico, etc. El nivel morfológico tiene que ver con la estructura interna de las palabras (**?**); en particular, existe un tipo de etiquetado, de gran importancia para el análisis lingüístico, llamado glosado que asigna etiquetas a las unidades que conforman a una palabra.

Para lograr lo anterior, los enfoques actuales aplican técnicas de ML. El ML es un subcampo de la Inteligencia Artificial (IA), que constituye un enfoque de resolución de problemas caracterizado por estimar una solución a partir de la experiencia (**?**). La experiencia se refiere a datos etiquetados (ejemplos) que permiten inferir un modelo estadístico de aprendizaje. Entre los métodos de ML ampliamente utilizados se pueden mencionar las Support Vector Machines (SVMs), árboles de decisión, o bien los modelos gráficos, como las redes neuronales o los métodos generativos, solo por mencionar algunos. Para las tareas de etiquetado en NLP generalmente se utilizan modelos gráficos supervisados, por ejemplo, modelos ocultos de Markov (*Hidden Markov Models, HMM*).

No obstante, el lenguaje natural es complejo y dinámico, ya que tiene fenómenos que hacen que las tareas de reconocimiento, generación y procesamiento se vuelvan difíciles para las computadoras. Adicionalmente, existen escenarios donde estos métodos no son efectivos como es el caso de las lenguas de bajos recursos, que son lenguas que tienen pocos recursos digitales con los que trabajar. Por ejemplo, si se tienen pocos datos iniciales para el entrenamiento del modelo de aprendizaje las predicciones serán poco precisas o equivocadas. Los bajos recursos son un escenario común en México donde, a pesar de que existe una rica diversidad lingüística, gran parte de las lenguas originarias no poseen contenido web ni publicaciones digitales y por tanto carecen también de tecnologías del lenguaje. El escenario mencionado anteriormente supone un reto para los métodos de aprendizaje convencionales, que requieren de grandes cantidades de datos de entrenamiento para funcionar correctamente. Por lo tanto, es un importante reto de investigación desarrollar aproximaciones que funcionen con lenguas de escasos recursos. En particular, en este trabajo nos enfocamos en el glosado automático del otomí, una lengua con gran riqueza morfológica y con escasez de recursos digitales.

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.

1.3. Objetivo

Diseñar e implementar un etiquetador morfológico para el otomí basado en técnicas de Procesamiento del Lenguaje Natural (*Natural Language Processing, NLP*) con Aprendizaje de Máquina (*Machine Learning, ML*). En particular, se hará énfasis en métodos de aprendizaje estructurado débilmente supervisados. Específicamente, se aplicará *Conditional Random Fields (CRF)* para etiquetado morfológico (glosado) del otomí, una lengua de bajos recursos.

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

1.4. Hipótesis

Se espera obtener un modelo que produzca glosa para el otomí, generada automáticamente, con base en el entrenamiento con pocos ejemplos previamente etiquetados. Al obtener una buena exactitud en la predicción automática de glosa se apoyaría a los anotadores humanos a reducir trabajo repetitivo y exhaustivo. Además, se espera obtener avances de una metodología adaptable a un mayor número de lenguas mexicanas. Sería deseable que esta metodología experimental pueda ser replícale en otras lenguas habladas en México.

Capítulo 2

Avances en etiquetadores automáticos

2.1. Marco teórico

2.1.1. Natural Language Processing (NLP)

2.1.2. Etiquetadores

2.1.3. Machine Learning (ML)

2.1.4. Modelos gráficos

Los límites de los modelos gráficos en para bajos recursos

En este capítulo se explicará qué ventajas tienen los *Conditional Random Fields (CRF)* sobre otros modelos de aprendizaje, se mencionan formalmente los elementos fundamentales que describen los *CRF's*.

En lingüística computacional una tarea de interés es el procesamiento estadístico del lenguaje natural, en particular, el etiquetado y segmentación de secuencias de datos. En ese sentido, es habitual la utilización de **modelos generativos**, cómo los *Hidden Markov Models (HMMs)*, o **modelos condicionales**, como los *Maximum Entropy Markov Models (MEMMs)*.

Por una parte, los modelos generativos intentan modelar una probabilidad conjunta $P(x, y)$ sobre observaciones y etiquetas. Para definir esta probabilidad conjunta se necesita enumerar todas las observaciones posibles. Las limitantes de este enfoque son de diversas índoles como las grandes dimen-

sionalidades en el vector de entrada X, la dificultad de representar múltiples características que interactúan unas con otras y dependencias complejas que hacen la construcción de la distribución de probabilidad un problema intratable con un enfoque computacional.

Por otro lado, una solución a las limitantes de los modelos generativos es un modelo condicional. 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.

Consecuencia de lo anterior, no se gasta esfuerzo en modelar las observaciones, dado que en al momento de realizar pruebas estas observaciones son fijas. Segundo, 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).

Un ejemplo de estas ventajas con los *MEMMs* que son modelos secuenciales de probabilidad condicional. Sin embargo, estos modelos y otros que son no generativos, de estados finitos y que son clasificadores basados en el estado siguiente 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".

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.

2.1.5. Conditional Random Fields

Como menciona Sutton et al. (2012) modelar las dependencias entre las entrada puede conducir a modelos intratables, pero ignorar estas dependencias puede reducir el rendimiento.

Dado el que problema abordado en este trabajo, dónde se requiere del etiquetado de secuencias y es en contexto de bajos recursos lingüísticos, se hace necesario utilizar un enfoque más conveniente.

Los *Conditional Random Fields (CRFs)* son un framework para la creación de modelos probabilístico utilizado en técnicas de aprendizaje estructurado. Tienen las ventajas de los *MEMMs* y, en principio, solucionan el *label*

bias problem. El framework tiene un solo modelo exponencial para la probabilidad conjunta de todas las secuencias de las etiquetas de salida dada la secuencia de observación. En contraste los *MEMMs* usan modelos exponenciales para cada probabilidad condicional de los estados siguientes dado el estado actual.

Formalmente Lafferty et al. (2001) definen los *CRFs* como a continuacion se enuncia:

Definición 1. Sea $G = (V, E)$ una gráfica tal que $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$, entonces esa $\mathbf{Y}$ es indexada por los vertices de G. Entonces $(\mathbf{X}, \mathbf{Y})$ es un conditional random field en caso de que las variables aleatorias $\mathbf{Y}$ se condicionen por $\mathbf{X}$, la variable aleatoria $\mathbf{Y}_v$ cumple la *propiedad de Markov* con respecto a la gráfica: $p(\mathbf{Y}_v|\mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v|\mathbf{X}, \mathbf{Y}_w, w \sim v)$, dónde $w \sim v$ significa que w y v son vecinos en G.

En esta tesis, para el modelado de secuencias, se utiliza la forma más sencilla de la gráfica G dónde es una cadena simple o línea. Esto quiere decir que $G = (V = \{1, 2, ...m\}, E = \{(i, i + 1)\})$. A este tipo de *CRFs* se les conoce como *linear-chain CRFs*. Como menciona Lafferty et al. (2001) "si la gráfica $G = (V, E)$ de $\mathbf{Y}$ es un árbol (del cual una cadena es el ejemplo más sencillo), los *cliques* son los límites y vertices. Entonces, por el teorema de los *random fields* (Hammersley y Clifford, 1971), la distribución conjunta sobre las etiquetas de secuencias $\mathbf{Y}$ y $\mathbf{X}$ tiene la forma:

$$p_\theta(y|x) \propto \left(\sum_{e \in \mathbf{E}, k} \lambda_k f_k(e, \mathbf{y}|_e, \mathbf{x}) + \sum_{v \in \mathbf{V}, k} \mu_k g_k(v, \mathbf{y}|_v, \mathbf{x}) \right) \tag{2.1}$$

De la ecuación se destacan f_k y g_k que representan las *feature functions*. Estas están definidas y son fijas. Las *feature functions* de está tesis serán descritas más adelante.

2.2. Estado del arte

Los *CRFs* han sido utilizados para la clasificación de regiones en una imagen, estimar el puntaje en un juego de Go, segmentar genes en una hebra de ADN y análisis sintáctico de lenguaje natural en un texto por mencionar algunas (Sutton et al., 2012).

2.2.1. Trabajos sobre bajos recursos

Con base en el trabajo previo de (Moeller y Hulden, 2018) donde se utilizan técnicas de NLP y ML para tratar para el idioma Lezgi se plantea como hipótesis que dado el tamaño del corpus y la glosa que contiene se obtendrá texto correctamente glosado con una precisión de al menos 80 %.

Diego: Descripción de la estimación de parámetros y característica de la función de perdida

Capítulo 3

Etiquetador morfológico para el otomí (Metodología)

El objetivo de este capítulo es la presentación de la metodología utilizada en esta tesis. La metodología está constituida por 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 del los *CRFs* y la determinación de las *feature functions*.

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 normali-

Categoría	**Cuenta**
Tokens (POS)	8578
Tipos (POS)	44
Tokens (Glosa)	14477
Tipos (Glosa)	112
Total de oraciones etiquetadas	**1786**

Tabla 3.1: Tamaño del corpus

zados y homogéneos. Dicha normalización propicia la obtención del mejor desempeño posible en los diversos métodos de aprendizaje automático. 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 y de Suárez (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-otomi que se encuentra en la plataforma web Tsunkua[2].

Además, se agregaron 81 lineas de casos poco usuales que son fenómenos poco frecuentes y, por tanto, particularmente difíciles de predecir. El subconjunto del corpus utilizado en la sección experimental está descrito en la tabla 3.1, donde se encuentra el tamaño de las etiquetas POS y el tamaño de la glosa y en la tabla 3.2, donde se puede 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 masculino (Lastra y de Suárez, 1992).

Los tipos de etiquetas *POS* presentes en el corpus se pueden observar en la tabla 3.3. Dentro del corpus se encontrón 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

[1]TODO: Liga del repo

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

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

Tabla 3.2: Textos del corpus

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*

prinsipal..." ("...empezado un Zapata, pero lo principa...") 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.

Por otra parte, presentamos una descripción de las etiquetas *POS* en la tabla 3.4 dónde se muestra el significado de cada una. 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 considerara 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 **?** 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 (**?**). Si bien las reglas cubren parte de las necesidades 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 3.5. En algunos casos aparecen números al inicio de etiquetas que significan las personas gramaticales. Existen combinaciones de varias etiquetas

Etiqueta	**Significado**	**Etiqueta**	**Significado**
v	verbo	obl	oblicuo
det	determinante	cnj	conjunción
dem	demostrativo	unkwn	desconocido
n	sustantivo	neg	negativo
p.loc	partícula locativoa	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 3.4: Descripción de etiquetas *POS*

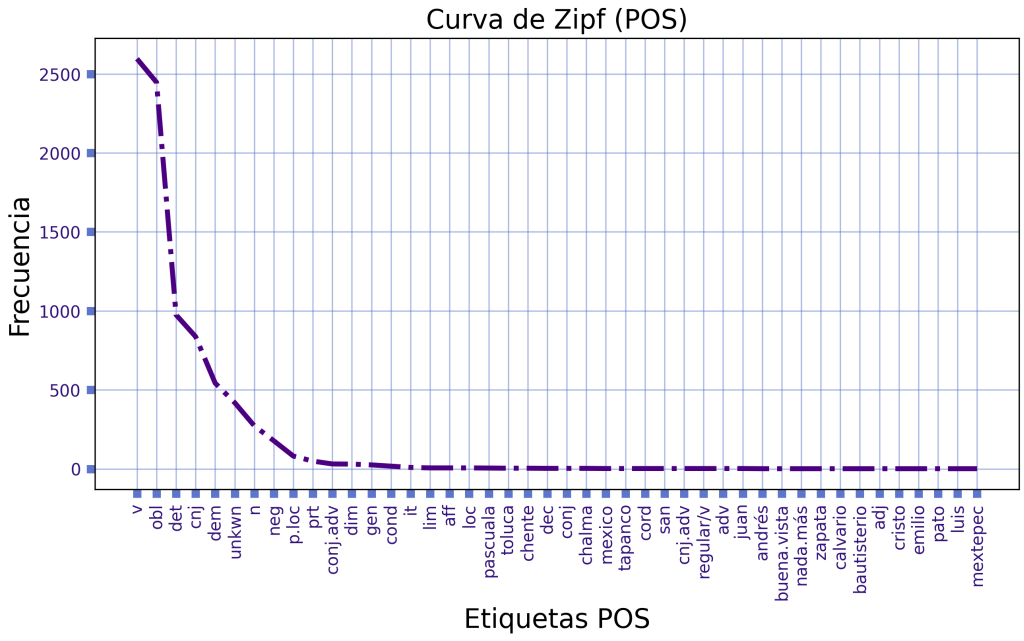

Figura 3.1: Distribución de etiquetas POS

stem	det	3.cpl	psd	lim
prag	3.icp	lig	det.pl	1.icp
3.pot	ctrf	1.pot	pl.exc	1.cpl
dem	1.pss	dim	pl	1.obj
ila	2.icp	1.prf	3.cnt	3.obj
loc	mod	1.cnt	3.pls	prt
it	dual.exc	3.prf	3.icp.irr	3.pss
2.pss	1.enf	med	dual	p.loc
2.cnt	2	3.imp	int	neg
1.icp.irr	1.cpl.irr	2.obj	aum	1.pls
2.cpl	2.prf	gen	com	2.pot
adj	cond	3.cpl.irr	1.sg	encl
3.sg	3.pss.pl	spt	1.irr	2.enf
conj.adv	caus	con	chico	eh
comp	prf	dist mov	3.irr	det.dem
dcl	nom	2.icp.irr		

Tabla 3.5: Glosa

que son separadas por puntos. Por ejemplo, pl.exc es una combinación de las etiquetas ”plural” y ”exclusivo”.

El significado para cada etiqueta de glosa se muestra en la tabla 3.6. En esta tabla se omitieron las variaciones de etiquetas con personas gramaticales para compactarla. 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 una carga pronunciada en la distribución hacia pocas etiquetas como stem. Este fenómeno es propiciado, en parte, a que son escasos los textos digitales disponibles para el idioma otomí.

En las tablas 3.7 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 3.7 ya que representan un conjunto de 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 **?** estas lenguas son aquellas que tienen una cantidad limitada de recursos digitales a consecuencia de

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 3.6: Descripción de Glosa

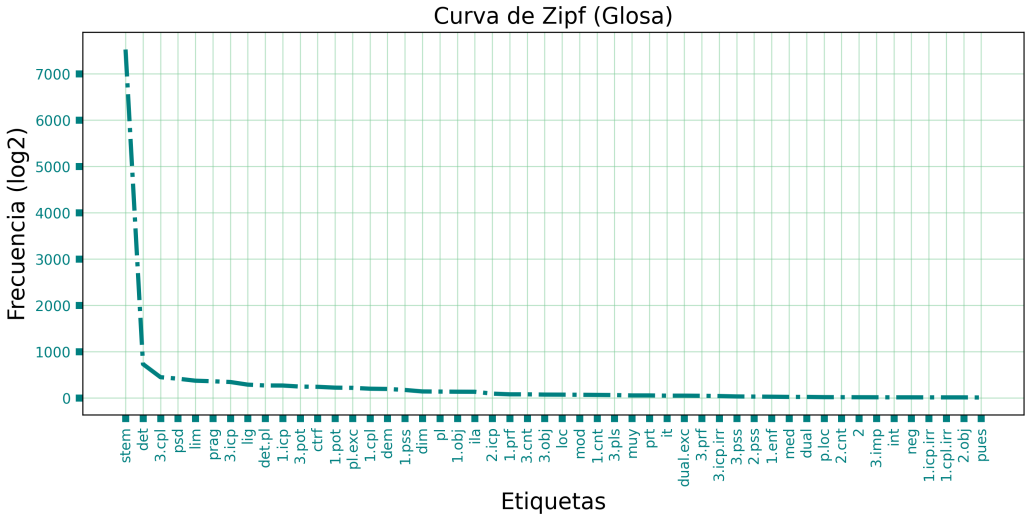

Figura 3.2: Distribución de glosa (primeras 50)

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.

Sin embargo, existen métodos que buscan solucionar el problema de los bajos recursos digitales. Los trabajos de Moeller y Hulden (2018) y **?** muestran como métodos de aprendizaje estructurado, en particular los *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 problema del *bias* que Lafferty et al. (2001) define como "el momento en el cual las transiciones que salen de un estado sólo compiten entre sí, y no con todas las demás transiciones".

3.2. Arquitectura

Para esta tesis proponemos una arquitectura de aprendizaje estructurado débilmente supervisado utilizando el método gráfico *Conditional Random Fields (CRF)* que permitirá la predicción de secuencias que describen las unidades morfológicas (glosa) dentro de una palabra en otomí. Ya que el resultado esperado es la generación de etiquetas *BIO* que, en principio, son

Etiqueta *POS*	**Tokens**
v	2596
obl	2447
det	975
cnj	837
dem	543
unkwn	419
n	273
neg	178
p.loc	81
prt	49

Glosa	**Tokens**
stem	7527
det	733
3.cpl	450
psd	418
lim	374
prag	362
3.icp	346
lig	289
det.pl	271
1.icp	270

Tabla 3.7: Tokens de etiquetas (Primeras 10)

secuenciales, un método que construye el modelo de aprendizaje basándose en gráficas 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 functions*, que definimos como X. Tales funciones contienen información sobre la estructura de la lengua y permiten que la máquina aprenda un modelo para etiquetar secuencias. Cabe señalar que X contiene las *feature functions* que son las entrada de los *CRFs*.

Debido al enfoque de aprendizaje automático 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 obtuvieron, entre otras medidas de rendimiento, el *accuarcy* del modelo. Las secuencias de etiquetas o salidas generadas por el modelo, dadas las observaciones de X, fueron definidas como Y.

En general, para el glosado se realiza manualmente. Esta tarea requiere del expertiz del 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 débilmente 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 functions*. Estas funciones describen características de la lengua considerando, detalladamente, el contexto, la posición y etiquetas gramaticales que brindan información útil para la fase de entrenamiento.

Puntualmente, se utiliza un modelo del tipo **Markov CRF de primer orden con *features* de estado y de transición**. Adicionalmente, fue utilizado el algoritmo de aprendizaje **Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)**. El algoritmo busca maximizar el logaritmo de probabilidad. Para mejorar el rendimiento son utilizados y modificados un conjunto hiperparámetros.

Estos hiperparámetros son los coeficientes de regularización L1 y L2 necesarios para evitar posible *overfitting*, 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 y el algoritmo de aprendizaje fueron tratados a detalle en la sección 2.1.

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 conjuntos 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 abordaran las etapas de generación de glosa y pruebas.

La arquitectura de aprendizaje semi-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 validas para el lenguaje `Python`.

La glosa esta 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 el ejemplo 3.2.1.

Ejemplo 3.2.1. [[['hi', 'stem'], 'neg'], [['tó', '1.prf'], ['tsogí', 'stem'], 'v']]

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 **?** 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	ɘ
Ortografía práctica	u̲	e̲	a̲	i̲	o̲
Convención para este trabajo	μ	ϵ	α	ι	

Tabla 3.8: 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, 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 de caracteres que por si solos no tiene sentido. Entonces, fue necesario modificar algunas de las vocales del otomí. Las equivalencias de tales modificaciones pueden observarse en la tabla 3.8

Entonces, la estructura de las listas, por renglón, tiene la estructura `[[[letras, glosa], [letras, glosa], ..., POS],...]`. Una vez que se

obtuvo el corpus de entrenamiento se le aplicó preprocesamiento. El preprocesamiento consistió en asociar, a cada letra de cada palabra, dos elementos; la etiqueta *POS* y una *Bio Label* correspondiente a esa letra.

Retomando el ejemplo 3.2.1 y después de aplicar el preprocesamiento el resultado sería el que se muestra en el ejemplo 3.2.2. Cabe señalar que las *Bio Labels* asignadas dependieron de la posición de la letra como se explico en la sección 2.1.

Ejemplo 3.2.2. [[['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']]]

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* del modelos etiquetando frases no vistas. Es importante destacar que estos dos conjuntos estuvieron completamente separados ya que mezclar el conjunto de entrenamiento y de pruebas pudo habernos llevado a resultados erroneos y sesgados.

Previo al entrenamiento se construyeron las *feature functions* con el conjunto de entrenamiento. En ese sentido, por cada letra etiquetada de las palabras en otomí se tuvo una *feature function* y por cada *feature function* se tiene una *Bio Label* asociada. La construcción de estas funciones será descrita a profundidad en la subsección 3.2.2.

Los *CRFs* recibieron como entrada las *feature functions*, representadas por un vector X y sus respectivas *Bio Labels*, representadas por un vector y, asociadas con cada *feature function* con concordancia uno a uno 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 y es importante para predecir la morfología. Para construir las *feature functions* 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 esta viendo en un momento determinado para las predicciones posteriores.

prevpostag y nxtpostag: 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 toma la etiqueta POS previa (Si existe) y nxtpostag toma la etiqueta POS siguiente (Si existe)

BOS, EOS, BOW y EOW: Indicar el inicio y fin de frase y palabras otorga la capacidad de ver que tipo de palabras se está viendo 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 dónde, 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

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

nombre de la *feature* que indica el tamaño de la ventana a partir de la letra que se esta viendo en ese momento y va desde el número uno, caso dónde 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.

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 primer letra de una palabra la que se observa no estarán presentes las características prevletter, prev2letters o EOW por mencionar algunas. Características como letterLowercase o bias siempre estuvieron presentes.

La construcción de las *feature functions* dependió del experimento en turno. Por ejemplo, en algunos casos fueron modificadas reduciéndolas 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

El termino *CRFs* se refiere a una familia de métodos de aprendizaje basados en gráficas. En concreto, para la generación de secuencias de etiquetas utilizamos *linear-chain CRF*. Este método requiere de un algoritmo de optimización. Como ya mencionamos previamente utilizamos el *L-BFGS*. 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 L1 + L2. 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 3.9 se muestran los valores que no se modificaron para los hiperparámetros.

Además de los hiperparámetros se configuraron diferentes entornos de experimentación dónde se construyeron *feature functions* con mas o menos

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

Tabla 3.9: Valores de los hiperparámetros

información. En la tabla 3.10 se listan las diferentes configuraciones implementadas en esta tesis.

Modelo	**L1**	**L2**	**Features**
linearCRF_l2_zero	0.1	0	Todas las disponibles
linearCRF_featured	0.1	$1 * 10^{-3}$	Todas las disponibles
linearCRF_l1_l2_zero	0	0	Todas las disponibles
POS_Less	0.1	$1 * 10^{-3}$	Sin etiquetas *POS*
linearCRF_l1_zero	0	$1 * 10^{-3}$	Todas las disponibles
POS_Less_l1_zero	0	$1 * 10^{-3}$	Sin etiquetas *POS*
POS_Less_l1_l2_zero	0	0	Sin etiquetas *POS*
HMMLike_l2_zero	0.1	0	Mínimas
HMMLike_regularization	0.1	$1 * 10^{-3}$	Mínimas
baseline_HMMLike_zero	0	0	Mínimas
HMMLike_l1_zero	0	$1 * 10^{-3}$	Mínimas

Tabla 3.10: Modelos de aprendizaje

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 functions* y la *Bio Labels* asociada a cada *feature function*. Utilizando el modelo de aprendizaje se etiquetó cada letra con base en la estructura de la lengua codificada en las *feature functions*. Las etiquetas generadas fueron comparadas con las reales y con ello se obtuvo la exactitud del modelo.

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*

functions que serán las entradas de los *CRFs*, continuamos con la fase de entrenamiento, creación del modelo de aprendizaje. Por último, se ejecuto 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 3.3.

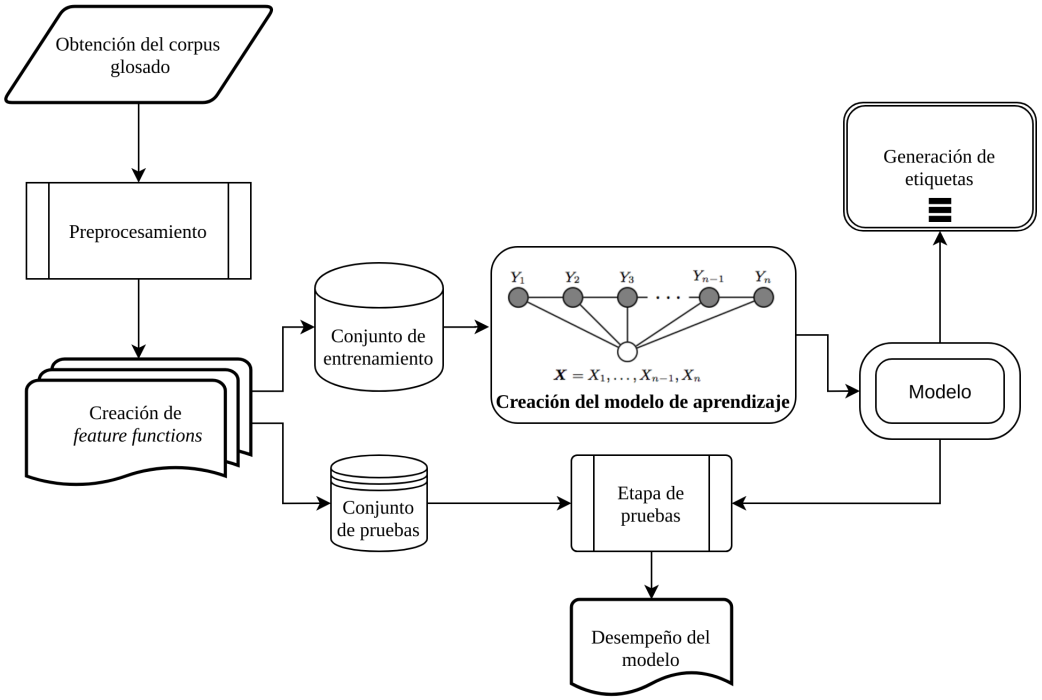

Figura 3.3: Arquitectura de aprendizaje

Hardware utilizado

En la fase de experimentación fue utilizado el paquete python-crfsuite que es un *binding* de la implementación de **?** para CRFs para lenguaje de programación *Python* en su versión *3.7*. La fase de experimentación corrió en una maquina con un procesador Intel i7-7700HQ @ 3.800GHz con 16 GB de memoria principal. En promedio una corrida de entrenamiento y evaluación *K-folds* con *k=10* tomó 52 minutos.

Capítulo 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 functions* y los parámetros de regularización *Elastic Net* L1 + L2. En concreto, 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, que se mencionaron en la sección 3.2.2.

Para comparar el desempeño entre los modelos construidos, presentes en la tabla 3.10, seleccionamos como base el entorno dónde se simula un *HMM* que corresponde con el modelo baseline_HMMLike_zero.

Para saber que tan buenas son las predicciones de un modelo generado es necesario validar su desempeño. La forma mas sencilla de realizar esta validación es partir 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. Está 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 modelos podría mostrar valores muy optimistas o muy pesimistas dependiendo solo de cómo se realiza la partición del corpus.

Un método de validación más adecuado para nuestros entornos de experimentación es el de *K-fold cross-validation*. Esta técnica está diseñada para medir el desempeño 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 obtiene el *accuracy* utilizando el subconjunto de pruebas antes mencionado. El *accuracy* es obtenido como se muestra en la ecuación 4.1.

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

dónde VP es Verdadero Positivo, VN es Verdadero Negativo, FP es Falso Positivo y FN es Falso Negativo.

El proceso se repite K veces hasta que cada subconjunto fue utilizado en la etapa de pruebas. Finalmente, se promedian los K *accuracy* obtenidos en cada prueba para tener el *accuracy* general del modelo que es el que reportamos. 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 $K = 10$.

Consideramos exitosa la predicción si se logra maximizar la correcta clasificación de las etiquetas de salida.

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

En la tabla 4.1 se muestran los modelos dónde se recuperaron todas las *features* disponibles. A este entorno de experimentación lo llamaremos *LinearCRF*.[1]

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

[1]Tiempos de entrenamiento (minutos): linearCRF_l2_zero=41.86, linearCRF_featured=42.91, linearCRF_l1_l2_zero=39.46, linearCRF_l1_zero=41.03

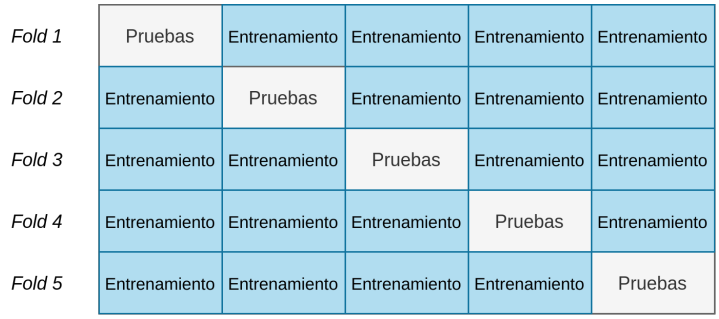

Figura 4.1: Representación gráfica de *K-fold cross-validation*

Modelo	Accuracy
linearCRF_l2_zero	**0.9516**
linearCRF_featured	0.9509
linearCRF_l1_l2_zero	0.9499
linearCRF_l1_zero	0.948

Tabla 4.1: Modelos con todas las *features* disponibles

experimentación lo llamaremos *POSLess*.[2]

Modelo	Accuracy
POS_Less	**0.9499**
POS_Less_l1_zero	0.9473
POS_Less_l1_l2_zero	0.9452

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

En la tabla 4.3 se aprecian los modelos generados en el entorno de experimentación con las *feature functions* reducidas al mínimo, simulando *HMMs*. Este entrono de experimentación será llamado *HMMLike*.[3]

[2]Tiempos de entrenamiento (minutos): POS_Less=37.91, POS_Less_l1_zero=40.25, POS_Less_l1_l2_zero=36.67

[3]Tiempos de entrenamiento (minutos): HMMLike_l2_zero=43.71, HMMLike_regularization=42.98, baseline_HMMLike_zero=40.44, HMMLike_l1_zero=43.9

Modelo	Accuracy
HMMLike_l2_zero	**0.8818**
HMMLike_regularization	0.8791
baseline_HMMLike_zero	0.8762
HMMLike_l1_zero	0.8745

Tabla 4.3: Modelos que simulan *HMMs*

4.2. Análisis de resultados

En principio, 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 desempeños de los *CRFs*. En la tabla 4.3 se pueden ver los modelos construidos en este entorno.

Es importante destacar que este entorno de mínima información simula el método de generación de etiquetas *HMM* pero no es, en sentido estricto, un *HMM*[4] ya que fueron utilizados los *CRFs* para su construcción.

Diego

referencia

El modelo con el *accuracy* más bajo se encuentra en este entorno de experimentación y es HMMLike_l1_zero con un 87.45 % de *accuracy*. Por otra parte, el modelo propuesto como base supera en al modelo HMMLike_l1_zero por apenas $1.7*10^{-3}$ unidades. Notamos que variando los hiperparametros L_1 y L_2 para este entorno de experimentación se obtuvo una mejora de apenas $5.6 * 10^{-3}$ respecto al modelo base.

Continuando con el segundo entorno de experimentación, dónde se construyó el *linear Chain CRF* ignorando las etiquetas *POS*, 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 con 94.99 %.

Por último, el entorno dónde se construyeron los *linear Chain CRFs* tomando todas las *features* disponibles, mencionadas en 3.2.2, fue en el que se obtuvo el mejor desempeño. Los modelos generados en este entorno obtuvieron entre 94 % y 95 % de *accuracy*. Una vez más, la variación de hiperparametros de regularización comprobó mejoras leves en los modelos. Destacamos

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

además que la mejora con respecto al entorno de experimentación *POSLess* es de apenas 1 % en los modelos con mejor desempeño del entorno *linearCRF*. En la tabla 4.4 se muestra la comparación entre el modelo base y los modelos con mejor *accuracy* en los otros entornos de experimentación.

Modelo	**Accuracy**
linearCRF_l2_zero	**0.9516**
POS_Less	0.9499
baseline_HMMLike_zero	0.8762

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 functions*, obtienen un desempeño mayor en comparación con los entornos dónde se disponía de información mínima.

Con base en la arquitectura propuesta observamos que el modelo **linearCRF_l2_zero** 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, POS_Less, HMMLike) (Ver tablas 4.1, 4.2, 4.3), se obtuvo una mejora en el *accuracy* poco representativa; por otra parte, el *accuracy* que muestran los modelos POS_Less y linearCRF es muy 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 *features functions*, es la que determinan, en gran medida, el desempeño de los *CRFs* para la correcta generación de *BIO Labels* (glosa) para el idioma otomí.

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 'ri' o 'yi' (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.

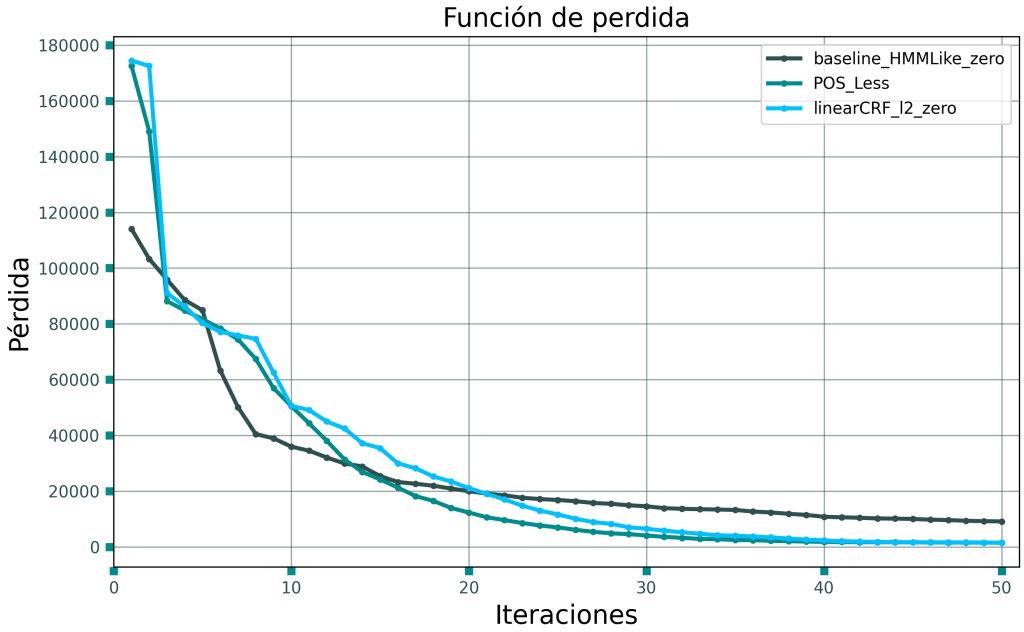

Figura 4.2: Función de perdida de los modelos con mejor desempeño

Mostramos una comparación de la función de perdida en la figura 4.2 de los modelos que se encuentran en la tabla 4.4. En la figura se encuentran las curvas para la iteración $K = 10$ de cada modelo.

En la figura 4.2 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 no es tan buena. Esto puede deberse a la cantidad de información codificada en la *feature functions* de inicio.

Por otro lado, las curvas de los modelos POS_Less y linearCRF_l2_zero 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 functions* el desempeño es mejor.

La información codificada en ambos modelos es casi la misma salvo por que en los modelos POS_Less se omitieron las etiquetas *POS*. Ergo, las curvas de los modelos POS_Less y linearCRF_l2_zero son similares.

En la tabla 4.5 se observa la precisión, el *recall* y el *f1-score* que dan información detallada de la eficiencia del modelo linearCRF_l2_zero en el etiquetado del último *fold*. Con base en dicha tabla hacemos las siguientes

observaciones.

Etiqueta	**Precisión**	**Recall**	**F1-score**	**Instancias**
prt	0.33	0.33	0.33	3
pl	0.71	0.45	0.56	11
2.cnt	0.75	1	0.86	3
lig	0.87	0.9	0.88	29
3.cnt	0.88	0.88	0.88	8
1.obj	0.91	0.83	0.87	12
ctrf	0.95	0.95	0.95	22
stem	0.96	0.96	0.96	771
prag	0.97	0.97	0.97	29
3.icp	0.97	0.9	0.94	40

Tabla 4.5: Métricas por etiqueta del modelo linearCRF_l2_zero

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. Además, el bajo *recall* en esta etiqueta puede deberse 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 (Raíz) o CTRF ('Contrafactual') al ser sistemáticas y muy frecuentes se ven beneficiadas por lo que tienen una *precisión* y un *recall* altos.

PRAG ('Pragmático') aparece en los verbos y siempre en la misma posición (al final de la palabra en la posicion del verbo) lo que hace que tambien 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 ('Tercera persona incompletivo'), 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 mas altas en desempeño.

Cabe destacar que las etiquetas STEM, PRAG Y 3.ICP, que son las etiquetas con mayor desempeño, no necesariamente comparte la frecuencia. Mientras que STEAM es la etiqueta con mayor frecuencia PRAG y 3.ICP muestran una frecuencia considerablemente mas baja. Aún asi, las tres etiquetas son

las que muestran la *precisión*, *recall* y *f1-score* más altos.

En suma, vemos que el modelo tiene dificultades para asignar las etiquetas a frases que no son comunes y que son dependientes del contexto. Contrario, las que tiene alta *precisión* y *recall* son sistemáticas aunque vemos que la frecuencia no es un factor determinante. Entonces, las variaciones morfológicas y las etiquetas con apariciones no sistémicas propician un bajo rendimiento en el etiquetado.

Diego

Mas bien que son independientes del contexto ¿no?

En la tabla 4.6 se muestran la *precisión*, 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	0.33	0.33	0.33	3
med	0.4	1	0.57	2
mod	0.43	0.21	0.29	14
dual.exc	0.5	1	0.67	1
muy	0.5	0.5	0.5	4
1.prf	0.57	0.8	0.67	5
2.cnt	0.6	1	0.75	3
3.prf	0.6	0.75	0.67	4
loc	0.62	0.83	0.71	6
3.pls	0.67	1	0.8	6
stem	0.83	0.81	0.82	810

Tabla 4.6: Métricas por etiqueta del modelo baseline_HMMLike_zero

Etiquetas como DUAL.EXC ('Dual Exclusivo') y MED ('Voz media') tienen un *recall* alto pero una *precisión* baja lo que sugiere que el modelo esta 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 *precisión* de las etiquetas STEM (raíz) es 13 % menor que la presente en el modelo linearCRF_l2_zero 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 mas 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 functions*) es mínima.

A continuación mostramos ejemplos de frases etiquetadas en los entornos de experimentación linearCRF_l2_zero y baseline_HMMLike_zero

Ejemplos de etiquedado del modelo linearCRF_l2_zero

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

En el etiquetado 1 vemos la aparición de etiquetas sistemáticas como STEM y 3.icp, por mencionar algunas, por lo que la frase es correctamente etiquetada.

(2) a. g-i-né
PSD-3.ICP-STEM
'*Quería'

b. gi-né
2.POT-STEM
'Vas a querer'

En el ejemplo 2a 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 2b

Curiosamente la letra 'g' por si misma no coincide con ningún patrón del tiempo pasado pero la letra 'i' siguiente si corresponde a un 3.ICP por lo que el modelo esta 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 3a.

Las características de las clases abiertas las hacen difíciles de ver por completo en el entrenamiento contrario a las clases cerradas que con clases finitas. Además, notamos que el modelo linearCRF_zero_l2 se equivoca cuando no hay suficiente contexto, por ejemplo, frases cortas de una sola palabra.

(3) a. bi-ndμzú
3.CPL-STEM
(No traducible con esta segmentación)

b. bi-ndμ-zú
3.icp-muy-stem
'Se asustó'

Ejemplo de frase etiquetada por el modelo baseline_HMMLike_zero

(4) bi-μn-gí yι mbí 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 4 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 observo, los ejemplos largos son mejores para estos modelos de baja información.

(5) 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 5a 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 3a. 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 esta sobre segmentando y encontrando morfemas que no son correctos. Confunde secuencias con patrones frecuentes y no es raro por que a este modelo se le otorgó mínima información por lo que no esta tomando tanto en cuenta el contexto de las oraciones. Con mas 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 6.

(6) i b i t h ó p a
I-IT B-3.CPL I-3.CPL **B-ILA** **I-ILA** **I-ILA** B-STEM I-STEM
n ι r ι c h á i
B-STEM **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'`. En concreto la salida con las *BIO Labels* bien formadas sería la que se ve en el ejemplo 7.

(7) i b i t h ó p a
B-STEM B-3.CPL I-3.CPL B-STEM I-STEM I-STEM B-STEM I-STEM
n ι r ι c h á i
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 7 se tendría la glosa que se muestra en el ejemplo 8.

(8) i bi-thó pa nι rι chái
STEM 3.CPL-STEM STEM DET DET STEM
'TODO'

Capítulo 5

Conclusiones

La información lingüística codificada en las fuature functions es muy importante para mejorar el desempeño del etiquetador

Sin embargo, las etiquetas pos no son restrictivas

Ximena: el hecho de no necesitar las pos es bueno para las lenguas minorizadas pues no siempre hay etiquetadores automáticos para estas lenguas

Cuando se le quita información al modelo en las *feature functions* la frecuencia de las instancias tiene mayor pero. Por lo tanto, para bajos recursos, donde las frecuencias de las instancias son bajas, es necesario dar un contexto más amplio y agregar información lingüística mas detallada.

5.1. Trabajo Futuro

Poner en un ambiente de transferencia y probar si los modelos que entrene pueden aplicarse a otras variantes de otomí con un zero show learning