Universidad Nacional Autónoma de México

facultad de estudios superiores acatlán

Traductor automático de textos de la lengua *Ayuuk* variante del municipio de San Juan Güichicovi Oaxaca, al español

T E S I S

que para obtener el título de:

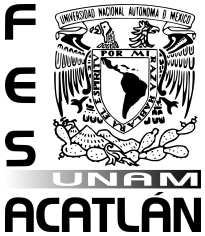
Licenciado en Matemáticas Aplicadas y Computación

presenta:

Delfino Zacarías Márquez

Tutor:

Dr. Ivan Vladimir Meza Ruiz



Santa Cruz Acatlán, Naucalpan, Estado de México, 2022

Agradecimientos

Llegar a este momento tan importante de mi vida ha sido un camino muy difícil, pero no imposible, detrás de este logro hay personas increíbles que en diferentes puntos de mi vida se han cruzado en mi camino y me han impulsado a seguir adelante, por tal motivo quiero darles mis sinceros agradecimientos.

En primer lugar, gracias Dios, por prestarme la vida para poder sentir lo maravilloso que es vivir, por todas tus bendiciones y cuidarme en todo momento, por los padres que me diste y por las personas maravillosas que has puesto en mi camino.

A mis padres, Margarita y Raúl, gracias por ser los mejores padres del mundo, por todo el amor que me tienen. Gracias mamá, a pesar de que nunca pisaste una escuela siempre estuviste para apoyarme en todo pese a las carencias, gracias por tus desvelos, por trabajar arduamente haciendo totopos para comprar mi lápiz y cuaderno. Gracias papá, por todos tus consejos, por enseñarme a trabajar en el campo, por tus desvelos para salir a sembrar maíz nunca nos ha faltado tortilla en la casa, aunque solo estudiaste hasta el primer grado de primaria pudiste enseñarme a leer. Gracias a ambos por los principios y valores que me han inculcado, por apoyarme en las decisiones que he tomado, por todo el tiempo que me dieron para que yo me dedicara a estudiar. Ustedes son las personas que más admiro y son el motor de mi vida. Quiero decirles que este logro es de ustedes, los amo con todo mi corazón.

A mis abuelitos Delfino † y Tomasa †. Gracias por consentirme, por platicarme historias mientras nos mecíamos en la hamaca.

A mi asesor Dr. Iván Vladimir Meza Ruiz. A pesar de que no me conocía, agradezco su disposición de recibirme en el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) y aceptar dirigir mi tesis, gracias por todo el apoyo que me brindó, por su paciencia para explicarme lo que no

comprendía durante el desarrollo de este trabajo y por enriquecer mi interés por el procesamiento de lenguaje natural. Es un honor ser su alumno.

A mis sinodales: Mtra. Georgina Eslava García, Mtro. Rubén Romero Ruiz, Dr. Iván Vladimir Meza, Dr. Jorge Vasconcelos Santillan y Dr. Eduardo Eloy Loza Pacheco. Muchas gracias a cada uno de ustedes por tomarse el tiempo para revisar y retroalimentar este trabajo.

A los traductores de la Unión Nacional de Traductores Indígenas (UNTI): Martha Ramírez, Jonathan Santiago y Rosalba Antonio, muchas gracias por su disposición. En especial agradezco al Tr. Victoriano Santiago Cayetano, por su confianza desde el primer momento en el que le platiqué sobre este trabajo, por la gran cantidad de materiales traducidos que me proporcionó para poder realizar los experimentos de esta tesis. Gracias por enriquecer mi interés por la lingüística con las clases de lecto-escritura del Ayuuk.

Al maestro Albino Pedro Juan. Gracias por su confianza, por los materiales proporcionados para realizar los experimentos de este trabajo, por las pláticas sobre historia y lingüística.

A la maestra Mercedes Azcona Figueroa y al profesor Melchor Escobar Ocaña. Gracias por su tiempo para atenderme y por compartirme material para realizar los experimentos de este trabajo, y por esas pláticas sobre la escritura del Ayuuk.

A mi hermano Cuauhtemoc. Por estar siempre al pendiente de mi y el apoyo que me has brindado.

A mi primo Leonel. Aunque eres mi primo te considero como mi hermano mayor y tengo mucho que agradecerte, gracias por darme un tour en Ciudad Universitaria y motivarme a estudiar Matemáticas Aplicadas en la UNAM, por sacarme de muchos apuros, por esas pláticas mientras salíamos a correr, por invitarme al IMATE de Cuernavaca y por muchos buenos momentos que hemos pasado juntos. Gracias a ti, por tus ánimos y consejos, he acabado la carrera. Siempre estaré muy agradecido contigo.

A mi prima Dalid. Gracias por toda esa paciencia para escucharme y darme consejos, eres como mi hermana mayor. Gracias por animarme a estudiar matemáticas aplicadas, por acompañarme el día de la entrega de mis documen-

tos en la UNAM, por estar siempre al pendiente de mi y por todo ese apoyo incondicional.

A mis primos: Beto, Erika, Chuy, Olegario, Miguel y a mi tía Toñita. Por todo el apoyo que me han brindado, las motivaciones y todos sus consejos. Nada de esto sería posible sin el apoyo de ustedes, siempre estaré muy agradecido.

A la tía Elisa. Por ser siempre muy atenta, por todo el apoyo que me ha brindado. Gracias por darme espacio en su casa cuando fui a hacer el examen de admisión en la CDMX.

A mis tíos: Pancho, Zeferino y tía Consuelo †, Edelmira y Héctor †. Gracias a cada uno de ustedes por consentirme cuando los iba a visitar. Siempre estaré muy agradecido con ustedes.

A mis primos: Héctor, Josué, Aarón, Raciel, Noé, Norma. Por hacerme sentir como en casa cuando los visitaba.

A mis primos Israel e Isaac, tía Raquel y en especial a mis tíos Enrique (Meno) y Elodia, por el hospedaje que me ofrecieron cuando estaba en mis primeras semanas de la universidad mientras buscaba un lugar para rentar.

A Lalo Jacobo. Por las comidas improvisadas, por todas esas pláticas interesantes, por sacarme de varios apuros, por todos tus consejos, por los bailongos en Ciencias y el buen mezcal. Gracias por ser mi amigo.

A Mayra. Por tener esa paciencia para escucharme, por todo el apoyo que me has brindado, por todas esas aventuras cuando salíamos a correr, por los bailongos y las buenas anécdotas.

A Itzel Chavira, David Chavira y su mamá Rocío. Gracias por el alojamiento que me ofrecieron. Siempre estaré muy agradecido.

A Ángel Francisco. Por ser mi amigo y competencia intelectual en la prepa, nos volvimos a encontrar en la UNAM, gracias por sacarme de varios apuros, por las invitaciones a los bailongos istmeños, por la comida y por tu apoyo incondicional.

A Luis Ángel. Por ser mi amigo desde la infancia, por esas largas pláticas por teléfono y recordar viejos tiempos, gracias por asistir a mi foto de

generación.

A Luis Miguel Bailey. Gracias amigo porque siempre me has sacado de varios apuros apoyandome desde la distancia.

A Fany Fuentevilla. Porque siempre estuviste presente para apoyarme y animarme, por esas palabras de aliento, por asistir a mi foto de generación.

A Kary Granados. Gracias por cruzarte en mi camino, por confiar en mi y ofrecerme un lugar para rentar con ustedes, por todos los ánimos y ese apoyo incondicional, por todos los buenos momentos. Siempre estaré muy agradecido contigo.

A Miry Cervantes. Por escucharme siempre y darme muchos ánimos mientras hacíamos tarea, por todos los buenos momentos.

A Diana Laura Hernández. Por todas esas pláticas y risas, por sacarme de varios apuros y por todos los buenos momentos.

A don Héctor. Por todos los consejos y las buenas historias.

A doña Nora Levario. Gracias por todo el apoyo que me brindó, por las largas platicas y sus consejos que siempre tendré presente, por estar siempre al pendiente de mi. Siempre estaré muy agradecido con usted.

A Daniel (Mandy). Por esas largas pláticas sobre el software libre, gracias por instalarle Xubuntu a mi netbook cuando estaba en mis inicios en el mundo del Linux.

A Sergio Estrella. Por estar siempre disponible para ayudarme, por arreglar mi laptop en varias ocasiones, por esas pláticas sobre Linux.

A Juan Carlos. Gracias por apoyarme en todo momento, por regalarle comida a este foráneo, por enseñarme el lado divertido de la vida, por ser mi amigo.

A Jordan Arenas. Por compartirme tus conocimientos sobre matemáticas, por escuchar mis ideas, por las buenas anécdotas, gracias por invitarme a la prefiesta de graduación.

Luis Alfredo. Gracias por animarme a seguir cuando estaba desanimado en

las primeras semanas de la carrera. Siempre estaré agradecido.

A Matilde Manzano. Gracias por esa paciencia para escucharme, por tu disposición en apoyarme siempre, por todos los momentos divertidos.

A Julio López. Por ser mi amigo, por tu paciencia y disposición para explicarme lo que no entendía. Siempre estaré muy agradecido.

A Fernando Santana. Gracias por enseñarme inducción matemática, por los consejos y el apoyo en los primeros meses de la carrera.

A Ana Amezcua y Ana Trejo. Chicas, muchas gracias por darme ánimos y por acompañarme en mi foto de generación.

A Daisy, Sheila, Ulises y Jonathan. Gracias a cada uno de ustedes por esas aventuras de foráneos, por sacarme de varios apuros, por todos los buenos momentos que pasamos.

A Evy. Por escucharme siempre, por esas risas que nunca han faltado y por los buenos momentos.

A Mario Uunk Zacarías. Gracias por ayudarme a contactar a los profesores que se dedican a la traducción en el municipio de Güichicovi.

A mis profesores de la telesecundaria: Erika, Jorge y especialmente al profesor Roque Matus, por la manera de enseñar matemáticas y animarme a seguir en este camino.

A mis profesores del IEBO plantel 196: Ing. Cleyver, Ing. Paulino, Lic. José Antonio Mayo y en especial al Mtro. Mario Zacarías, por sus consejos y por apoyarme durante mi trabajo de campo para la recolección de datos.

A mis profesores de la FES Acatlán: Mtra. Araceli Álvarez, Mtro. Ricardo Esquerro, Lic. Ernesto Baltazar, Mtro. René Martínez, Mtro. Carlos Ortega, Mtra. Silvia Larraza, Lic. Pablo Ávalos, Mtra. Georgina Eslava, Dra. Mayra Olguín, Lic. Christian Elizondo, Mtra. Liliana Flores, Dra. Beatriz Trueba, Act. Héctor Morales, Mtra. Jeanett López, Dr. Jorge Zamudio, Lic. Sergio Rosales, Lic. Samuel Ornelas, Dr. Sergio Chapa, Dr. Jaime Muñóz, Mtra. Guadalupe Rodríguez, Lic. Virginia Haro, Mtro. Marco Pichardo, Mtro. Javier Rosas, Lic. Óscar Caballero, Lic. Fernando Trejo. Gracias a cada uno de ustedes por

las enseñanzas que me brindaron, por tomarse el tiempo de responder mis dudas. En especial al Dr. Jorge Vasconcelos, por insistirme en seguir con el trabajo investigación que inicié en IMAC y que finalmente se convirtió en esta tesis; y al Dr. Eloy Loza, por su gran disposición para atender mis dudas, por recomendarme en la escuela de verano "Días de Combinatoria".

A los profesores de la Coordinación de Matemáticas Aplicadas y Computación (MAC) en especial al Lic. Christian Delgado, muchas gracias por toda su disposición y su apoyo brindado para que todo esto fuera posible; a la Lic. Janeth Flores, muchas gracias por toda su disposición y las facilidades que me brindó para realizar mis trámites a distancia. A los profesores el Mtro. Mauricio Rico y al Lic. Francisco López.

A mis amigos y compañeros de la carrera de MAC: Daniel Zermeño, Gerardo Padilla, Valeria Coahuila, Miguel Coahuila, Miguel Ríos, Vladi, Equihua, Cristian Martínez, Daniel Licona, Gerry, Armando López, Xotla, Brenda Berenice, Majo Martínez, Diego Colín, Marco Lugo, Hugo Iván, David Plácido, Susana, José Ponce, Alex Jergas, Paco Molina, Jorge Ramos, Edgar Agustín, Victor Humberto, Jesús, Diego Alfaro, Isa Luna, Jocelyn Díaz, Karen Cerón, Brenda Mayoral, Mariana Paola, Astrid, Alejandro Lázaro, Jessica Cecilia, Rubí Arenas, Noemí Pesquera, Ángel Gabriel, Lily Hernández y a los que me faltó mencionar. Es difícil escribir todos buenos momentos que pasé con cada uno de ustedes, nunca acabaría. Gracias a cada uno de ustedes por enseñarme el lado divertido de la vida, por compartirme sus experiencias y conocimientos. Es un honor haberlos conocido.

Al Sistema de Becas para Estudiantes Indígenas (SBEI) del Programa Universitario de Estudios de la Diversidad y la Interculturalidad (PUIC) de la UNAM, por el apoyo económico que recibí en la licenciatura y un año como tesista, gracias a esta beca muchos de nosotros (los estudiantes indígenas que venimos de comunidades de escasos recursos) podemos terminar una carrera universitaria en la UNAM. Gracias por promover la convivencia entre becarios, ha sido una de mis mejores experiencias como universitario. Asimismo, agradezco a los tutores del programa: Miguel Ibarra, Wil, Darío, Ofelia y en especial a mi tutora Karina Pérez, por ser la mejor tutora del mundo, gracias por escucharme cuando más lo necesité, por darme consejos para tomar decisiones

difíciles y por todo el apoyo incondicional. Siempre estaré muy agradecido.

A mis amigos del PUIC: Betsy, Marinely, Gilberto, Rolando, Elvira, Jocelyn, Frida, Claudia, Zoé, Palemón, Omar, Mary Zurita, Rocío Mestas, David Mendoza, Carlos Martínez, Keila, Kupij, Viviana Sixto, Ives Salgado y a los que me faltó mencionar. Con cada uno de ustedes pasé muy buenos momentos, aunque algunos ya no los he visto, pero siempre los recuerdo, es un privilegio haberlos conocido.

Al personal del Departamento de Control de Procesos de Estadísticas Sociodemográficas del INEGI Dirección Regional Centro: Lic. Ana Bertha, Adrián, Lalo, Ara, Violeta, Jaime, Azu, Paty, Nereo y a los que me faltó mencionar. Gracias por integrarme al equipo para realizar mi servicio social.

A los organizadores de "Días de Combinatoria 2019 Popayan, Colombia": Dra. Carolina Benedetti, Dr. Rafael Gonzáles, Dr. Jhon Bravo, Dr. Nantel Bergeron, Dra. Lucía Medrano, Dra. Laura Escobar y Dr. Alejandro Morales, por aceptarme en la escuela de verano. Fue una de las mejores experiencias de mi vida.

A mis compañeros de la escuela de verano "Días de Combinatoria": Jhan Carlo, Alexis, Daniela, Nicolás, Jessica, Keila, Harold, Jhonatan, Emerson, Sergio, Alondra, Daniel y a los que me faltó mencionar, gracias chicos, por mostrarme un poco de su cultura, por enseñarme cómo se baila salsa en Colombia y por compartirme un poco de su conocimiento sobre combinatoria.

Al CONACYT por los recursos proporcionados a través de la Plataforma de Aprendizaje Profundo del Laboratorio de Supercomputación del Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) para Tecnologías del Lenguaje.

Al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS), por abrirme las puertas y el apoyo económico mientras escribía esta tesis con el proyecto "Traducción automática para lenguas indígenas de México" PAPIIT-IA104420, UNAM.

Gracias a los que han luchado para que la educación en la Univesidad Nacional Autónoma de México siga siendo gratuita y accesible para todos.

Nada de lo que he escrito hubiera sido posible sin la existencia de mi alma mater, la Univesidad Nacional Autónoma de México y la Facultad de Estudios Superiores Acatlán. Agradezco la formación personal y profesional, todas las facilidades brindadas, las becas, las actividades deportivas y culturales, todas las amistades que me diste, las mejores clases con los mejores profesores, por las mejores experiencias de mi vida. Estudiar Matemáticas Aplicadas y Computación en la UNAM es lo mejor que me ha pasado en la vida.

!!MUCHAS GRACIAS POR TODO UNAM!!

Resumen

Este trabajo presenta el primer sistema de traducción automática neuronal para la lengua *Ayuuk*. En los experimentos se probó la traducción de *Ayuuk* al español y de *español* a *Ayuuk*. La lengua *Ayuuk* es hablada en el estado de Oaxaca en México por los *Ayuukjya'ay* (en español comúnmente referidos como *Mixes*). Se recolectaron diferentes fuentes escritas para crear un corpus paralelo (esencial para hacer traducción automática), de más de $6,500$ frases que se considera como de escasos recursos. Para algunas de estas fuentes se utilizó una herramienta para alineación automática de frases. El sistema propuesto se basa en la arquitectura neuronal Transformer (actual estado del arte), y utiliza la tokenización a nivel de subpalabras como entrada. En los resultados se muestra el desempeño actual dado los recursos que se han recolectado para la variante del municipio de San Juan Güichicovi, los resultados de los experimentos son prometedores, hasta 7 en BLEU y perplejidad menor a 50. Cabe destacar que el núcleo de esta propuesta sigue los pasos del proyecto *Masakhane* para lenguas africanas.

Índice general

Índice de figuras

Índice de cuadros

Introducción

El lenguaje natural es el medio que los seres humanos hemos desarrollado durante miles de años para poder comunicarnos entre nosotros, ya sea de manera oral, escrita o de otras formas (Ejemplo; gestos). Es un símbolo de identidad de cada comunidad que comparte el mismo código lingüístico.

El Procesamiento de Lenguaje Natural (PLN) es una intersección entre la ciencia de la computación y la lingüística (Gelbukh, 2007), encargada de estudiar y construir sistemas para la interacción entre las computadoras y humanos mediante el uso del lenguaje natural, como son: reconocimiento de voz, recuperación de información, análisis de sentimientos, traducción automática, por mencionar algunos. Siendo la Traducción Automática (TA) una de las primera aplicaciones relacionadas con el lenguaje natural (W. J. Hutchins y Somers, 1992; Schwartz, 2018). Sin embargo, las investigaciones en TA han sido trabajados en lenguas con mayor número de hablantes en el mundo y que han estado inmersos en la competencia tecnológica, algunas de ellas son: inglés, español, alemán, japonés, chino, ruso, solo por mencionar algunos, por el contrario, las lenguas indígenas han recibido poca atención en PLN debido a factores como escasos recursos en documentación lingüística, problemas de escritura, falta de financiamiento, entre otras causas; como es el caso de la lengua *Ayuuk*.

En los últimos años se han incrementado los esfuerzos para preservar y promover la creación de herramientas de PLN para las lenguas indígenas de las Américas, en particular abordando los desafíos que este esfuerzo conlleva (Mager, Gutierrez-Vasques, et al., 2018). La traducción automática de lenguas indígenas es una de las metas a perseguir, ya que a largo plazo puede ofrecer beneficios a la comunidad hablante de dichas lenguas. Asimismo, con esta herramienta se brindará acceso al conocimiento a alguna lengua; fortalecerá la escritura y la tradición oral; así como otorgar el fácil acceso a diversos servicios a los hablantes como la asistencia legal, médica, financiera, entre otros. La TA

de textos es uno de los primeros pasos.

De acuerdo con el INALI (2009), en México hay 68 lenguas indígenas clasificados en 11 familias lingüísticas sin considerar las variantes que tiene cada una. El estado de Oaxaca es el estado que cuenta con el mayor número de lenguas indígenas en el país; con 16 distribuidas en sus ocho regiones. Donde la lengua *Ayuuk* (Mixe) es la cuarta lengua con más hablantes a nivel estado estado y la decimosexta a nivel nacional, de acuerdo con cifras del último censo del INEGI (2020). El municipio de San Juan Güichicovi perteneciente al estado de Oaxaca, se encuentra dentro de los municipios con mayor número hablantes de la lengua *Ayuuk*. Sin embargo, a pesar de tener una población donde la mayoría habla su lengua materna, existen pocas documentaciones lingüísticas, dado que el principal medio de transmisión de la lengua ha sido de manera oral; dejando a un lado la escritura. Actualmente hay pocas publicaciones de la escritura del *Ayuuk* de la variante del municipio de San Juan Güichicovi (o variante de Güichicovi), las principales son por parte de organizaciones religiosas y particulares que han realizado estudios en la región, por lo que hay pocos recursos digitalizados de la lengua. Tampoco hay antecedentes de creación de algún corpus bilingüe de frases (Español-*Ayuuk*) y mucho menos un sistema que traduzca textos de manera automática de la lengua *Ayuuk* variante de Güichicovi al español y viceversa.

Los problemas con la lengua *Ayuuk* al igual que otras lenguas indígenas, son tan complejos que van más allá de la disponibilidad de datos, éstos tienen un trasfondo político y social. Lo anterior se refleja en la poca atención y difusión que se le da a la lengua, llegando a provocar poco interés en seguir conservando la lengua materna principalmente en los niños y jóvenes, tanto del municipio de San Juan Güichicovi como en el resto de la región mixe.

A pesar de que el artículo 5 de la Ley General de Derechos Lingüísticos de los Pueblos Indígenas[1] menciona que:

"*El Estado a través de sus tres órdenes de gobierno, -Federación, Entidades Federativas y municipios-, en los ámbitos de sus respectivas competencias, reconocerá, protegerá y promoverá la preservación, desarrollo y uso de las lenguas indígenas nacionales.*", existen pocos trabajos relacionados a un traductor

[1] https://www.inali.gob.mx/pdf/ley-GDLPI.pdf

automático de textos de lenguas indígenas usando técnicas de PLN.

Hasta el momento los trabajos de traducción realizados con las lenguas indígenas mexicanas, son los siguientes: Traductor híbrido wixarika-español con escasos recursos bilingües (Mager Hois, 2017); Traductor Automático Español-Purépecha Mediante OpenNMT (Soriano García, 2018); Traductor automático neuronal (NMT) del Maya Yucateco y Otomí de Querétero, Sistema Comercial Microsoft Translator; y alineación del náhuatl con el español mediante representaciones morfológicas (Gutierrez-Vasques, 2015).

Actualmente, el estado del arte para los modelos de Traducción Automática es la Traducción Automática Neuronal (NMT, por sus siglas en inglés) basada en arquitecturas Transformer (Vaswani et al., 2017). Con la llegada de esta nueva arquitectura los modelos de TA basados en *seq2seq* fueron superados y desplazados por Transformers, llegando a dominar en el campo del Procesamiento de Lenguaje Natural. Los modelos de traducción del proyecto *Masakhane* de lenguas africanas llegaron a obtener buenos resultados basadas en esta arquitectura.

Este trabajo busca aportar en el campo de la TA de lenguas indígenas, un traductor automático para una variante específica de la cuarta lengua más hablada en el estado de Oaxaca; un traductor automático de textos del español a la lengua *Ayuuk* variante de Güichicovi y viceversa. Dicho traductor busca adecuadar técnicas de TA del estado del arte, mismas que han dado resultados exitosos en la traducción de algunas lenguas nativas del mundo, como es el caso del proyecto *Masakhane*[2]. Aunado a lo anterior, el éxito de este proyecto se sustenta en la alta consideración de los recursos disponibles para el entrenamiento del sistema.

Hasta donde se sabe, no hay antecedente de una construcción de un sistema de traducción para la lengua *Ayuuk*, aunque existe un recurso para otra variante[3] en el corpus JW300[4] (Agić y Vulić, 2019).

[2]El proyecto *Masakhane* (https://github.com/masakhane-io/masakhane-mt); es un proyecto de traducción de lenguas africanas de código abierto. Sus modelos de traducción son entrenados con un corpus paralelo de JW300 [ver https://opus.nlpl.eu/JW300-v1.php] que contiene varias lenguas indígenas del mundo y pueden ser usados libremente. Este corpus contiene publicaciones de los Testigos de Jehová que no necesariamente son textos religiosos.

[3]*Ayuuk* de la región de Coatlán (ISO-639-3 *mco)*

[4]https://opus.nlpl.eu/JW300-v1.php

Esta tesis fue basada en los múltiples esfuerzos previos que se han realizado en el campo, tanto para lenguas que cuentan con grandes recursos, como para las que no. La idea principal de este trabajo siguió los pasos del proyecto *Masakhane* de lenguas africanas (Nekoto et al., 2020), donde en sus modelos de traducción utilizan arquitecturas del estado del arte basado en redes neuronales. De igual manera, para la alineación automática de los recursos recolectados se utilizó la herramienta de alineación YASA[5] (Lamraoui y Langlais, 2013); la tokenización implementada fue por subpalabras y para ello se usó la biblioteca *subword-nmt*[6] (Sennrich et al., 2016); y el entrenamiento de los modelos de traducción se realizó utilizando JoeyNMT[7] (Kreutzer et al., 2019). Con las herramientas mencionadas se desarrolló el código base para este trabajo. El repositorio se puede consultar en línea[8], así como la parte del corpus creado que está disponible con licencia libre.

Como es sabido, el PLN es una de las disciplinas de la inteligencia artificial (IA) que en las últimas décadas ha tenido un avance muy significativo. Empresas como Facebook, Google, Microsoft, entre otros, han invertido fuertemente en investigaciones en PLN, sobre todo en el campo de la TA. Pese a ello, estos esfuerzos sólo han beneficiado a lenguas que cuentan con grandes recursos en datos, en cambio, lenguas de escasos recursos como las lenguas nativas no han sido tomadas en cuenta en las investigaciones. En *Challenges of language technologies for the indigenous languages of the Americas* (Mager, Gutierrez-Vasques, et al., 2018) los autores hacen un compendio de los avances y retos que hay en las lenguas nativas de América. En cuanto a traducción automática se refiere, se visualiza que en México se ha trabajado muy poco el tema y únicamente en lenguas como el Wixarika (huichol), Purépecha, Náhuatl, Maya Yucateco y Otomí de Querétaro, se han empleado modelos de traducción basadas en reglas, traducción estadística y traducción neuronal con Redes Neuronales Recurrentes (RNN's). También se ha expuesto que el estado de Oaxaca es el estado que cuenta con el mayor número de lenguas indígenas en México, y a pesar de ello, no existe trabajo de TA en alguna de esas lenguas.

Los modelos de *aprendizaje profundo* para TA, están diseñados para ser

[5] https://github.com/anoidgit/yasa
[6] https://github.com/rsennrich/subword-nmt
[7] https://github.com/joeynmt/joeynmt
[8] https://github.com/DelfinoAyuuk/corpora_ayuuk-spanish_nmt

generalizados a otras lenguas, lo que abre la posibilidad de ser usados en una lengua indígena como el *Ayuuk*, específicamente con la variante del municipio de San Juan Güichicovi. Por lo que, los objetivos de este trabajo fueron recolectar y digitalizar textos; alinear frases del *Ayuuk* de Güichicovi con su correspondiente al español; normalizar los textos de acuerdo a las características de la escritura de la lengua; crear los corpus de entrenamiento, desarrollo y prueba; y generar una herramienta que traduzca de manera automática, textos de *Ayuuk* al español, y viceversa.

Con los resultados de este trabajo se busca generar un modelo base que servirá para investigaciones futuras. Las traducciones entre una lengua aglutinante como el *Ayuuk* y una fusionante como el español, permite conocer el rendimiento de las técnicas empleadas en este trabajo. Asimismo, propicia el espacio de trabajo necesario para la creación del primer sistema de traducción automática de una lengua indígena de la familia mixe-zoqueana. El trabajo de campo para la recolección de textos permitió conocer los datos disponibles de la lengua *Ayuuk* de la variante de Güichicovi, es decir, si son o no de libre acceso como los datos utilizados en otras lenguas indígenas. Además, conocer el estado del sistema de escritura de la variante explorada en este trabajo, permite valorar la viabilidad de crear un sistema más robusto (por ejemplo: segmentaciones morfológicas) en proyectos futuros. Por último, ante la problemática de pérdida y olvido de la lengua en cuestión, por parte de las nuevas generaciones; este trabajo desde el ámbito del PLN busca contribuir a la toma de conciencia e invitar a más personas a sumarse al proyecto de revitalización de la lengua *Ayuuk*, para de esta manera prolongar y garantizar su transmisión de generación en generación.

La hipótesis de este trabajo sostiene la afirmación de que es posible, a partir de la adecuación de los sistemas de traducción automáticos del estado del arte, basados en redes neuronales; crear traducciones de la lengua *Ayuuk* (variante del municipio de San Juan Güichicovi) al español, y viceversa.

Asimismo, como se trata del primer trabajo de traducción automática para la lengua *Ayuuk*, las principales preguntas a responder son:

1. ¿Es posible construir un traductor automático del *Ayuuk* variante del municipio de San Juan Güichicovi al español utilizando métodos de traducción automática del estado del arte?

2. ¿El modelo generado solo servirá para la variante del municipio de San Juan Güichicovi o también podrá incorporar otras variantes?

3. ¿Cuál será la calidad de las traducciones producidas por los modelos de traducción automática del estado del arte?

4. ¿Si los modelos del estado del arte en traducción automática no son adecuados para la lengua *Ayuuk* se pueden modificar para lograr un desempeño adecuado?

Por otro lado, el objetivo principal es desarrollar un traductor automático que convierta textos cortos del *Ayuuk*, variante del municipio de San Juan Güichicovi, al español y viceversa. En el trabajo se emplean herramientas de Procesamiento de Lenguaje Natural basadas en el estado del arte del campo, como redes neuronales y librerías especializadas. Para lograr lo anterior, se deben de cumplir los siguientes objetivos específicos:

- Recolectar documentos que hayan sido traducidos del español al *Ayuuk* de la variante que se habla en el municipio de San Juan Güichicovi.
- Alinear frases con un alineador automático.
- Construir corpus con frases alineadas entre el *Ayuuk* y español.
- Crear corpus de entrenamiento y pruebas.
- Crear tokenizador y normalizador que ayudará a la traducción.
- Entrenar el traductor.

Finalmente, este trabajo está compuesto por cuatro capítulos. El capítulo 1 aborda el estado actual de la lengua *Ayuuk*, un poco de la historia de la TA, los conceptos y técnicas de la TA que en su momento fueron el estado del arte, entre ellas la traducción basada en reglas (RBMT), traducción basada en ejemplos (EBMT) y traducción estadística (SMT). El capítulo 2 describe las técnicas que se utilizan en la Traducción Automática Neuronal (NMT), el estado del arte de la TA, tipos de alineadores de textos, métricas de evaluación de resultados, así como los datos disponibles que se tienen actualmente para

la lengua *Ayuuk*. En el capítulo 3 se detalla la metodología que se siguió para la preparación de los datos, la alineación, normalización y la instalación de las librerías necesarias para el entrenamiento del sistema de traducción automática. En el capítulo 4 se presentan los resultados de varios experimentos, así como la evaluación del desempeño y alcance de la arquitectura Transformer, usando los datos disponibles actualmente para la lengua *Ayuuk* de Güichicovi. Asimismo, se muestran las traducciones que genera el modelo entrenado. Por último, se presentan las conclusiones y los trabajos futuros que se pueden desarrollar para la lengua *Ayuuk* con base en los resultados obtenidos en este trabajo.

Capítulo 1

Antecedentes

En este capítulo se presenta a la lengua *Ayuuk* y su escritura; la historia de la Traducción Automática (TA) y cómo el Procesamiento de Lenguaje Natural (PLN) se ha aplicado en las lenguas indígenas de México; así como las arquitecturas de TA que han sido utilizadas antes del actual estado del arte.

1.1. La lengua *Ayuuk*

La lengua *Ayuuk* pertenece a una de las 11 familias lingüísticas que hay en México: la familia *mixe-zoqueana* (ver Figura 1.1). Esta familia lingüística está compuesta por las lenguas mixe (*Ayuuk*), sayulteco, oluteco, zoque, texiste pequeño y popoluca de la sierra (INALI, 2009), aunque Reyes Gómez afirma que hay dos variantes extintas y es necesario incluirlas, éstas son la tapachulteca[9] y el ayapaneco[10] (Reyes Gómez, 2005) (ver Cuadro 1.1).

Los *Ayuuk jya'ay* normalmente son identificados como *mixes*. La palabra *Ayuuk jya'ay* se compone las siguientes morfemas:

- *A, Aaw* = Lengua, palabra.
- *Yuuk* = Selva, montaña.
- *Jya'ay* = Gente.

[9]Variante extinta de la lengua mixe que se hablaba en las costas del Pacífico del estado de Chiapas.

[10]Variante extinta de la lengua zoque que se hablaba en una comunidad del estado de Tabasco.

Familia Mixe-Zoqueano
- Mixeano
  - **Mixe**
  - Tapachulteco †
  - **Sayulteco**
  - **Oluteco**
- Zoqueano
  - Zoqueano del Golfo
    - **Texiste pequeño**
    - Ayapaneco †
    - **Popoluca de la sierra**
  - **Zoque**

Cuadro 1.1: Variantes lingüísticas de las agrupaciones de la familia mixe-zoque. Adaptado de (INALI, 2009, p.291).

De ahí que, coloquialmente se puede traducir al español como *gente de la lengua de las montañas*.

Así también, el Instituto Nacional de los Pueblos Indígenas (INPI, 2017) afirma que de acuerdo a la tradición oral de los habitantes de la región *Ayuuk*, la palabra "mixes" pudo haber sido una descomposición de la palabra nativa *mixy* que en español significa (hombre, varón), y al agregarle el plural "es" da como significado "hombres", o también el origen del término pudo haber nacido de la mala pronunciación de los conquistadores españoles.

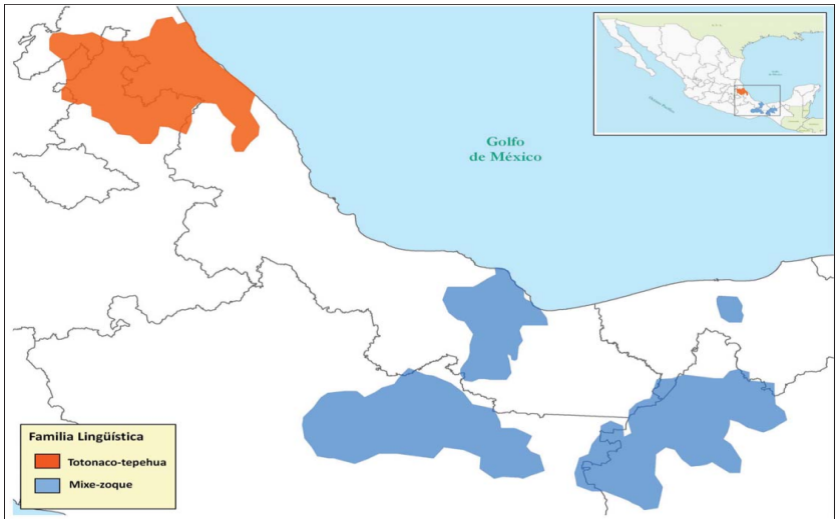


Figura 1.1: Región mixe-zoqueana abarca los estados de Oaxaca, Chiapas, Veracruz y Tabasco. Tomado de (INALI, 2009, p.316).

La región donde habitan los mixes se encuentra al noreste del estado de

Oaxaca (ver Figura 1.2). Colinda al norte con el distrito de Choapam y con el estado de Veracruz, al noroeste con el distrito de Villa Alta, al sur con el distrito de Yautepec y al sureste con los distritos de Juchitán y Tehuantepec; aunque es importante señalar que algunos municipios con hablantes del *Ayuuk* pertenecen a los distritos aledaños, como es el caso del municipio de San Juan Güichicovi que le corresponde al distrito de Juchitán (INPI, 2017). El territorio habitado por los *Ayuuk* abarca una superficie aproximada de 6,000 km^2 (Torres Cisneros, 2004).

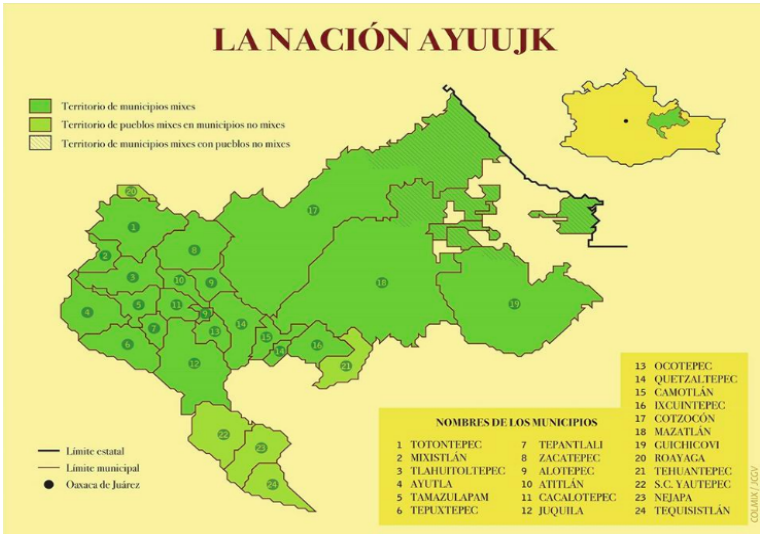


Figura 1.2: La nación *ayuujk*. Tomado de Colegio Mixe (COLMIX, 2016)[11] .

Con un total de 118, 882 hablantes, la lengua *Ayuuk* es la cuarta lengua con más hablantes en el estado de Oaxaca, y la decimosexta a nivel nacional con 139,760 hablantes de acuerdo al Censo de Población y Vivienda 2020 del Instituto Nacional de Estadística y Geografía (INEGI, 2020), sólo por detrás del zapoteco, mixteco y mazateco. Asimismo, al igual que las 68 lenguas indígenas habladas en la república mexicana (INALI, 2009, p.32) posee variantes como resultado de las distintas ubicaciones geográficas de su población hablante, asentándose ésta en 24 municipios del estado de Oaxaca, de acuerdo al Catálogo de las Lenguas Indígenas Nacionales del Instituto Nacional de Lenguas Indígenas (INALI, 2009, p.293), ver Cuadro 1.2.

[11]Tomado de: https://www.facebook.com/colmixe/photos/a.430737630369266/1043145989128424/?type=3&theater (Última visita 8 de junio de 2020).

Distrito	Municipio	Variante lingüística
Mixe y Villa Alta	Santa María Tlahuitoltepec, Santo Domingo Roayaga y Totontepec Villa de Morelos	Mixe alto del norte
Mixe	San Pedro y San Pablo Ayutla, Santa María Tlahuitoltepec y Tamazulapan del Espíritu Santo	Mixe alto del centro
Mixe	Mixistlán de la Reforma, Santa María Tepantlani y Santo Domingo Tepuxtepec	Mixe alto del sur
Mixe	San Juan Cotzocón, San Juan Juquila Mixes, San Miguel Quetzaltepec, San Pedro Ocotepec, Santa María Alotepec, Santiago Atitlán y Santiago Zacatepec	Mixe medio del este
Mixe	Asunción Cacalotepec y Santa María Alotepec	Mixe medio del oeste
Mixe, Tehuantepec y Juchitán	Camotlán, Coatlán, Matías Romero, San Juan Mazatlán, **San Juan Güichicovi**, Santo Domingo Petapa y Santo Domingo Tehuantepec	Mixe bajo

Cuadro 1.2: Municipios y distritos de Oaxaca donde se asientan los *Ayuuk jya'ay* (INALI, 2009, p. 293).

1.1.1. Variante del *Ayuuk* del municipio de San Juan Güichicovi

La variante del *Ayuuk* del municipio de San Juan Güichicovi está clasificada dentro de la variante lingüística del mixe bajo, concentrando una población de 29,802 habitantes (INEGI, 2020). Del total de la población, 28,298 tiene tres años y más (ver Cuadro 1.3), de los cuales:

- 18,984 habla lengua indígena (18,000 habla *Ayuuk* y 984 otra lengua)
  - 15,679 es bilingüe (habla español y *Ayuuk* u otra lengua)
  - 3,205 es monolingüe (únicamente habla *Ayuuk* u otra lengua)
- 9,310 no habla ninguna lengua indígena
- 4 no especificados

Sexo	**Población de 3 años y más**	**Situación en relación con la lengua indígena**					
		Habla lengua indígena				No habla lengua indígena	No especificado
		Total	Condición de habla española				
			Habla español	No habla español	No especificado		
Total	28 298	63.09 %	82.59 %	16.88 %	0.53 %	32.90 %	0.01 %

Cuadro 1.3: Datos del habla indígena del municipio de San Juan Güichicovi. Datos tomados de (INEGI, 2020). Elaboración propia.

Es importante destacar que debido a la constante migración de las personas y que el municipio mixe de San Juan Güichicovi históricamente ha sido una zona de paso, actualmente están asentados tres grupos étnicos en el área geográfica. La lengua predominante es el *Ayuuk* con un 94.82 % de hablantes, seguido del zapoteco, mixteco, chinanteco y otras lenguas con un 3.36 %, 0.63 %, 0.59 % y 0.60 %, respectivamente (INEGI,2020).

1.1.2. Estado actual de la escritura del *Ayuuk* variante del municipio de San Juan Güichicovi

De acuerdo con el lingüista mixe Juan Carlos Reyes, desde la llegada de los frailes dominicos a la región mixe en el siglo XVI hasta antes de los años ochentas del siglo pasado, personas provenientes de diversas instituciones han tenido el interés en estudiar la lengua *Ayuuk* ya sea para fines de evangelización o educación (Reyes Gómez,2005).

No fue hasta en el gobierno de Lázaro Cárdenas cuando el Instituto Lingüístico de Verano (ILV)[12] fue invitado a México para ayudar al gobierno en la misión de estudiar las lenguas indígenas del país, crear alfabetos y cartillas para la alfabetización y educación de los indígenas en su misma lengua.

La lengua *Ayuuk* variante del municipio de San Juan Güichicovi fue estudiada y documentada por el lingüista del ILV Norman W. Nordell, él estableció los primeros alfabetos y elaboró la cartilla de alfabetización del mixe Sanjua-

[12] https://mexico.sil.org/es. ILV o SIL(Summer Institute of Linguistics) es una organización evangélica norteamericana fundada por William C. Townsend dedicada a la investigación científica, materiales para la promoción de lecto-escritura y traducción de textos bíblicos a las lenguas indígenas.

nero[13].

Sin embargo, aunque la lengua mixe ha sido estudiada por muchas décadas, la escritura no se ha formalizado totalmente, esto se debe al florecimiento de partidarios locales (bodegueros y petakeros) que tenían la idea de trabajar y crear un alfabeto independiente al propuesto inicialmente por los lingüistas del ILV, pero retomando parte del material existente elaborado por dicho instituto (Sagi-Vela González, 2019). Actualmente, la escritura del *Ayuuk* variante del municipio de San Juan Güichicovi, así como las demás variantes de la lengua, sigue en controversia entre ambos partidarios, aunque en los primeros encuentros del SEVILEM (Semana de Vida y Lengua Mixe) se propuso la eliminación y sustitución de algunas consonantes, por ejemplo: reemplazar la *c* por la *k*; la *ñ* por la *ny*; la *ch* por la *tsy*.

Los investigadores que han documentado la lengua *Ayuuk* han identificado cuatro alófonos[14], mismos que son identificados principalmente por las personas hablantes nativas del *Ayuuk* que también hablan español. Estos alófonos[15] se representan como b,d, g y ds, las cuales son alófonos de los fonemas[16] p,t, k y ts. Dichos alófonos se presentan entre vocales ya sea en su forma alargada o rearticulada, y después de los fonemas consonánticos nasales (*m*, *n*) (Willett et al., 2018).

Los partidarios llamados "bodegueros" defienden el uso de las grafías y reglas de pronunciación igual que las del español porque ya se conoce el abecedario a emplear, lo que lo hace más intuitivo y que evita confusiones a la hora de leer y escribir en mixe, es por ello que defienden el uso de las consonantes *b*, *d*, *g* y *ds*; por el contrario, los "petakeros" argumentan que es mejor escribir con *p*, *t*, *k* y *ts* respectivamente (Sagi-Vela González, 2019), porque permite la simplificación y reducción del alfabeto pasando de 20 consonantes a solamente 14. En el subtema 1.1.2.2 "Consonantes para la escritura *Ayuuk*", se pueden apreciar las dos propuestas del uso y desuso de ciertos consonantes. Cabe seña-

[13]Esta cartilla se puede encontrar en la página del SIL como *Cartilla mixe Sanjuanero: En el idioma mixe de San Juan Guichicovi, Oaxaca*. https:/mexico.sil.org/es/resources/archives/11685

[14]De acuerdo con Oxford Languages un alófono es un sonido propio de la pronunciación de un fonema (sonidos de la lengua), que puede variar según su posición en la palabra o en la sílaba y en relación con los sonidos vecinos, aunque sigue considerándose el mismo fonema.

[15]*b*, *d*, *g* y *ds* presentan un sonido bajo.

[16]*p*, *t*, *k* y *ts* presentan un sonido alto

lar que de acuerdo con los trabajos de traducción recopilados, se puede notar que la escritura predominante en el *Ayuuk* del municipio de Güichicovi es la denominada "escritura bodega".

1.1.2.1. Vocales para la escritura del *Ayuuk*

Actualmente la lengua *Ayuuk* cuenta con nueve representaciones vocálicas. El número de vocales varía de acuerdo a la variante en la que se esté escribiendo, ya que en unas variantes existen más sonidos que en otras. Estas vocales son:

(1) a ä e ë i o ö u ü

En la variante del municipio de **San Juan Güichicovi** los sonidos se representan usando las cinco vocales del español y una sexta vocal[17], donde ésta es la más frecuente y se representa con la grafía **ë**. Las vocales son:

(2) a e ë i o u

1.1.2.2. Consonantes para la escritura del *Ayuuk*

Para los "bodegueros" las representaciones consonánticas se escriben con (Willett et al., 2018):

(3) b ch d ds g j k l m n ñ p r s t ts w x y ’

En cambio, los "petakeros" afirman que es mejor reducir las representaciones consonánticas a solamente catorce, debido a que facilita tanto la escritura y la comprensión lectora (Reyes Gómez, 2005), éstas son:

(4) p t k x ts m n w y j l r s ’

[17] En las primeras documentaciones la sexta vocal se representaba con las grafías ø, ä y con una u atravesada por una barra horizontal. De acuerdo con Reyes Gómez (2005) la grafía ë representa el sonido medio central.

1.1.2.3. Algunos ejemplos de la escritura del *Ayuuk* de la variante del municipio de San Juan Güichicovi

Las grafías vocálicas antes mencionados se pueden representar de seis maneras diferentes dependiendo del tipo de sonido que se esté representando (Willett et al., 2018; Reyes Gómez, 2005), se ejemplifican en el Cuadro 1.4 y se representan de la siguiente manera:

- Vocal sencilla (a, e, i, o, u, ë).
- Vocal alargada (aa, ee, ii, oo, uu, ëë).
- Vocal quebrada (a'a, e'e, i'i, o'o, u'u, ë'ë).
- Vocal sencilla cortada (a', e', i', o', u', ë)'.
- Vocal alargada cortada (aa', ee', ii', oo', uu', ëë').
- Vocal aspirada (aj, ej, ij, oj, uj, ëj).

Vocales	**a**	**e**	**i**	**o**	**u**	**ë**
Sencilla	Pax *Novia, mazate*	Teky *Pie*	Pixk *Pulga*	Tok *Peste*	Tuk *Viejo, tortuga*	Pën *¿Quién?*
Alargada	Kaa *Tigre*	Eexy *Cangrejo*	Piik *Trompo*	Poo *Tlacuache*	Tuu *Lluvia*	Xëë *Sol, fiesta, nombre*
Quebrada	Ta'ak *Tejer, Chahuitera*	Pe'et *Barrer*	Xi'ik *Reír*	Po'ot *Pulir*	Pu'uts *Grano, lepra*	Të'ëts *Flaco, seco*
Sencilla cortada	Xa'k *Astilla, maguey*	Te'k *Sucio*	Ti'ty *Amargo*	Po't *Estropajo*	Pu'ts *Amarillo*	Pë'ty *Limpio*
Alargada cortada	Kaa'pt *Cuerno, cuchara*	Xee'x *Gotea*	Lii'xy *Poquito*	Too'pë *Vendedor*	Xuu'p *Apesta*	Këë'ts *Roto*
Aspirada	Kaj *No*	Xej *Respira*	Pijty *Bolsa*	Xoj *Encino*	Puj *Lavar*	Këjx *Se acabó*

Cuadro 1.4: Ejemplos del uso de las diferentes representaciones vocálicas en la escritura del *Ayuuk* de la variante de Güichicovi. Adaptado de (Santiago Cayetano, s.f.).

A continuación se muestra un poco del uso de las consonantes con la escritura recolectada, donde se puede apreciar que aún siendo de la misma variante hay palabras que se escriben diferente, una de las razones la falta de estandarización como se ha venido mencionando. Con lo anterior se da un panorama de cómo es la escritura *Ayuuk*. Se han etiquetado y subrayado algunas observaciones para hacer un breve análisis.

Fragmentos de la fábula de *ESOPO*[18].

a1) *ja**t**u'un ta'ook je jyajtë, ja xëkaa pajk yo't je'x.*
Érase una vez, cuando el coyote se atragantó con un hueso.

a2) *janytyim xyexëk jeya'ay ijty.*
Era un hombre muy testarudo.

a3) *xëts xëts je'e yajtsiiy ka**p**ëk je'e tsyejky.*
Aunque lo metió dentro de agua enjabonada no pudo quitarle su color.

Fragmentos de *Na Ap Na Deedy Kuentë*[19].

b1) *jantim xyondaak ja koy jadu'un.*
El conejo se puso muy feliz.

b2) *kabëk je'e ti y'ok ëjy y'ok nójnë.*
Cuando todo se quedó en silencio.

Fragmentos del libro de San Mateo de la Biblia[20].

c1) *jëdu'un je'e jyajtë tyuunë mëna Jesucristo myiiñ gya'ayën.*
Esto fue lo que sucedió cuando Jesucristo nació.

c2) *kabëk tyijy tyeety o tyaj myëjpëdaaknët.*
No debería hacerle caso a sus padres.

La primera observación corresponde a las palabras que tienen el mismo significado y están en las frases:

[18]Fábulas traducidas por Melchor Escobar Ocaña y Gláfira Azcona Figueroa, usando el método de escritura "petaka".

[19]Cuentos populares recolectados y escritos por Albino Pedro Jacinto, siguiendo el sistema de escritura "bodega".

[20]Traducido por Victoriano Santiago Cayetano, siguiendo el sistema de escritura "bodega".

- a1) *ja**t**u'un.*
- a2) *ja**d**u'un.*
- a3) *jë**d**u'un.*

La palabra que es parte de la frase **a1** está escrita con la consonante *t*, a comparación de las palabras que se encuentran en las frases **a2** y **a3** que usan la *d*, asimismo, estas dos últimas cambian con el uso de la *a* y *ë* respectivamente. Por lo que se observa el choque de escrituras entre los sistemas "bodega" y "petaka", así como una falta de acuerdo con la correcta escritura de las palabras.

La siguiente observación corresponde a las palabras que se encuentran en las frases:

- a2) *janytyim.*
- b1) *jantim.*

Ambas palabras son correctas y tienen el mismo significado.

La tercera observación corresponde a las palabras que están contenidas en las frases:

- a3) *ka**p**ëk.*
- b2) *ka**b**ëk.*
- c2) *ka**b**ëk.*

La palabra que es parte de la frase **a3** utiliza la consonante *p*, en cambio las palabras que se encuentran en las frases **b2** y **c2** utilizan *b*, presentándose nuevamente el problema de la estandarización de la escritura.

(El proceso de normalización de los textos se abarca en el Capítulo 3 de este trabajo).

1.1.2.4. Estructura morfológica de la lengua *Ayuuk*

Antes de pasar a los experimentos es necesario tener identificada la estructura morfológica de la lengua en cuestión, la lengua *Ayuuk* es considerada una

lengua lengua aglutinante, llamada de esta manera debido a que las palabras se forman de varios monemas independientes[21]. Ejemplos:

(5) yajnë'ixëëy
prefijo + raíz + sufijo
yaj + në'ix + ëëy

La traducción de la palabra *yajnë'ixëëy* significa *le enseñó*. Donde, *yaj* es un prefijo que indica ejecutar una acción; *në'ix* es la raíz de la palabra y significa mirar, observar; *ëëy* es un sufijo que indica tiempo pasado.

(6) mdëjk
prefijo + raíz
m + tëjk

La traducción de la palabra *mdëjk* significa *tu casa*. Donde, *m* indica posesión en segunda persona y *tëjk* es la raíz de la palabra, que significa casa.

1.2. La traducción automática y su historia

La TA es una de las aplicaciones del PLN, donde mediante un programa computacional, un lenguaje natural *origen* en forma de texto se transforma en otro lenguaje natural *meta* (EAMT, 2008; W. J. Hutchins y Somers, 1992).

La calidad de la TA dependerá mucho del tipo de arquitectura que se utilice, cantidad de texto, características morfológicas de la lengua[22], estado de la escritura, entre otros factores. Aunque actualmente con el estado del arte basado en redes neuronales los resultados de traducción son más significativos, se debe tener en cuenta que el lenguaje humano es muy complejo, y lo que se pretendía lograr con la TA hace más de 60 años, aún no se alcanza, por lo que sigue siendo un campo abierto a la investigación. Así que para comprender cómo funciona el estado del arte actual, es necesario remontarnos unas décadas

[21]Donde un monema de puede descomponer en lexemas y morfemas. Por lo que, un monema es una unidad mínima del lenguaje dotada de significado; lexema es la raíz de la palabra; y un morfema aporta el aspecto gramatical de la palabra modificando el sentido de un lexema, por ejemplo los afijos, sufijos e infijos (Juan Ampuero, 2008)

[22]De acuerdo con Moreno Cabrera (2003) una lengua puede ser: Aislante, aglutinante, flexiva o fusionante, incorporante y polisintética

atrás.

Schwartz (2018) menciona que las primeras ideas sobre la creación de un lenguaje universal para que la humanidad se pudiera comunicar entre sí, se remontan hasta el siglo XVII, cuando filósofos como Descartes, Leibniz, Wilkins, entre otros, proponen la construcción de un lenguaje universal. Sin embargo, no fue hasta en la década de los 30's del siglo pasado cuando aparecen las primeras propuestas de mecanizar la traducción.

En 1933, antes del nacimiento de la computadora, el francés Georges Artsrouni y el ruso Petr Trojanskij [23], de manera independiente en sus respectivos países, patentaron las primeras propuestas para mecanizar el proceso de traducción con un diccionario bilingüe mecánico. Pese a que una de estás propuestas resultaba más interesante que otra, en aquel tiempo, estas ideas de mecanización no fueron conocidos en otros países (W. J. Hutchins, 2001).

Después de la segunda guerra mundial en 1946, el científico británico Andrew Booth y el matemático estadounidense Warren Weaver, discutieron sobre la posibilidad de implementar las computadoras para traducir lenguajes naturales que recientemente habían tenido éxito descifrando mensajes alemanes. En el año de 1949 Warren Weaver presenta varias propuestas en un *memorándum* sobre traducción, donde abordaba varios métodos para implementarlo en una máquina de traducción; entre ellos, criptográficos y estadísticos. Gracias a este *memorándum*, pocos años después comenzaron a formarse grupos de investigación en TA en varias universidades de los Estados Unidos, así como en el resto del mundo (W. J. Hutchins y Somers, 1992; Schwartz, 2018).

Años más tarde, el 7 de enero de 1954, investigadores de la Universidad de Georgetown e IBM realizan la primera demostración en público para mostrar la factibilidad de la TA, intentando traducir oraciones del ruso al inglés previamente seleccionadas, limitado a solo 250 palabras y con solo seis reglas gramaticales; como resultado, el programa pudo traducir 49 oraciones (W. J. Hutchins, 2001). Este acontecimiento llamó la atención del gobierno estadounidense y fue lo suficientemente impresionante para que se comenzaran a financiar grupos de investigación en TA, dado que en plena guerra fría Estados Unidos necesitaba conocer el contenido de documentos que aparecían en revistas científicas rusas.

[23]La propuesta de Trojanskij abarcaba reglas gramaticales

Después del acontecimiento histórico en los 50's en la Universidad de Georgetown, años después todo ese optimismo fue desapareciendo cuando empezaron a enfrentarse con problemas lingüísticos demasiado complejos. En 1966, se emite el famoso "informe ALPAC" que frenó la inversión en la investigación en TA por la falta de viabilidad. Pese a ello, Estados Unidos no detuvo sus investigaciones por completo, así como en Canadá y Europa (W. J. Hutchins, 2001).

En el año de 1970 Latsec Inc. lanza la primera versión del sistema Systran. Inicialmente este sistema trabajó con la traducción del ruso al inglés para ser usado en la fuerza aérea de los E.E.U.U, su arquitectura era por traducción directa (ver Figura 1.3); siendo así el primer traductor basado en reglas (W. J. Hutchins y Somers, 1992). En 1972 en la Universidad de Grenoble, aparece el Groupe d'Etudes pour la Traduction Automatique (GETA) dirigido por Bernard Vauquois, para diseñar un sistema de traducción del ruso al francés, como resultado aparece el sistema Ariane, con una arquitectura basada en transferencias; inicialmente había sido diseñado con un enfoque interlingüístico (ver Figura 1.3). En 1977 entra en operación el sistema de traducción meteorológico Météo desarrollado con el proyecto TAUM (Traduction Automatique de l'Université de Montréal), este sistema funcionaba por transferencia sintáctica, para traducir reportes meteorológicos del inglés al francés (W. J. Hutchins y Somers, 1992).

De acuerdo con W. J. Hutchins (2001), en los 80's surgen muchos sistemas de TA y se suman más países al desarrollo de sistemas de traducción, aparece el sistema Logos para la traducción del alemán al inglés y principal competencia comercial de Systran, este sistema de traducción basado en transferencia inicialmente traducía manuales de aviones del inglés al vietnamita. Japón empieza a desarrollar el sistema MU, para la traducción del inglés al japonés. Asimismo, arranca el proyecto EUROTRA por parte de las Comunidades Europeas (CE) para crear un sistema de transferencia multilingüe, debido a que en la CE había diferencias lingüísticas. En Alemania aparece el sistema multilingüe SUSY (Saarbrücker Übersetzungssystem) que se desarrolló con una arquitectura basada en transferencia para traducir del ruso al alemán, inglés al francés y del inglés al alemán. Otro sistema basado en transferencia aparece en 1989 en la Universidad de Texas; el sistema de traducción METAL para la traducción del alemán al inglés (W. J. Hutchins y Somers, 1992).

A mediados de la década de los 80's las investigaciones vuelven a centrarse en el método por interlingua, principalmente en los Países Bajos. En el laboratorio de investigación de Phillips en Holanda, desarrollaron el sistema Rosseta; sistema basado en interlingua que hacía traducciones del holandés al inglés y español, y de inglés o español al holandés. Distributed Language Translation (DTL), es el otro sistema de traducción basado en interlingua que emergió de la compañía Utretch, fue un sistema multilingüe basado en el Esperanto[24] (W. J. Hutchins y Somers, 1992). De igual manera, en Japón aparecen proyectos de TA multinacionales, con la participación de China, Indonesia, Malasia y Tailandia (J. Hutchins, 2006). En la Unión Soviética aumentó la actividad en proyectos de TA del inglés al ruso y del alemán al ruso, donde los sistemas de TA ya implicaban análisis estadísticos superficiales. Asimismo, a finales de la década varios proyectos en Japón, Europa y en los Estados Unidos empezaban a aplicar enfoques basados en ejemplos (J. Hutchins, 2006), ver sección 1.3.2.

En los 90' la capacidad computacional había aumentado y el método estadístico era el principal tema de investigación, ver sección 1.3.3, aparecen los sistemas basados en corpus[25], por lo que los textos paralelos empezaron a usarse. A principios de esta década, IBM lanza el primer sistema de traducción estadístico del francés al inglés: CANDIDE (Berger et al., 1994). El alineador de textos juega un papel importante en la traducción estadística, por lo que en la primera mitad de esta década, IBM desarrolló cinco modelos de alineación cada vez más sofisticados (Brown et al., 1993).

En los 2000 se desarrolla un nuevo paradigma de TA; traducción estadística por frases (Koehn et al., 2003). Este nuevo modelo reemplazó la traducción estadística basada en palabras. Por mucho tiempo la traducción estadística por frases fue el estado del arte y traductores web como el de Google funcionaban con esta arquitectura. Sin embargo, con el aumento de la capacidad computacional y la cantidad de datos, después de 2010 aparecen las primeras arquitecturas de traducción neuronal con RNN's (Cho, van Merrienboer, Gulcehre, et al., 2014). Poco después surgen los modelos con atención para

[24]Lengua creada a finales del siglo XIX por el polaco Ludwik Zamenhof, con la finalidad de mejorar la comunicación entre personas de diferentes comunidades que hablaban distintos lenguajes.

[25]Un corpus es una colección de diversos tipos de textos que representan el lenguaje natural de una lengua en específico, generalmente almacenados en una base de datos electrónica (Dash, 2008).

mejorar el desempeño de las RNN's (Bahdanau et al., 2014). En 2017 aparece la arquitectura Transformer (Vaswani et al., 2017), donde los modelos RNN codificador-decodificador con mecanismos de atención, fueron superados y desplazados por esta nueva arquitectura, por lo que actualmente Transformer es estado del arte en traducción automática y en PLN.

1.2.1. Traducción automática en las lenguas indígenas de México

Actualmente en México existen 68 lenguas, algunas de ellas están en riesgo de desaparecer. En la mayoría de los casos, la tradición oral ha sido el medio principal para la herencia de la lengua de una generación a otra. Sin embargo, no tener un registro lingüístico de una lengua aumenta el riesgo de desaparecer más rápido.

La inexistencia de una escritura formal para cada lengua en el país, en la mayoría de los casos se debe a la falta de especialistas en el área, la dificultad para la representación de los sonidos, la complejidad lingüística, entre otros factores. Por lo tanto, el avance del estudio lingüístico de las lenguas indígenas ha sido muy paulatino y, como consecuencia, los materiales son escasos.

Derivado de lo anterior, en México la mayoría de las investigaciones en PLN se han centrado en las lenguas predominantes como español e inglés, pero en los últimos años ha habido interés por trabajar con lenguas indígenas, ya sea en la creación de corpus y recursos digitales, analizador y segmentador morfológico o en la traducción automática. En el artículo *Challenges of language technologies for the indigenous languages of the Americas* de Mager, Gutierrez-Vasques, et al. (2018) se hace un compendio de lo que se ha trabajado hasta el momento con las lenguas de América, las líneas de investigación y problemas a los que se enfrentan, éstos son:

Corpus y recursos digitales

- Axolotl[26], corpus paralelo Náhuatl-Español (Gutierrez-Vasques et al., 2016). Cuenta con 18,000 frases.
- Diccionario Náhuatl[27] (UNAM, 2012).

[26] http://www.corpus.unam.mx/axolotl
[27] http://www.gdn.unam.mx/

- Corpus paralelo Wixarika-Español[28] (Mager, Carrillo, y Meza, 2018). Cuenta con 8,000 frases.
- Corpus de voz del Chatino[29] [30] (Cavar et al., 2016). Cuenta con 10 horas de audio.
- Inflexión morfológica[31] de las 20 lenguas de la familia Oto-Mangue (Feist y Palancar, 2015). Cuenta con 13,000 verbos.
- Segmentación morfológica[32] con modelos neuronales de las lenguas (Mexicanero,Náhuatl,Wixarika, Yorem Nokki) de la familia Uto-Azteca (Kann, Mager Hois, Meza Ruiz, y Schütze, 2018). Cuenta con 4,468 palabras segmentadas.

Herramientas de segmentación y análisis morfológico

- Segmentador morfológico *"chachalaca"* basado en reglas para la lengua Náhuatl[33] (Thouvenot, 2011).
- Segmentador morfológico para la lengua Wixarika[34] (Mager, Carrillo, y Meza, 2018).
- Detección automática de afijos sin supervisión para las lenguas Ralámuli (Rarámuri o Tarahumara) (lengua de la familia Uto-Azteca) y Chuj (Variante del Maya) (Medina-Urrea, 2007).

Traductor automático

- Gutierrez-Vasques (2015) utiliza representaciones morfológicas del náhuatl para alinearlo con palabras del español.
- Sistema comercial Microsoft Translator[35] [36], es un traductor automático neuronal (NMT) que incluye el Maya Yucateco y Otomí de Querétaro.
- Traductor automático estadístico Wixarika-Español[37] (Mager Hois, 2017).

[28] https://github.com/pywirrarika/wixarikacorpora
[29] https://elar.soas.ac.uk/Collection/MPI113663
[30] https://www.ailla.utexas.org/
[31] https://oto-manguean.surrey.ac.uk/
[32] http://turing.iimas.unam.mx/wix/mexseg
[33] https://cen.sup-infor.com/#/home/hellow
[34] https://github.com/pywirrarika/smtwixes/tree/master/wixnlp
[35] https://www.microsoft.com/en-us/translator/business/languages/
[36] https://www.bing.com/translator?ref=MsftMT
[37] http://turing.iimas.unam.mx/wix

- Traductor automático Purépecha-Español[38] usando la herramienta OpenNMT (Soriano García, 2018).

Otros enfoques

- Diseño de producción artificial del habla Ralámuli (Rarámuri o Tarahumara) (lengua de la familia Uto-Azteca) utilizando palabras funcionales, secuencias de sufijos y difonos de la lengua (Urrea et al., 2009).
- Digitalización de diccionarios utilizando Reconocimiento Óptico de Caracteres (OCR, por sus siglas en inglés) aplicado a la lengua tzetzal (lengua de la familia Maya) (Maxwell y Bills, 2017).

1.3. Arquitecturas de traducción automática

Desde las primeras propuestas para mecanizar la traducción del lenguaje natural hasta en la actualidad, en los diferentes momentos de la historia de la TA los investigadores se han enfrentado a la complejidad del lenguaje humano y como consecuencia han surgido distintos métodos, desde la traducción directa hasta las arquitecturas basadas en redes neuronales. De igual manera, es indispensable señalar que otro factor importante para el desarrollo de las diferentes arquitecturas ha sido el incremento de la capacidad de procesamiento de las computadoras.

Actualmente los paradigmas existentes de la Traducción Automática son: traducción basada en reglas (RBMT, por sus siglas en inglés), traducción basada en ejemplos (EBMT, por sus siglas en inglés), traducción automática estadística (SMT, por sus siglas en inglés) y el estado del arte actual; traducción automática neuronal (NMT, por sus siglas en inglés).

1.3.1. Traducción basada en reglas

Este paradigma se basa principalmente en el análisis morfológico, sintáctico y semántico de las lenguas origen y lenguas meta, así como el uso de diccionarios bilingües (Bhattacharyya, 2015). El conocimiento sobre las teorías del lenguaje es muy indispensable en este enfoque, por lo que se necesitan expertos lingüistas para construir el modelo de traducción, lo que lo hace muy costoso.

[38] http://turing.iimas.unam.mx/purepecha/

Una manera de visualizar esta arquitectura es con el triángulo de Bernard Vauquois (ver Figura 1.3), donde se muestran los niveles de complejidad dependiendo de la profundidad de la arquitectura.

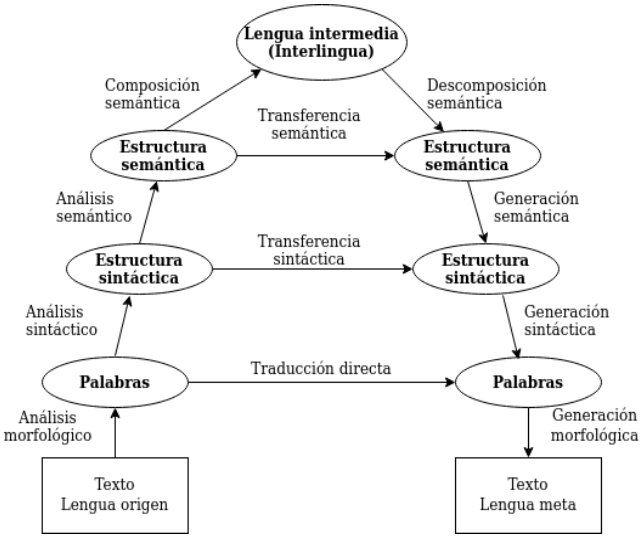


Figura 1.3: Triángulo Vauquois. Adaptado de (W. J. Hutchins y Somers, 1992, p.107).

A su vez la RBTM se divide en traducción directa, traducción por transferencia y traducción por interlingua.

1.3.1.1. Traducción directa

El sistema de traducción directa, es la estrategia menos compleja de todas y también es considerada como la primera generación de sistemas de TA (ver Figura 1.4). La traducción consiste en reemplazar palabra por palabra el texto de una lengua origen a una lengua meta, con un mínimo análisis morfológico al texto origen para identificar la clase[39] a la que pertenece cada palabra, y con los resultados obtenidos se hace una consulta en un diccionario bilingüe que proporciona significados equivalentes a la lengua destino, aplicando pequeñas reglas de ordenamiento (J. Hutchins, 1982; Poibeau,2017; W. J. Hutchins y Somers, 1992).

Esta estrategia de traducción depende mucho de la información contenida en el diccionario, ya que no es capaz de identificar palabras con varios significados. El par de lenguas con el que esté diseñado debe tener un vocabulario bien delimitado.

[39]Adverbio, determinante, pronombre, sustantivo, verbo y preposición

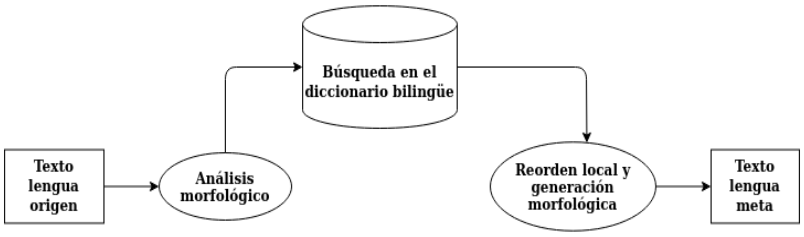


Figura 1.4: Arquitectura de traducción directa. Adaptado de (W. J. Hutchins y Somers, 1992, p.72).

1.3.1.2. Traducción por transferencia

En el sistema de traducción por transferencia, tanto la lengua origen como el destino cuentan con sus propias representaciones sintácticas y semánticas (ver Figura 1.5). Asimismo, utiliza una representación intermedia el cual sirve como enlace para traducir de la lengua origen a la lengua meta. Por lo tanto, la traducción se lleva a cabo en tres etapas: primero se realiza un análisis de la estructura gramatical de la lengua origen para generar una representación intermedia con información morfológica, sintáctica y/o semántica; en la etapa de transferencia, las representaciones intermedias de la lengua origen se transforman en representaciones intermedias de la lengua destino, mediante el uso de un diccionario bilingüe; y en la fase de generación, se hace una reconstrucción de los textos en la lengua meta (J. Hutchins, 1982; W. J. Hutchins y Somers, 1992; Li, 2013).

Este método resulta ser muy costoso conforme se van agregando más pares de lenguas, porque se tiene que diseñar las fases de transferencia para cada par de éstas.

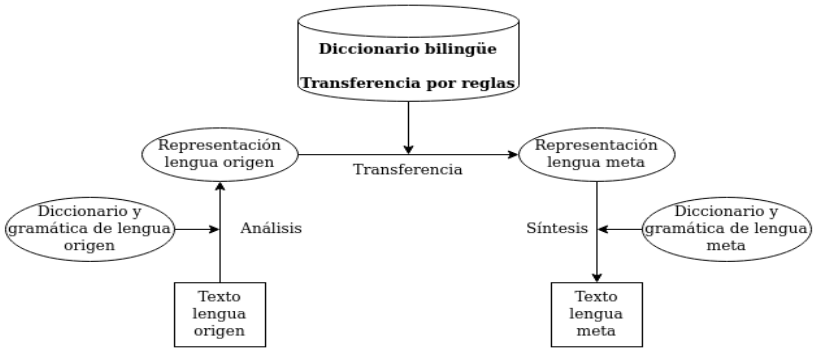


Figura 1.5: Arquitectura de traducción por transferencia. Adaptado de (J. Hutchins, 1982, p.29).

1.3.1.3. Traducción por interlingua

El sistema de traducción por interlingua o lenguaje pivote (ver Figura 1.6), surge de la idea de que todas las lenguas cuentan con características comunes. Como ejemplo, podemos mencionar que en el siglo XIX se creó una lengua artificial llamada Esperanto, que fue resultado de la extracción de características comunes de lenguas europeas, su intención era ser universal para mejorar la comunicación entre personas que no compartían el mismo lenguaje.

En ese sentido, el interlingua es una representación intermedia y abstracta tanto de la lengua origen como de la lengua destino, por lo que hay dos etapas en el proceso de traducción: análisis y síntesis. En la etapa de análisis, de la lengua origen se genera la representación intermedia y, después, a partir de esa representación, se genera el texto a la lengua destino. En este sistema no existe la fase de transferencia (Li, 2013;W. J. Hutchins y Somers, 1992).

Es un método económico para diseñar sistemas de traducción multilingües, debido a que evita el diseño de la fase de transferencia.

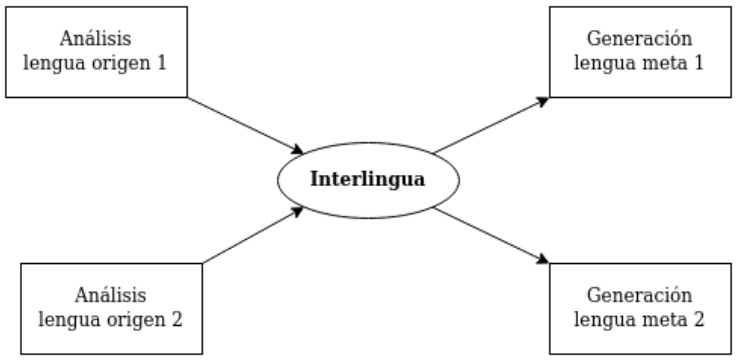


Figura 1.6: Arquitectura de traducción por interlingua. Adaptado de (W. J. Hutchins y Somers, 1992, p.74).

1.3.2. Traducción basada en ejemplos

La traducción basada en ejemplos o paradigma basado en analogía, es un método intermedio entre la traducción basada en reglas y la traducción estadística.

El entrenamiento de la EBMT es similar a la SMT basada en frases, donde ambos utilizan un corpus alineado (Matiss, 2019). Asimismo, los patrones de traducción son extraídos de los datos, pero en el mayor de los casos se utilizan un conjunto de reglas para establecer los patrones (Bhattacharyya, 2015).

Entonces, lo que EBTM hace es obtener frases similares o plantillas en el corpus de la lengua origen, después recupera la frase equivalente en el corpus de la lengua destino, logrando así una traducción parcial que posteriormente se combina. En cambio, la traducción estadística obtiene probabilidades (Bhattacharyya, 2015).

1.3.3. Traducción automática estadística

La traducción automática estadística es capaz de traducir de una lengua cualquiera a otra, ya que se genera a partir de un corpus alineado, es decir, dos textos emparejados de diferentes lenguas, por lo que no necesita tener conocimientos lingüísticos (Li, 2013), ver Figura 1.7.

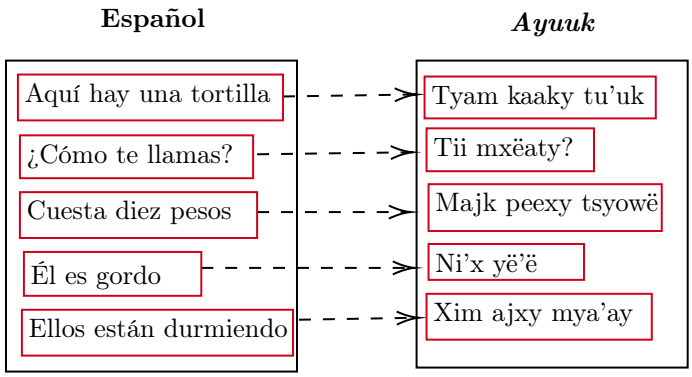


Figura 1.7: Ejemplo de un corpus paralelo, cada oración del español está alineado a su correspondiente al *Ayuuk*. Elaboración propia.

Formalizando la idea anterior, de acuerdo con Brown et al. (1993) en la SMT se toman dos oraciones, una en una lengua origen f y otra en una lengua destino e. Así pues, el sistema de traducción calcula la probabilidad de que una oración del lenguaje meta e sea la traducción de una oración de una lengua origen f, esto es $P(e|f)$. Usando el teorema de Bayes, se puede escribir como:

$$P(e|f) = \frac{P(e)P(f|e)}{P(f)} \tag{1.1}$$

Sin embargo, la cantidad de posibles traducciones e es muy grande, por lo que se maximiza la probabilidad para encontrar la mejor traducción $\hat{e}$, esto se logra haciendo el producto de la probabilidad $P(e)$ con la probabilidad condicional $P(f|e)$, es decir:

$$\hat{e} = argmax_e P(e)P(f|e) \tag{1.2}$$

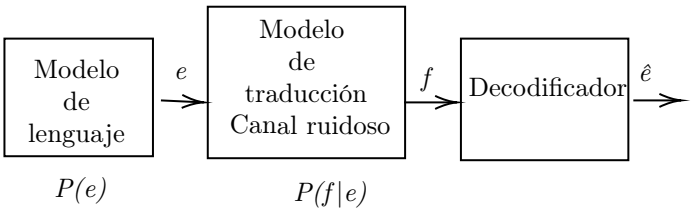


Figura 1.8: Sistema de traducción estadístico. Adaptado de (Brown et al., 1990).

Donde $P(f|e)$ representa la probabilidad de que f surja del canal ruidoso cuando se le presenta la entrada e, esta función se conoce como *modelo de traducción*, donde el dominio son los pares de cadenas (f, e), ver ecuación 1.4. Asimismo, esta función está compuesta por el modelo léxico y el modelo de alineación. De igual modo, $P(e)$ representa la probabilidad *a priori* de e y, ayuda a evitar que las oraciones destino tengan una salida sintáctica incorrecta. Esta función es el *modelo de lenguaje* (Brown et al., 1990) expresado como:

$$P(e_1, e_2, e_3, ..., e_I) = P(e_1)P(e_2|e_1)P(e_3|e_1, e_2)...P(e_n|e_1, e_2, ..., e_{I-1}) \tag{1.3}$$

1.3.3.1. Estadístico por palabras

La traducción basada en palabras no puede identificar términos que tienen múltiples significados, por lo que no toma en cuenta el contexto de las oraciones, siendo así, puramente léxica.

Para empezar a definir la correspondencia entre las palabras de las oraciones de origen f_1^J y destino e_1^I, se lleva a cabo la *alineación*. Sea I la longitud de e en palabras; J la longitud de f en palabras; f_j representación de una palabra de f; a una función de posiciones para la representación de la alineación entre ambas lenguas.
Entonces la *alineación* (ver Figura 1.9), es representada como $j \rightarrow i = a_j$,

donde a_j es la posición e_i con el que f_j está alineado, e_{a_j} la palabra en e con el que f_j está alineado y $a_j = 0$ si una palabra f_j está alineado a una palabra vacía e_0.[40]

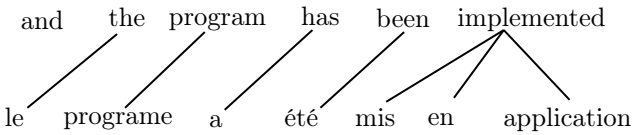


Figura 1.9: Ejemplo de alineación palabras del francés al inglés, donde $a = \{ 1 \to 2, 2 \to 3, 3 \to 4, 4 \to 5, 5 \to 6, 6 \to 6, 7 \to 6 \}$. Adaptado de (Brown et al., 1993).

Entonces la probabilidad del *modelo de traducción* $P(f|e)$ se puede escribir como (Zens et al., 2002)

$$P(f_1^J|e_1^I) = P(J|e_1^I) \sum_{a_1^J} \prod_{j=1}^{J} [P(a_j|a_{j-1}, I, J) P(f_j|e_{a_j})] \tag{1.4}$$

donde:

$P(J|e_1^I)$ es la probabilidad de la longitud de la oración.
$P(f|e)$ probabilidad del léxico.
$P(a_j|a_{j-1}, I, J)$ la probabilidad de alineación.

1.3.3.2. Estadístico por frases

La traducción por palabras tiene la desventaja de no poder identificar términos que tienen múltiples traducciones, por lo que el contexto de la información es ignorado. La traducción por frases como se ejemplifica en la Figura 1.10, llega a resolver este problema, donde una secuencia de palabras de una oración de entrada son frases (Zens et al., 2002).

Sean $\bar{f}_i$ y $\bar{e}_i$ frases del lenguaje origen y objetivo respectivamente. Una oración de origen f es segmentado en I frases $\bar{f}_1^I$, cada frase $\bar{f}_i$ es traducido a una frase $\bar{e}_i$ y donde $\phi(\bar{f}_i|\bar{e}_i)$ es la distribución de probabilidad de traducción, ver ecuación 1.6. Asimismo, cuenta con un modelo de reordenamiento definido como $d(a_i - b_{i-1})$, siendo a_i la representación de la posición inicial de la frase

[40]De acuerdo con Brown et al., (1993) las palabras que no tienen ninguna correspondencia con ningún término en la lengua objetivo, se denota como e_0 y por convención se sitúa en la posición 0.

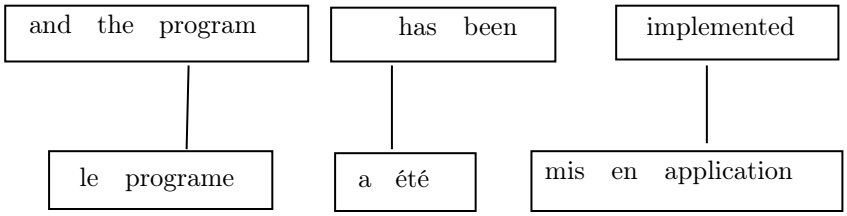


Figura 1.10: Ejemplo de alineación de frases del francés al inglés. Adaptado de (Brown et al., 1993).

de origen que se traduce a la i-ésima frase destino, y b_{i-1} es la última posición de la frase origen traducida a la $(i-1)$ frase destino (Koehn et al., 2003). Por lo tanto el *modelo de traducción* por frases queda representado como:

$$P(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) d(a_i - b_{i-1}) \tag{1.5}$$

Donde la distribución de probabilidad de traducción por frases se expresa como:

$$\phi(\bar{f}|\bar{e}) = \frac{count(\bar{f}, \bar{e})}{\sum_{\bar{f}} count(\bar{f}, \bar{e})} \tag{1.6}$$

Finalmente, para calibrar la longitud de salida se introduce un factor ω para cada palabra destino generado, así como el *modelo de lenguaje* $P(e)$. Por lo tanto, el sistema de traducción estadístico por frases se puede escribir como:

$$\hat{e} = argmax_e P(f|e) P(e) \omega^{longitud(e)} \tag{1.7}$$

Capítulo 2

Estado del arte

En este capítulo se explican las arquitecturas de traducción automática neuronal (NMT), como los modelos *codificador-decodificador* con RNN's y Transformers; herramienta para el entrenamiento del modelo, JoeyNMT; alineadores automáticos; métricas de evaluación que miden el desempeño del modelo de traducción; por último, los datos disponibles del *Ayuuk* para la variante de Güichicovi.

2.1. Traducción neuronal

Actualmente la traducción neuronal (NMT, por sus siglas en inglés) constituyen el estado del arte de la traducción automática (Koehn, 2017), éste es un método basado en redes neuronales artificiales que, al igual que el método estadístico, son entrenados a partir de un corpus bilingüe alineado.

Una red neuronal analiza datos representados en forma numérica, por lo que en la NMT cada palabra es asociada a un vector de características (Bengio et al., 2003).

2.1.1. ¿Qué es una red neuronal?

El concepto de red neuronal surge de la imitación del funcionamiento de las conexiones neuronales que ocurre dentro del cerebro humano (ver Figura 2.1). Estas neuronas se conectan mediante *sinapsis*, intercambiando pequeñas descargas eléctricas entre los *botones sinápticos* de una neurona y las *dendritas* de otra. Donde las *dendritas* son las entradas de una neurona, *soma* es la función de activación y el *axón* la salida.

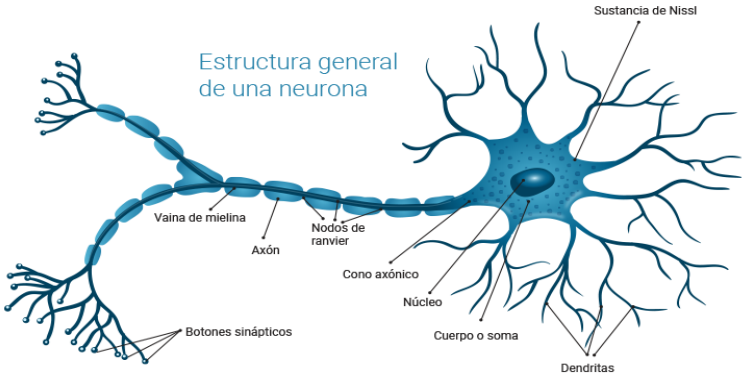


Figura 2.1: Inspiración biológica de las redes neuronales artificiales: Estructura de una neurona [41] .

Una neurona artificial o *perceptrón* tiene entradas que se multiplican por sus pesos y luego se suman, posteriormente se agrega un sesgo y aplica una función no lineal para dar una salida (Goldberg, 2017), ver Figura 2.2.

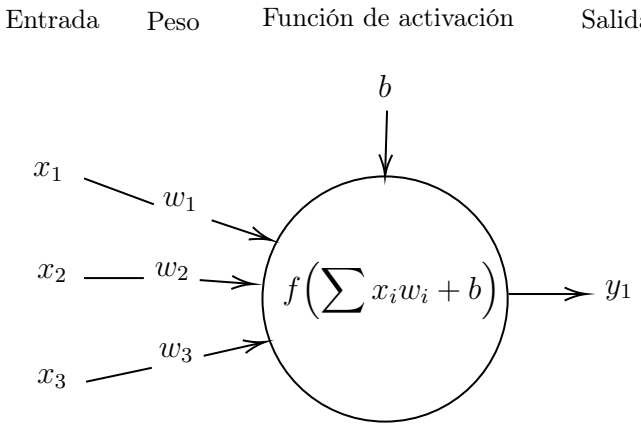


Figura 2.2: Perceptrón o neurona. Elaboración propia.

Sin embargo, las capacidades de cálculo de una neurona son limitadas. Para contrarrestar este problema, se recurre a una estructura de red neuronal conocida como perceptrón multicapa (MLP, por sus siglas en inglés) o *feed-forward* y es el parteaguas del aprendizaje profundo -*deep learning*, por sus siglas en inglés-.

Una red neuronal cuenta con una capa de entrada, una o más capas ocultas

[41]Tomado de Campos Soberanis, Mario (2018): https://medium.com/soldai/inspiraci%C3%B3n-biol%C3%B3gica-de-las-redes-neuronales-artificiales-9af7d7b906a (Última visita 4 de febrero de 2021).

y una capa de salida (ver Figura 2.3). En el *feed-forward* el procesamiento de la información va de capa en capa en una sola dirección, de modo que cada nodo de la capa de entrada se conecta con todos los nodos de la capa oculta y, éstas a su vez se conectan con los nodos de la siguiente capa hasta llegar a la capa de salida.

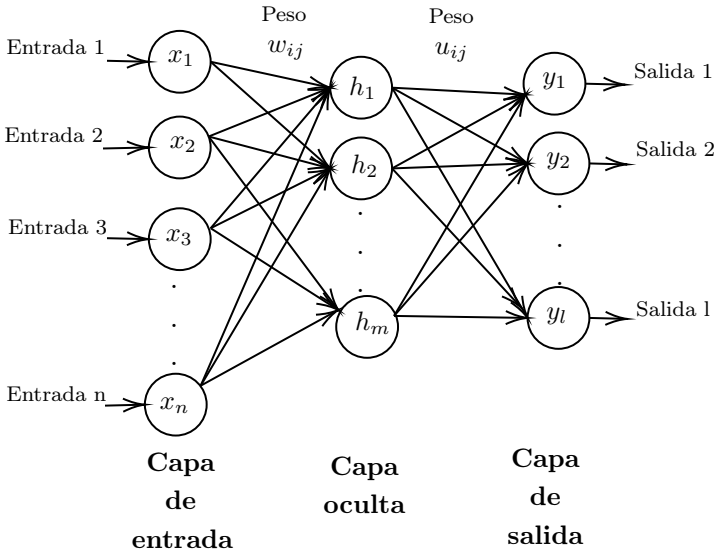


Figura 2.3: Perceptrón multicapa. Elaboración propia.

De Koehn (2017), los componentes de la red neuronal se pueden escribir en términos de vectores y matrices de la siguiente manera:

- Un vector de nodos de entrada $\vec{x} = (x_1, x_2, x_3, ...x_n)^T$
- Un vector de nodos ocultos $\vec{h} = (h_1, h_2, h_3, ...h_m)^T$
- Un vector de nodos de salida $\vec{y} = (y_1, y_2, y_3, ...y_l)^T$
- Matriz de pesos que conecta los nodos de entrada con los nodos ocultos $W = w_{ij}$
- Matriz de pesos que conecta los nodos ocultos con los nodos de salida $U = u_{ij}$

Entonces, los cálculos para una red con una capa oculta son

$$h_j = \sigma(\sum_i x_i w_{ij}) \tag{2.1}$$

$$y_k = \sigma(\sum_j h_j u_{kj}) \tag{2.2}$$

donde σ es la función de activación no lineal y diferenciable.

De acuerdo con Orozco Camacho (2018), al generalizar los cálculos para una MLP, tenemos que una capa oculta h_1 obtiene su valor utilizando un vector de entrada x, una matriz de pesos W_1, un sesgo b_1 y pasando por una función de activación σ de la siguiente forma:

$$h_1 = \sigma(W_1^T x + b_1) \tag{2.3}$$

de igual manera para la i-ésima capa oculta los valores se actualizan como:

$$h_{i+1} = \sigma(W_{i+1}^T h_i + b_{i+1}) \tag{2.4}$$

Finalmente, suponiendo que hay m capas ocultas la capa de salida obtiene su valor con la siguiente ecuación:

$$\hat{y} = \sigma(W_{m+1}^T h_m + b_{m+1}) \tag{2.5}$$

2.1.1.1. Funciones de activación

Para optimizar el desempeño de una red neuronal, la *función de activación* es la encargada de minimizar o maximizar la función de cada neurona. En la Figura 2.4 se muestran algunas funciones.

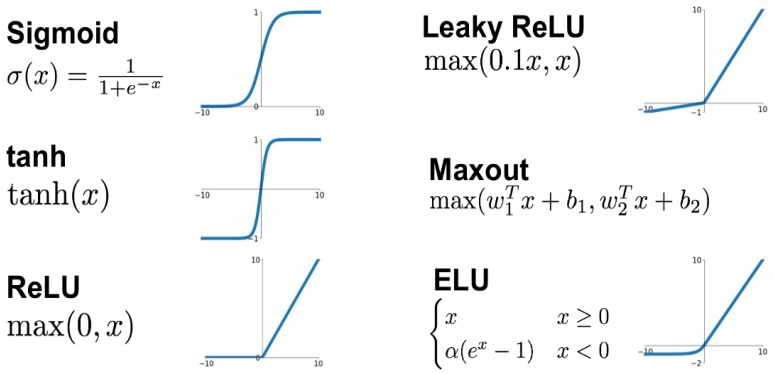


Figura 2.4: Principales funciones de activación no lineales[42] .

[42]Tomado de https://miro.medium.com/max/1192/1*4ZEDRpFuCIpUjNgjDdT2Lg.png (Última visita 6 de febrero de 2021).

2.1.2. Tokenización

Generalmente el primer paso para el entrenamiento de una red neuronal es el proceso de tokenización de un corpus. Donde V es el conjunto de palabras del vocabulario del corpus y $|V|$ el tamaño del vocabulario[43], es decir, el número único de palabras, y cada elemento de V se llama *token* (Jurafsky y Martin, 2019). Cuando un *token* es una palabra se considera a nivel de palabra, nivel de caracteres si es un caracter, también existe la segmentación por subpalabras, donde se separan algunas palabras en pequeñas secuencias de cadenas que permite generar combinaciones no vistas en el corpus (Bragagnini Mendizabal, 2019).

2.1.2.1. Tokenización por subpalabras

Un problema en PLN es el tratamiento de palabras desconocidas, debido a que en la traducción se presenta el problema de vocabulario abierto. Los modelos a nivel de palabra no pueden traducir o generar palabras invisibles. Este problema se resuelve dividiendo algunas palabras pequeñas en unidades de subpalabras (Sennrich et al., 2016), lo que permite generar combinaciones no vistas en el corpus (ver Cuadro 2.1). Esto se logra aplicando el algoritmo Byte Pair Encoding (BPE) en el corpus paralelo. BPE es una técnica de compresión de datos que reemplaza iterativamente el par de bytes más frecuente en una secuencia con un solo byte no utilizado, el algoritmo es adaptado para la segmentación de palabras. Asimismo, en Sennrich y Zhang (2019) se menciona que optimizar el número de códigos BPE se puede mejorar significativamente los resultados para lenguas de escasos recursos.

Frase	jadu'unds ajxy jim nyageekë, mënit ajxy jya wi y'aijxy jya wi y'aijxy, kap ja tyeedy wyiin jyëjp ma kyëxë'ëgy tëëdëm ja tyeedy n'ajty je'e nyëkxnë ma tyëjkën, jim ja y'uung ajxy nyageeky yuuk pa'k, perë ja yayëna'k nyajuëëby je'e nepy je'e wyambida'any, mënit ja y'utsy nyëmaay utsy jam ma tëjkën, n'ajuëëyëptsy ja tu'u maa ajxy të nminëyëm
Tokenización por subpalabra	jadu'un@@ ds ajxy jim nya@@ gee@@ kë, mënit ajxy jya wi y'a@@ ij@@ xy jya wi y'a@@ ij@@ xy, kap ja tye@@ edy wyiin jyë@@ jp ma kyë@@ xë'ë@@ gy të@@ ëdë@@ m ja tye@@ edy n'ajty je'e nyëkx@@ në ma tyëjkën, jim ja y'uung ajxy nya@@ gee@@ ky yuuk pa'@@ k, perë ja ya@@ yë@@ n@@ a'k nyaj@@ u@@ ë@@ by je'e nepy je'e wya@@ mb@@ id@@ a'any, mënit ja y'@@ u@@ tsy nyëmaay u@@ tsy ja@@ m ma të@@ jkën, n'aj@@ u@@ ë@@ yë@@ p@@ tsy ja t@@ u'u maa ajxy të n@@ min@@ ë@@ yëm

Cuadro 2.1: Ejemplo de tokenización por subpalabras; frases del *Ayuuk* de Güichicovi.

[43]Dependiendo la tarea destinada, un vocabulario puede incluir números, caracteres especiales, signos de puntuación, etcétera.

2.1.3. Vectorización

Después de realizar la tokenización, cada *token* de V es representado en forma de vector de dimensión N[44], llamado vector *one-hot*. Donde cada posición del vector es independiente entre sí y representa a cada una de las palabras del vocabulario (ver Figura 2.5). Sin embargo, necesita de una alta dimensionalidad para representar palabras.

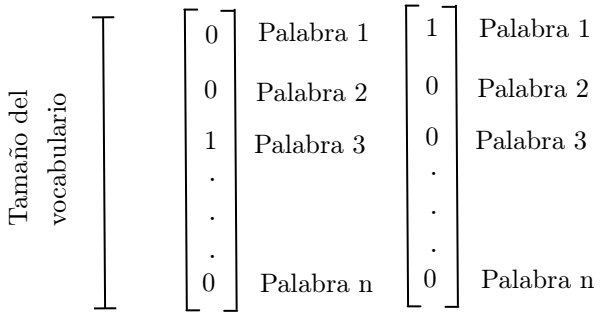


Figura 2.5: Vector one-hot. Elaboración propia.

El problema de la dimensionalidad se resuelve con *word embeddings*, que obtiene un vector más corto y menos disperso (ver Figura 2.6). Un *embedding* de palabras es la representación de un texto donde palabras con significado similar tienen una representación vectorial similar; es decir, el embedding de palabras se obtiene a partir de las palabras que aparecen a su alrededor y comparten características similares (Goldberg, 2017).

Actualmente existen diversos modelos preentrenados como Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), Fasttext (Joulin et al., 2016), (Bojanowski et al., 2017), ELMo (Peters et al., 2018) y BERT (Devlin et al., 2019).

2.1.4. Propagación hacia atrás (Backpropagation)

Sabemos que una red *feed-forward*, básicamente recibe una entrada x en la primera capa, que se propaga hacia adelante hasta llegar a la capa de salidas para producir un resultado $\hat{y}$. Posteriormente, esta salida se evalúa con los valores reales y con el fin de calcular la pérdida/error E. Para minimizar el error de salida es necesario obtener una combinación correcta de pesos en la

[44]La dimensión del vector es la misma que el tamaño del vocabulario.

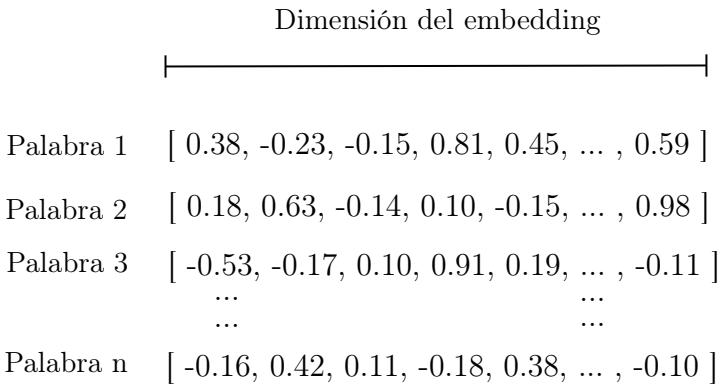


Figura 2.6: Ejemplo de *word embeddings*. Elaboración propia.

red neuronal, esto se logra con el algoritmo de retropropagación (backpropagation en inglés) del error a través de la red, calculando las derivadas parciales de cada uno de los parámetros con respecto al error para obtener el vector gradiente. Entonces, se usa el *descenso del gradiente* (Ruder, 2016) para minimizar iterativamente el error de salida. La actualización de los pesos w^{t+1} se genera con la siguiente expresión:

$$w^{t+1} = w^t - \eta \frac{\partial E}{\partial w} \tag{2.6}$$

Donde w^t es el peso de la actual iteración, η (learning rate) indica el tamaño del paso, es decir, cuánto va a descender y $\frac{\partial E}{\partial w}$ son las derivadas parciales del error respecto a todos los pesos de la red (Mota García, 2001).

A continuación se explica el algoritmo de propagación hacia atrás con una red simple con L capas y con una neurona en cada capa (ver Figura 2.7).

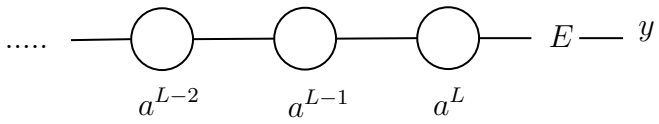


Figura 2.7: Red neuronal simple con L capas. Adaptado de (3Blue1BrownEspañol, 2020).

Primero nos enfocamos en la función de error E respecto a la última capa de la red, ya que este paso nos indica la responsabilidad que tiene esta neurona sobre el resultado final. Esta neurona está compuesta por un peso w multiplicado por la activación de la neurona anterior a^{L-1} y un sesgo b, expresado como:

$$z^L = w^L a^{L-1} + b^L \tag{2.7}$$

Después, Z^L es pasado por una función de activación $a((z^L))$, pero como el objetivo de la propagación hacia atrás es determinar cómo varía el error E ante un cambio de los parámetros w y b, entonces a es evaluada por la función de error $E(a(z^L))$ para determinar el error de la red. En la Figura 2.8 se muestran todos los parámetros que se toman en cuenta. Visto lo anterior, la derivada de la función compuesta $E(a(z^L))$ se resuelve con la regla de la cadena.

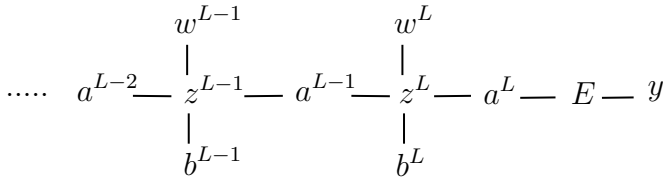


Figura 2.8: Red neuronal simple con L capas, detallado. Adaptado de (3Blue1BrownEspañol, 2020).

Entonces, para pasar de la función de error a la capa L de la red, se debe realizar la derivada de la función de error E respecto al peso w y respecto al sesgo b, representados en las ecuaciones 3.8 y 2.9, respectivamente (Dot-CSV, 2018).

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial w^L} \tag{2.8}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial b^L} \tag{2.9}$$

Donde la derivada de la función de activación respecto al error de la red $\frac{\partial E}{\partial a^L}$, se realiza con la función del error cuadrático medio.

$$E(a_j^L) = \frac{1}{2}\sum_j (y_j - a_j^L)^2 \tag{2.10}$$

$$\frac{\partial E}{\partial a^L} = (a_j^L - y_j) \tag{2.11}$$

Derivada de la función de activación respecto a z^L

$$\frac{\partial a^L}{\partial z^L} = a'(z^L) \tag{2.12}$$

Derivada de z^L respecto a la suma ponderada

$$\frac{\partial z^L}{\partial b^L} = 1 \tag{2.13}$$

$$\frac{\partial z^L}{\partial w^L} = a^{L-1} \tag{2.14}$$

Para terminar con el procedimiento de pasar de la función de error a la capa L, se sustituyen las derivadas en las ecuaciones 3.15 y 3.17, quedando como:

$$\frac{\partial E}{\partial w^L} = \underbrace{\frac{\partial E}{\partial a^L}\frac{\partial a^L}{\partial z^L}}_{\delta^L}\frac{\partial z^L}{\partial w^L} \tag{2.15}$$

$$\frac{\partial E}{\partial w^L} = \delta^L a^{L-1} \tag{2.16}$$

$$\frac{\partial E}{\partial b^L} = \underbrace{\frac{\partial E}{\partial a^L}\frac{\partial a^L}{\partial z^L}}_{\delta^L}\frac{\partial z^L}{\partial b^L} \tag{2.17}$$

$$\frac{\partial E}{\partial b^L} = \delta^L \tag{2.18}$$

Ahora, a partir de la capa $L-1$ se puede generalizar la retropropagación donde nuevamente tenemos:

$$z^{L-1} = w^{L-1}a^{L-2} + b^{L-1} \tag{2.19}$$

y la función de error como $E(a^L(w^L a^{L-1}(w^{L-1}a^{L-2}+b^{L-1})+b^L))$, entonces las derivadas de la función de error E respecto a los parámetros w y b de la capa $L-1$, se escriben de la siguiente manera:

$$\frac{\partial E}{\partial w^{L-1}} = \underbrace{\underbrace{\frac{\partial E}{\partial a^L}}_{\delta^L}\underbrace{\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial a^{L-1}}}_{w^L}\frac{\partial a^{L-1}}{\partial z^{L-1}}}_{\delta^{L-1}}\underbrace{\frac{\partial z^{L-1}}{\partial w^{L-1}}}_{a^{L-2}} \tag{2.20}$$

$$\frac{\partial E}{\partial w^{L-1}} = \delta^{L-1}a^{L-2} \tag{2.21}$$

$$\frac{\partial E}{\partial b^{L-1}} = \underbrace{\underbrace{\frac{\partial E}{\partial a^L}}_{\delta^L}\underbrace{\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial a^{L-1}}}_{w^L}\frac{\partial a^{L-1}}{\partial z^{L-1}}}_{\delta^{L-1}}\underbrace{\frac{\partial z^{L-1}}{\partial b^{L-1}}}_{1} \tag{2.22}$$

$$\frac{\partial E}{\partial b^{L-1}} = \delta^{L-1} \tag{2.23}$$

Teniendo esto, ya se puede retropropagar a todas las capas anteriores de la red neuronal, por lo que, el algoritmo de backpropagation necesita hacer cuatro derivadas parciales (Dot-CSV, 2018):

1. Calcular el error de la última capa $\delta^L = \frac{\partial E}{\partial a^L}\frac{\partial a^L}{\partial z^L}$
2. Retropropagar el error a la capa anterior $\delta^{L-1} = w^L\delta^L\frac{\partial a^{L-1}}{\partial z^{L-1}}$
3. Calcular las derivadas parciales respecto a los parámetros $\frac{\partial E}{\partial w^{L-1}} = \delta^{L-1}a^{L-2}$ y $\frac{\partial E}{\partial b^{L-1}} = \delta^{L-1}$

2.1.4.1. Lotes y épocas

Las entradas que recibe una red neuronal, es un conjunto de muestras que se recorre varias veces hasta llegar a la última muestra del lote (*batch*). Al final de cada lote, los resultados de las predicciones se comparan con las salidas esperadas, y con esto se calculan los gradientes usando una función de error con respecto a los valores actuales (Álvarez et al., 2019). A partir de este error el algoritmo de backpropagation se utiliza para mejorar el modelo. Por otro lado, en una época se recorren todas las muestras de un corpus de entrenamiento, es decir, está compuesta por uno o más lotes. Dado lo anterior, N es el conjunto de datos de entrenamiento, donde N se divide en P subconjuntos disjuntos de tamaño B, llamados lotes. Cada lote tiene un tamaño constante (ver Figura 2.9).

Finalmente, como ejemplo supongamos que tenemos un conjunto de datos con N igual a 6,000 muestras, un tamaño de lote ($batch\_size$) de 30 y 10 épocas.

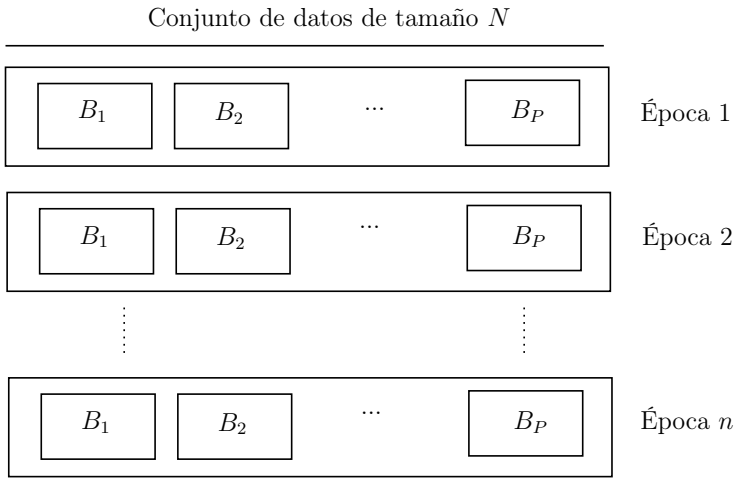


Figura 2.9: Ejemplo de lotes y épocas. Elaboración propia.

Esto significa que N será dividido en 200 lotes, cada uno con 30 muestras. Asimismo, los parámetros de la red se actualizarán por cada lote de 30 muestras y cada época tendrá 200 actualizaciones.

2.1.5. Red Neuronal Recurrente

Una Red Neuronal Recurrente (RNN, por sus siglas en inglés), es un modelo de aprendizaje que recibe una secuencia de vectores $x^{(t)}$ en un tiempo t y cada vector es $x^{(1)}, ..., x^{(\tau)}$ donde τ puede ser de longitud variable, además posee un ciclo en una capa oculta h que funciona como una memoria interna que preserva un estado del tiempo $t-1$. El estado oculto h_t es generado a partir de estados ocultos anteriores, en la Figura 2.10 se observa que los parámetros de la memoria de una RNN se almacenan en una matriz de pesos W y se comparten a través del tiempo t conforme la red va procesando la secuencia de entradas x, que típicamente corresponde a una cadena de eventos.

De la Figura 2.10, x son los datos de entrada, h el estado oculto, o los valores de salida y L la función de pérdida. Una pérdida L mide qué tan lejos está cada o del objetivo y correspondiente. La pérdida L se calcula internamente como $\hat{y} = softmax(o)$ y se compara con el objetivo y. Asimismo, la RNN tiene conexiones de la capa de entrada-oculta parametrizadas por una matriz U, conexiones recurrentes de la capa oculta-oculta por una matriz de peso W y conexiones de la capa oculta-salida parametrizadas por una matriz de peso V. A continuación se muestran las ecuaciones de propagación hacia adelante para la RNN.

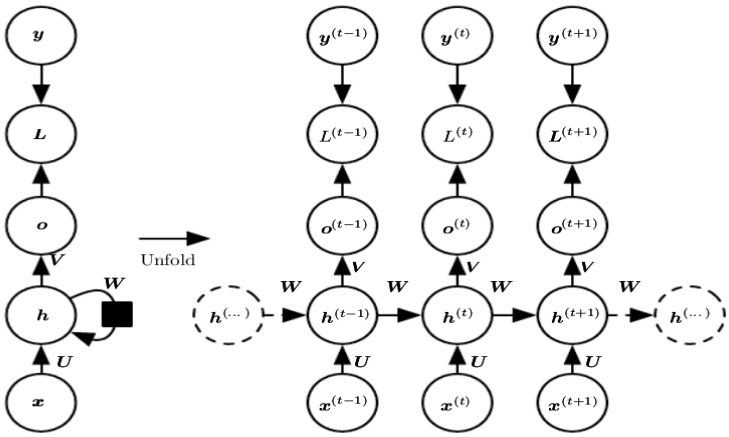


Figura 2.10: Arquitectura desglosada de una red neuronal recurrente. Tomado de (Goodfellow et al., 2016, p. 378).

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \tag{2.24}$$

$$h^{(t)} = tanh(a^{(t)}) \tag{2.25}$$

$$o^{(t)} = c + Vh^{(t)} \tag{2.26}$$

$$\hat{y} = softmax(o^{(t)}) \tag{2.27}$$

Donde $a^{(t)}$ es una variable auxiliar, $h^{(t-1)}$ es el estado oculto anterior, $h^{(t)}$ el estado oculto en el tiempo t y al que se le aplica la tangente hiperbólica como función de activación, $x^{(t)}$ es el vector de entrada, b y c son los vectores de sesgo/bias, o la salida de red, $\hat{y}$ la predicción de la red aplicando la función $softmax$.

El cálculo de la pérdida total para una secuencia dada de valores de x emparejados con una secuencia de valores de y, sería entonces la suma de las pérdidas en todos los pasos de tiempo, donde $L^{(t)}$ es la probabilidad logarítmica de $y^{(t)}$ dado $x^{(1)}, ..., x^{(t)}$, $p_{modelo}\left(y^{(t)} | \left\{x^{(1)}, ..., x^{(t)}\right\}\right)$ se obtiene leyendo la entrada para $y^{(t)}$ del vector de salida del modelo $\hat{y}^{(t)}$. Esto se expresa con la siguiente ecuación:

$$L\left(\left\{x^{(1)}, ..., x^{(\tau)}\right\}, \left\{y^{(1)}, ..., y^{(\tau)}\right\}\right) = \sum_t L^{(t)} \tag{2.28}$$

$$= -\sum_t log\ p_{modelo}\left(y^{(t)} | \left\{x^{(1)}, ..., x^{(t)}\right\}\right) \tag{2.29}$$

El cálculo del gradiente de una función de pérdida de una RNN se realiza con el algoritmo de retropropagación en el tiempo (BPTT, por sus siglas en inglés).

2.1.5.1. LSTM

A largo plazo las redes neuronales recurrentes se vuelven incapaces de preservar la información pasada. LSTM (Long Short-Term Memory, por sus siglas en inglés) tiene como propósito resolver el problema de la pérdida de memoria a largo plazo en una RNN (Hochreiter y Schmidhuber, 1997). De acuerdo con Goodfellow et al. (2016) en la arquitectura LSTM las celdas de memoria preservan la información, así como los gradientes de error a lo largo del tiempo y se controlan mediante funciones que simulan compuertas lógicas (ver Figura 2.11).

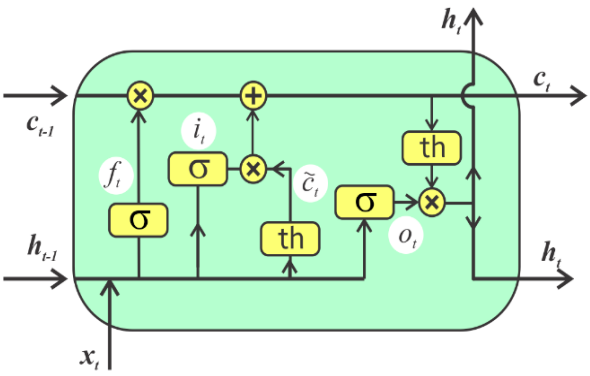


Figura 2.11: Celda de una LSTM [45] .

Matemáticamente una LSTM se representa con las siguientes ecuaciones:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.30}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.31}$$
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.32}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.33}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.34}$$
$$h_t = o_t * tanh(C_t) \tag{2.35}$$

[45]Tomado de Hrnjica, Bahrudin (2019): https://hrnjica.net/2019/04/08/in-depth-lstm-implementation-using-cntk-on-net-platform/ (Última visita 1 de julio de 2021).

Donde el vector f_t es la compuerta de olvido para eliminar la información de contexto que ya no es necesaria, i_t la compuerta de entrada que regula cuánto cambia la entrada nueva al estado de la memoria, o_t la compuerta de salida que decide qué información se requiere para el estado oculto actual, $\tilde{C}$ vector con nuevos valores candidatos que se agregan al estado, c_t estado de la celda y h_t la salida a la siguiente capa.

2.1.5.2. RNN Bidireccionales

Las RNN explicadas anteriormente solo capturan información del pasado en el momento t. Las RNN bidireccionales combinan una RNN que avanza en el tiempo comenzando desde el inicio de la secuencia con otro RNN que retrocede en el tiempo comenzando desde el final de la secuencia (Goodfellow et al., 2016), ver ejemplo en la Figura 2.12.

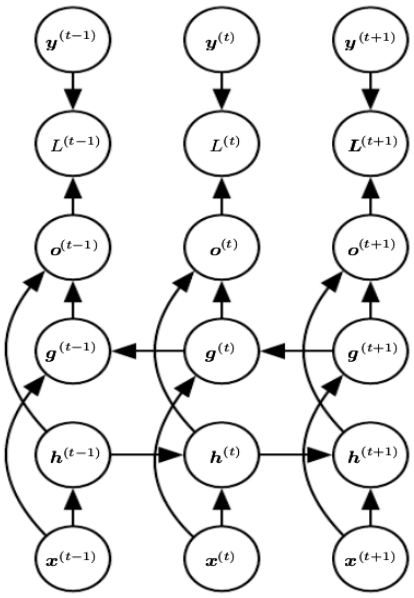


Figura 2.12: Arquitectura de una red neuronal recurrente bidireccional. Tomado de (Goodfellow et al., 2016, p. 395).

2.1.6. Sequence to Sequence

Con el objetivo de resolver el problema de secuencias de entrada y salida que tienen diferentes longitudes con relaciones complicadas, el modelo *sequence to sequence* o *encoding-decoding* fue introducido en el año 2014 por (Sutskever et al., 2014). Este método utiliza una memoria LSTM multicapa para mapear

una secuencia de entrada $x_1, ..., x_T$ a un vector de dimensión fija que representa el significado de una oración, llamado vector de contexto v, y luego una LSTM multicapa para decodificar la secuencia objetiva $y_1, ..., y_{T'}$ del vector v, esto es $p(y_1, ..., y_{T'}|x_1, ..., x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, ..., y_{t-1})$.

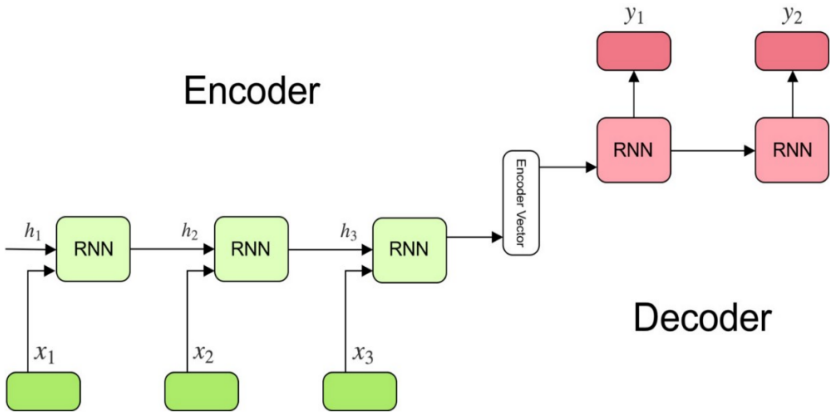


Figura 2.13: Modelo Sequence to Sequence [46] .

De la Figura 2.13, el *codificador* es una RNN multicapa que recopila la información y se propaga hacia adelante, los estados ocultos se calculan con la ecuación 2.36. El vector de contexto comprime la información de todos los elementos de entrada y actúa como el estado oculto inicial del decodificador del modelo. El *decodificador* es una RNN multicapa, donde la salida es generada de forma secuencial, es decir, cada paso genera una salida y_t representada en la ecuación 2.37, la cual será la entrada para el siguiente paso de tiempo t.

$$h_t = sigm(W^{hx}x_t + W^{hh}h_{t-1}) \tag{2.36}$$

$$y_t = W^{yh}h_t \tag{2.37}$$

2.1.7. Encoder-Decoder con Atención

El problema principal del enfoque *codificador-decodificador* es que la red neuronal necesita comprimir toda la información de una sentencia fuente en un vector de longitud fija, vector de contexto. Esto genera un cuello de botella en el vector de contexto, lo que dificulta que la red neuronal no pueda

[46]Tomado de Kostadinov Simeon (2019): https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346 (Última visita 4 de julio de 2021).

lidiar con sentencias largas. Cho, van Merrienboer, Bahdanau, y Bengio (2014) mostraron que el enfoque *codificador-decodificador* funciona relativamente en sentencias cortas sin palabras desconocidas, sin embargo, su rendimiento disminuye a medida que aumenta la longitud de la sentencia de entrada y el número de palabras desconocidas.

Para evitar codificar una sentencia completa en un solo vector de longitud fija, Bahdanau et al. (2014) presentan el modelo *encoder-decoder con atención*. La idea general de este modelo es que para traducir una oración se debe prestar atención a una parte de esta y traducir esa parte, así sucesivamente hasta cubrir toda la oración. Este mecanismo permite al decodificador obtener información de todos los estados ocultos del codificador, cuando el modelo predice una palabra objetivo se basa en los vectores de contexto asociado a las posiciones de origen y todas las palabras de destino generadas anteriormente.

En la Figura 2.14 se muestra un ejemplo del mecanismo de atención. Donde $\mathbf{x} = (x_1, ..., x_{T_x})$ es la sentencia de origen, $\mathbf{y} = (y_1, ..., y_{T_y})$ la sentencia destino y $(h_1, ..., h_{T_x})$ los estados ocultos.

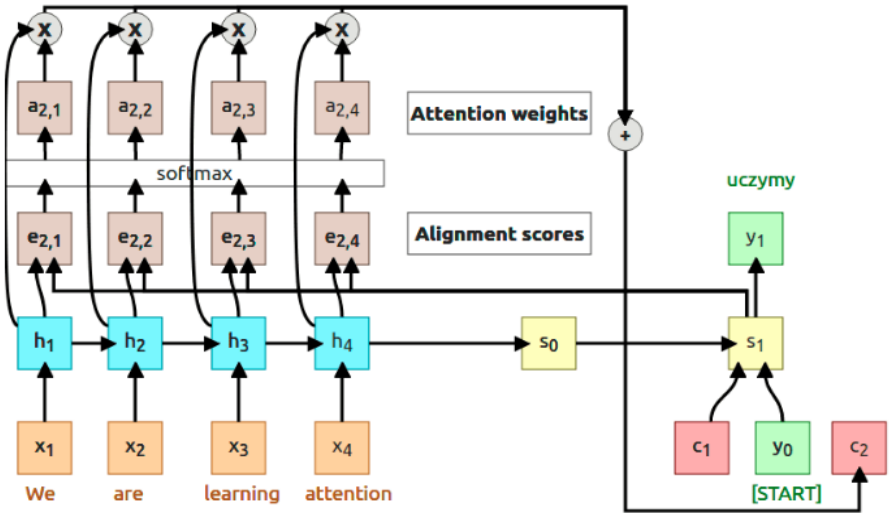


Figura 2.14: Codificador-Decodificador con atención [47] .

Para calcular la salida y_i con el mecanismo de atención se realiza con la siguiente probabilidad condicional (Bahdanau et al., 2014):

[47]Tomado de Erdem Kemal (2021): https://erdem.pl/2021/05/introduction-to-attention-mechanism (Última visita 9 de julio de 2021).

$$p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i) \tag{2.38}$$

Siendo s_i un estado oculto en el tiempo t de una RNN calculada como $s_i = f(s_{i-1}, y_{i-1}, c_i)$, donde el vector de contexto c_i depende de la secuencia $(h_1, ..., h_{T_x})$ que el codificador asigna a la entrada y cada h_i contiene información de cada entrada así como las que la rodean. Entonces, c_i se calcula con la siguiente ecuación:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.39}$$

Siendo α_{ij} el peso de atención,

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.40}$$

y e_{ij} es el puntaje de alineación entre el estado oculto del estado del tiempo anterior s_{i-1} del decodificador y las entradas h_j alrededor de la posición j del codificador.

$$e_{ij} = a(s_{i-1}, h_j) \tag{2.41}$$

2.1.8. Modelo Transformer

Las arquitecturas encoder-decoder con mecanismos de atención dominaban el campo del PLN. Sin embargo, Vaswani et al.(2017) proponen la arquitectura Transformer (ver Figura 2.17). Esta arquitectura se basa únicamente en el mecanismo de *atención* para mapear dependencias globales entre la entrada y la salida, utilizando una estructura codificador-decodificador con múltiples capas de *auto-atención*[48] conectadas entre sí, tanto para el codificador como para el decodificador procesando todo de manera simultánea, evitando la recurrencia. Asimismo, es un modelo altamente paralelizable, superando por mucho a las arquitecturas anteriores en el tiempo de entrenamiento y en la calidad de las traducciones, lo que lo hace el actual estado del arte en modelos de TA.

[48]De acuerdo con Vaswani et al. (2017), la auto-atención es un mecanismo de atención que relaciona diferentes posiciones de una sola secuencia para calcular una representación de la secuencia, es decir, todos los tokens de una secuencia interactúan entre sí.

Vaswani et al. (2017) describen a la función de atención como la asignación de una consulta (q) del decodificador y un conjunto de pares clave-valor (k-v) del codificador a una salida, donde q, k, v y la salida son vectores. Como se vio en la sección 2.14, la salida (Ecuación 2.39) se calcula como una suma ponderada de los valores v con α_{ij}, donde α_{ij} asignada a cada valor se calcula mediante una función de compatibilidad (Ecuación 2.41) de la consulta con la clave correspondiente (ver Figura 2.15).

Intuitivamente, la consulta se puede ver como la que pide la información; la clave es la que tiene la información; y el valor es el que proporciona la información.

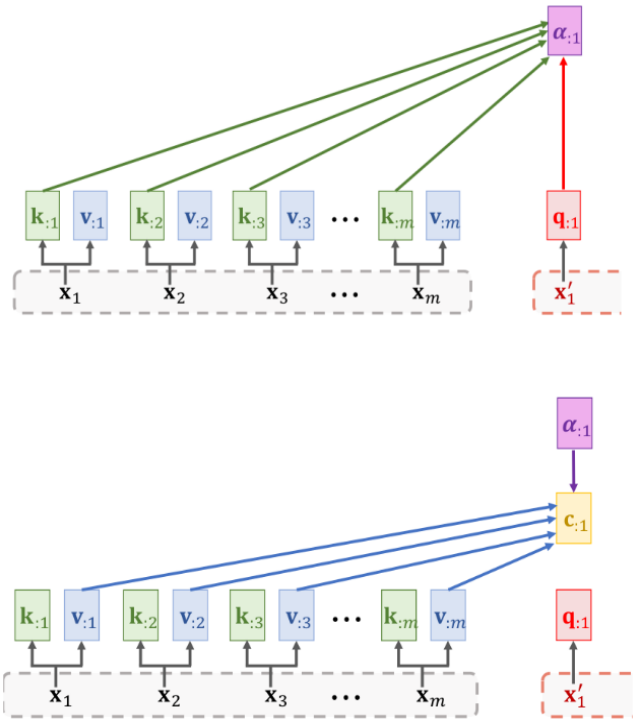


Figura 2.15: Mecanismo de atención de acuerdo con (Vaswani et al., 2017). Tomado de (Wang, s.f.)

El modelo con el mecanismo de atención no puede aprender las diversas formas en que las palabras pueden contribuir a la representación de entradas más largas (Jurafsky y Martin, 2019). Para resolver este problema, Transformer con su mecanismo de *auto-atención* captura los diferentes roles que desempeñan los embeddings de entrada, incluyendo nuevos parámetros en forma de matrices de pesos, estas matrices son W^Q, W^K y W^V (ver Figura 2.16). Entonces

para cada entrada x_i [49], estos pesos son usados para calcular los vectores q_i, k_i y v_i.

$$q_i = W^Q x_i \quad ; \quad k_i = W^K x_i \quad ; \quad v_i = W^V x_i$$

Donde cada embedding de entrada es de tamaño d_m , las matrices de tamaño $d_q \times d_m$, $d_k \times d_m$ y $d_v \times d_m$ respectivamente.

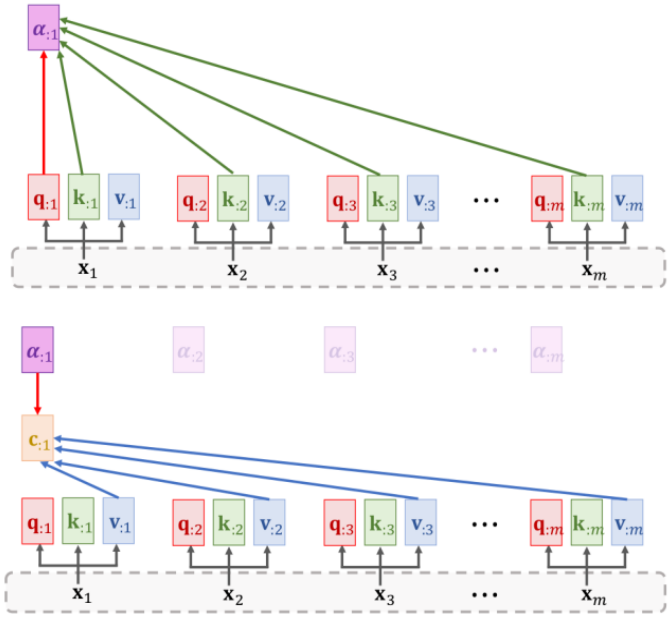


Figura 2.16: Mecanismo de auto-atención. Tomado de (Wang, s.f.)

Dado que se tiene un conjunto de entradas x, consultas, claves y valores, y donde cada salida y_i se calcula de forma independiente, todo este proceso se puede paralelizar mediante multiplicación de matrices, cada una con t columnas y cada vector columna con dimensión d_m, esto es:

$$Q = W^Q X \quad ; \quad K = W^K X \quad ; \quad V = W^V X$$

2.1.9. Codificador de Transformer

La tarea del codificador es mapear todas las secuencias (secuencia de una lengua) de entrada en una capa oculta. Se compone de un codificador posi-

[49]Cada x_i es un vector de dimensión d_m, por lo que un conjunto de t entradas $\{x_i\}_{i=1}^t = \{x_1, ..., x_t\}$ se puede representar como una matriz $X \in \mathbb{R}^{d_m \times t}$. De igual manera dado X se tiene $Q = \{q_i\}_{i=1}^t$; $K = \{k_i\}_{i=1}^t$; $V = \{v_i\}_{i=1}^t$ (Canziani, 2020).

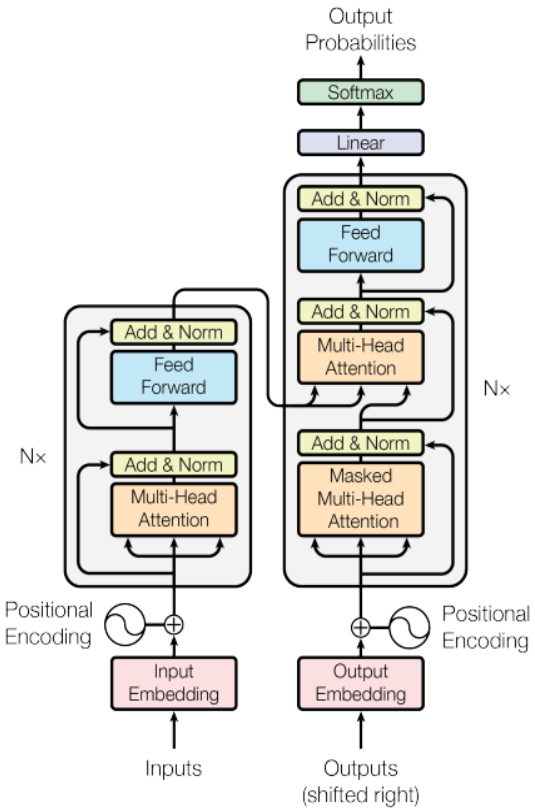


Figura 2.17: Modelo de arquitectura Transformer. Tomado de (Vaswani et al., 2017).

cional; mecanismo de autoatención multicabezal; conexión residual y normalización de capas; feedforward, que es un mapeo lineal que se activa con una función de activación.

2.1.9.1. Codificador posicional (Positional Encoding)

Dado que el modelo Transformer no contiene recurrencia, no cuenta con la información de las posiciones relativas ni absolutas de los tokens de entrada. Para abordar este problema, los embeddings de entrada se combinan[50] con unos codificadores posicionales específicos para cada posición de entrada (ver Figura 2.18). El cálculo de estos codificadores de posición se realiza mediante la combinación de funciones seno y coseno, que posteriormente se concatenan para formar cada uno de los vectores de codificación posicional, donde los tokens estarán más cerca unos de otros basándose en la similitud de su significado

[50]Estos codificadores posicionales tienen las mismas dimensiones que los embeddings de entrada, por lo que se pueden sumar.

y posición en la oración. Entonces para generar cada vector del codificador de posición se realiza con las siguientes ecuaciones:

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.42}$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.43}$$

donde *pos* es la posición, i la dimensión. Cada dimensión de la posición corresponde a una senoide (Vaswani et al., 2017).

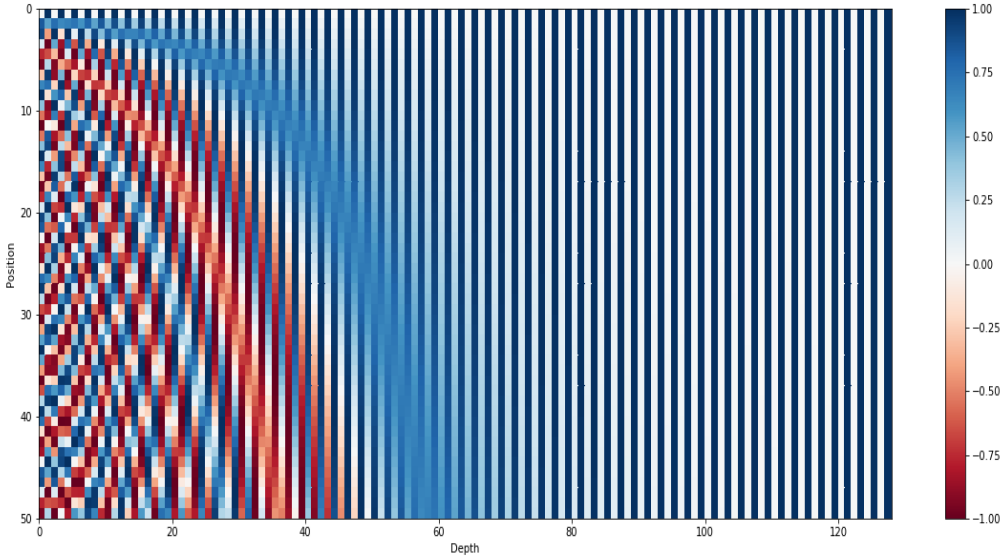


Figura 2.18: Ejemplo de codificación posicional[51] (pos) para 50 palabras con un embedding/dimensión (i) de 128.

2.1.9.2. Multi-Cabezales de atención

Después de realizar el proceso de codificación posicional y generar la matriz de codificación $W \in \mathbb{R}^{n \times d_m}$, estos datos se clonan tres veces que serán Q, K y V. Posteriormente se envían a un módulo de multi-cabezal de atención propia (ver Figura 2.19).

[51]Tomado de Kazemnejad Amirhossein (2019): https://kazemnejad.com/blog/transformer_architecture_positional_encoding/ (Última visita 21 de julio de 2021).

La atención multi-cabezal permite al modelo atender conjuntamente varios aspectos del lenguaje o subespacios de manera paralela. Es decir, en lugar de realizar una única función de atención, es más beneficioso proyectar h veces las consultas, claves y valores, donde h es el número de cabezales de atención. Esta idea se representa como:

$$Z_t = Multihead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.44}$$

donde

$$head_i = Atencion(QW_i^Q, KW_i^K, VW_i^V) \tag{2.45}$$

Donde las proyecciones son matrices de parámetros $W_i^Q \in \mathbb{R}^{d_m \times d_k}$, $W_i^K \in \mathbb{R}^{d_m \times d_k}$, $W_i^V \in \mathbb{R}^{d_m \times d_v}$.

En la Figura 2.19 se observa que dentro del módulo multi-head se realiza el producto punto escalado con las matrices Q, K y V utilizando la siguiente ecuación:

$$Atencion(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.46}$$

Donde d_k es la dimensión de la clave K. La ecuación 2.46 del producto punto escalado se utiliza para generar los valores de atención $\{Z_i\}_{i=1}^t$.

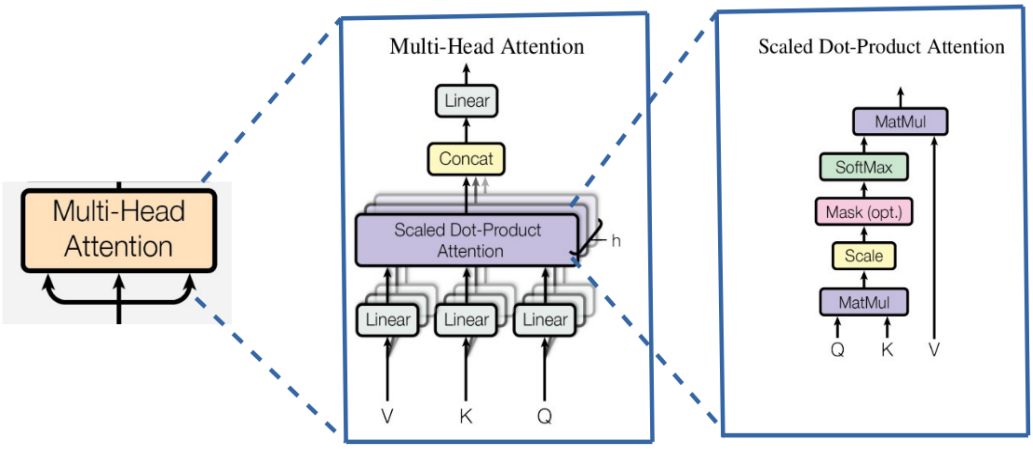


Figura 2.19: Multicapas de la arquitectura transformer. Adaptado de (Vaswani et al., 2017).

En la Figura 2.20 se muestra el proceso interno que se lleva a cabo para

llegar a la matriz Z_t del módulo de multi-head attention.

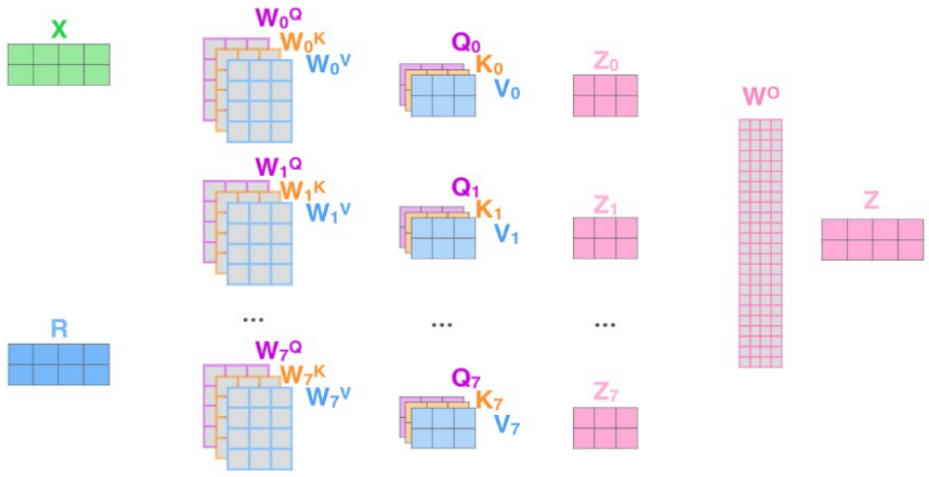


Figura 2.20: Proceso interno del Multi-Head Attention[52] .

Después de pasar por la capa de auto-atención se agrega una capa de normalización, donde al resultado Z_t se suma con X, generando un $Z_t^{'}$ (ecuación 2.47). Posteriormente, con este resultado se alimenta a la red *feed-forward*, donde R denota la salida de la red. Con esto, nuevamente se agrega a una capa de normalización, donde la salida se denota como $Z_t^{''}$ (ecuación 2.48).

$$Z_t^{'} = Z_t + X \tag{2.47}$$

$$Z_t^{''} = Z_t + R \tag{2.48}$$

2.1.10. Decodificador de Transformer

La tarea del decodificador es mapear la capa oculta para generar secuencias de textos. Al igual que el codificador, el decodificador cuenta con un codificador posicional; conexión residual y normalización de capas; un mecanismo de autoatención multicabezal; capa feedforward; cuenta con dos mecanismos de autoatención, multicabezal y multicabezal con enmascaramiento.

[52]Tomado de Jay Alammar (2018): http://jalammar.github.io/illustrated-transformer/ (Última visita 21 de julio de 2021).

2.1.10.1. Multi-cabezales de atención con enmascaramiento

El decodificador contiene un módulo de multi-cabezales de atención con enmascaramiento, donde sus entradas son las incrustaciones de salida desplazadas $\{y_i\}_{i=1}^{n}$ una palabra a la derecha (Ghojogh y Ghodsi, 2020). Asimismo, la codificación posicional también se agrega a las incrustaciones de salida para incluir la información de sus posiciones. En esta parte x_i se reemplaza por y_i.

El módulo multi-cabezal de atención con enmascaramiento es similar al módulo de atención multi-cabezal, pero oculta las palabras siguientes después de una palabra. Es decir, cada palabra solo atiende a sus palabras de salida anteriores.

$$Z_m = MaskedAttention(Q, K, V) \tag{2.49}$$

$$= softmax\left(\frac{QK^T + M}{\sqrt{d_k}}\right)V \tag{2.50}$$

Donde la matriz de enmascaramiento $M \in \mathbb{R}^{n\times n}$ es:

$$M_{ij} = \begin{cases} 0 & si \quad j \leq i \\ \\ -\infty & si \quad j > i \end{cases} \tag{2.51}$$

Dado que la función softmax tiene un operador exponencial, el enmascaramiento no tiene ningún impacto para $j \leq i$ [53] ni para $j > i$ [54]. Asimismo, en términos de posición en la secuencia corresponden a las palabras anterior y siguiente, respectivamente (Ghojogh y Ghodsi, 2020).

Después de pasar por el módulo de enmascaramiento, el procedimiento del decodificador es similar al codificar explicado anteriormente. Finalmente, todo este proceso de *codificador-decodificador* pasa por una capa lineal y con una función softmax se determinan las salidas y_i iterativamente hasta completar el entrenamiento.

[53] $e^0 = 1$
[54] $e^{-\infty} = 0$

2.2. JoeyNMT

Desde la llegada de la NMT, varios grupos de investigación de la industria han desarrollado herramientas de código abierto especializados para NMT como OpenNMT (Klein et al., 2018) y Neural Monkey (Helcl y Libovický, 2017). Sin embargo, estas herramientas han sido diseñadas para investigadores con formaciones sólidas en TA y aprendizaje profundo.

JoeyNMT[55] es una herramienta de NMT para fines educativos basada en PyTorch[56], y tiene como objetivo facilitar la comprensión y accesibilidad para principiantes en la traducción neuronal (Kreutzer et al., 2019). Esta herramienta proporciona muchas funciones populares de la NMT en una base de código pequeña y simple, entre sus características principales están la admisión de arquitecturas clásicas como RNN's y Transformers; codificador-decodificador configurable; mecanismos de atención; manejo de entrada basado en palabras, BPE y caracteres; evaluación BLEU; inicialización personalizable; herramientas de visualización y monitoreo de la curva de aprendizaje, entre otras funciones. Asimismo, aunque esta herramienta es de menor tamaño, logra un rendimiento comparable a las herramientas de última generación más complejas.

2.3. Alineadores

El primer paso para construir un sistema de traducción automática es conseguir textos alineados, ya sea por palabra o por frases, pero hay veces que las traducciones vienen en archivos[57] extensos, lo que tomaría mucho tiempo buscar las correspondencias de manera manual. Es ahí donde se requiere de las alineaciones automáticas. En SMT un alineado se puede ver como un subproducto de un *modelo de traducción*.

2.3.1. Alineación por palabra

A principios de los 90's, Brown et al. (1990) introducen la idea de una *alineación* entre un par de cadenas del francés al inglés, donde cada palabra

[55]Esta herramienta está disponible en https://github.com/joeynmt/joeynmt.

[56]PyTorch es un paquete de Python que se usa como un reemplazo de NumPy para usar la potencia de las GPU en el cálculo de tensores. Disponible en https://github.com/pytorch/pytorch.

[57]Dos archivos; una con textos que contienen la lengua origen y otro que contiene textos de la lengua destino.

origen f de una oración en francés corresponde a una palabra destino e de una oración en inglés.

Brown et al. (1993) presentan una serie de modelos de alineación con una complejidad creciente, es decir, que cada modelo IBM en específico incluye todos los modelo anteriores y agrega un nivel de complejidad. Estos modelos son: IBM-1, IBM-2, IBM-3, IBM-4 e IBM-5. En esta sección la explicación de los modelos se basa principalmente en (Koehn, 2010), (Mager Hois, 2017) y (Siebecke y Arvidson, 2016)

IBM-1 se basa principalmente en probabilidades de traducción léxica (ver Figura 2.21). Este modelo busca la probabilidad de que una palabra en una lengua origen se traduzca a una palabra de una lengua meta. Sin embargo, IBM-1 no puede reordenar, agregar o eliminar palabras de una oración. Tampoco toma en cuenta la *fertilidad*, es decir, palabras con múltiples traducciones.

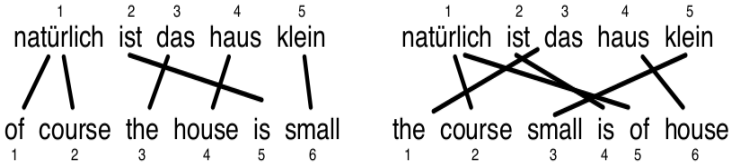


Figura 2.21: IBM-1. Tomado de (Koehn, 2010).

El modelo se describe con la siguiente ecuación.

$$p(e, a|f) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \tag{2.52}$$

Donde:

$f = (f_1, ..., f_{l_f})$ de longitud l_f
$e = (e_1, ..., e_{l_e})$ de longitud l_e
$a(j) =$ posición i de la alineación de e_j con f_i
$\epsilon =$ es una constante de normalización

IBM-2 es una extensión del modelo anterior al agregar un modelo de alineación posicional $a(i|j, l_e, l_f)$, donde $a(j)$ es la posición f_i con el que e_j está alineado.

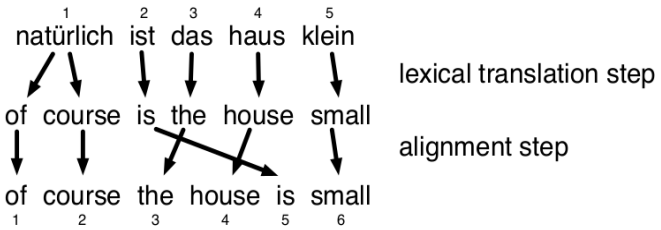


Figura 2.22: IBM-2. Tomado de (Koehn, 2010).

Por lo tanto, el nuevo modelo se expresa con la ecuación:

$$p(e, a|f) = \epsilon \prod_{j=1}^{l_e} t(e_j|f_{a_j}) a(a(j)|j, l_e, l_f) \tag{2.53}$$

Para resolver el problema de que algunas palabras de la lengua origen tienden a alinearse con varias palabras de la lengua destino o no estar alineadas a ninguna, **IBM-3** agrega un modelo de *fertilidad* expresado como $n(\phi|f)$, donde ϕ indica a cuántas palabras se traduce la palabra origen f. De igual manera, las palabras que no tienen traducción se alinean con el token NULL y $\phi = 0$. La fertilidad se utiliza antes de los pasos de inserción y traducción léxica.

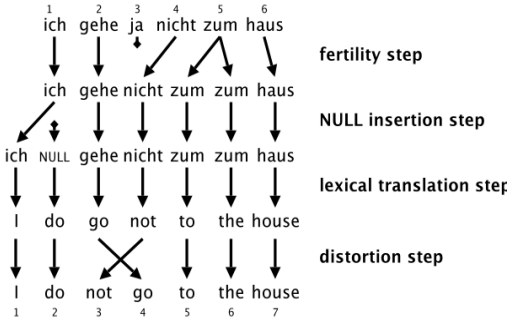


Figura 2.23: IBM-3. Tomado de (Koehn, 2010).

El modelo de IBM-3 se expresa con la siguiente ecuación (Mager Hois, 2017)

$$p(e|f) = \sum_{a} p(e, a|f) \tag{2.54}$$

$$= \sum_{a(1)=0}^{l_f} \sum_{a(l_e)=0}^{l_f} \binom{l_e - \phi_0}{\phi_0} p_1^{\phi_0} p_0^{l_e - 2\phi_0} \prod_{i=1}^{l_f} \phi_i! n(\phi_i|f_i) * \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) a(a(j)|j, l_e, l_f)$$

En los modelos de traducción anteriores, las frases de la lengua origen que se traducen a una frase en la lengua destino se reordenan como unidades, es decir, cada palabra se mueve de manera independiente y generan salidas dispersas. **IBM-4** mediante el uso de dos modelos de distorsión (ver 2.55 y 2.56) basados en septos $\odot$, se resuelve el problema de reordenamiento causado por la palabra previamente alineada. De modo que, el resultado es un modelo lexicalizado (Siebecke y Arvidson, 2016).

Sea

$$d_1(j - \odot_{[i-1]}|\mathcal{A}(f_{[i-1]}), \mathcal{B}(e_j)) \tag{2.55}$$

la probabilidad de asociación con la posición de la primera palabra y

$$d_{>1}(j - \Pi_{i,k-1}|\mathcal{B}(e_j)) \tag{2.56}$$

la distribución de probabilidad para las palabras subsecuentes.

Donde π son todas las palabras e alineadas a f y son llamados septos, el centro septo $\odot$ es la media de las posiciones de e, el operador $[i]$ mapea al septo con índice i a la posición que corresponde en f, $\pi_{i,k-1}$ es la k-ésima palabra en el i-ésimo septo. Asimismo, $\mathcal{A}$ y $\mathcal{B}$ son funciones sobre los vocabularios de origen y destino respectivamente, donde agrupan una cantidad de palabras en clases (Brunning, 2010).

El modelo anterior presenta deficiencias como otorgar probabilidades positivas a traducciones no válidas, así como colocar varias palabras de salida en el mismo lugar. **IBM-5** elimina estas deficiencias a costa de complicar el modelo añadiendo más parámetros que dificultan más el entrenamiento. En cuestiones de rendimiento, la alineación no es óptima. Por ende, IBM-4 es considerado como el modelo de alineación estándar.

De acuerdo con Och y Ney (2004) la alineación por palabras se puede mejorar mediante *heurísticas* de simetrización de crecimiento. Esta técnica combina alineaciones de ambas direcciones de traducción, es decir, lenguaje origen a lenguaje destino y viceversa. Entonces, para cada par de oraciones se calculan las alineaciones a_1^J y b_1^I. Donde $A_1 = \{(a_j, j)|a_j > 0\}$ y $A_2 = \{(i, b_i)|b_i > 0\}$ son dos conjuntos de alineaciones. Se combina A_1 y A_2 en una matriz A de alinea-

ción, ya sea por intersección[58] o unión[59].

Se propone un método de refinado para una alineación simétrica, donde el primer paso es realizar la intersección entre A_1 y A_2 que genere una alta precisión (ver Figura 2.24). Después, se extiende la alineación A con vecinos de enlace agregando alineaciones (i, j) que se toman del conjunto de unión. Los buenos enlaces suelen estar junto a las alineaciones que se intersectan, y esto lleva a varias heurísticas que utilizan vecinos para extender la intersección (ver Figura 2.25); es decir, la alineación (i, j) tiene un vecino horizontal $(i-1, j)$, $(i+1, j)$ o un vecino vertical $(i, j-1)$, $(i, j+1)$ o posiblemente un vecino diagonal que ya está en A.

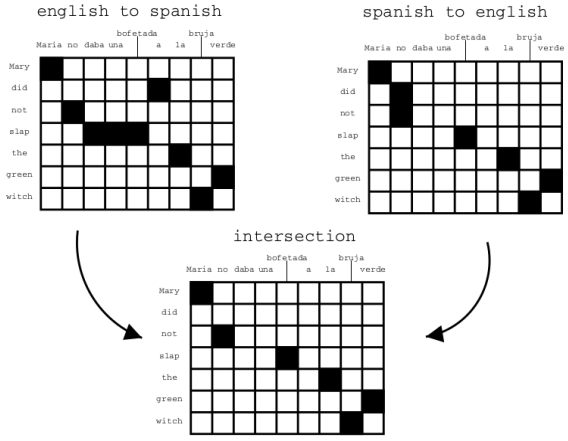


Figura 2.24: Intersección de una alineación bidireccional. Tomado de (Koehn y Knight, 2003).

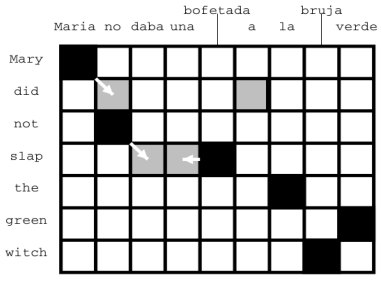


Figura 2.25: Método de refinado. Tomado de (Koehn y Knight, 2003).

[58] $A = A_1 \cap A_2$ genera una alto nivel de precisión.
[59] $A = A_1 \cup A_2$ genera una alto nivel de recuperación.

2.3.2. Alineación por frases

Una de las desventajas de la alineación por palabras es la omisión de la información contextual, así como la limitación de solamente usar relaciones de uno a muchos. Entonces, una forma de incorporar el contexto de la información es por medio de la traducción por frases en lugar de palabras.

La alineación por frases deriva de la alineación por palabras, por lo que el método de *extracción de frases* retoma la matriz simétrica A de la alineación por palabras (ver Figura 2.25). El criterio que define al conjunto de frases es la ecuación:

$$\mathcal{BP}(f_1^J, e_1^I, A) = \left\{ \left(f_j^{j+m}, e_i^{i+n} \right) : \forall (i', j') \in A : j \leq j' \leq j+m \leftrightarrow i \leq i' \leq i+n \right\} \tag{2.57}$$

Donde una frase bilingüe es un par de m palabras origen y n palabras destino. Zens et al. (2002) mencionan que para la extracción de frases del par de oraciones $(f_1^J; e_1^I)$ dentro de una alineación por palabras, debe cumplir que las palabras sean consecutivas y sean consistentes con la matriz de alineación de palabras, es decir, que las m palabras de origen se alinean con las n palabras de destino y viceversa.

Matricialmente esto se representa como la Figura 2.26:

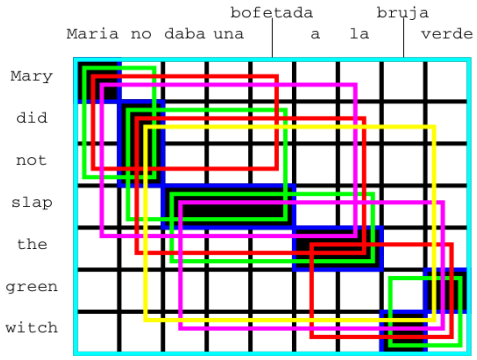


Figura 2.26: Extracción de frases. Tomado de (Koehn y Knight, 2003).

(María, Mary), (no, did not),(slap, daba una bofetada) (a la, the), (bruja, witch), (verde, green)
(María no, Mary did not), (no daba una bofetada, did not slap) (daba una bofetada a la, slap the), (bruja verde, green witch)
(María no daba una bofetada, Mary did not slap), (no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch)
(María no daba una bofetada a la, Mary did not slap the), (daba una bofetada a la bruja verde, slap the green witch)
(No daba una bofetada a la bruja verde, did not slap the green witch)
(María no daba una bofetada a la bruja verde, Mary did not slap the green witch)

Cuadro 2.2: Extracción de frases. Adaptado de (Koehn y Knight, 2003).

2.3.2.1. YASA

Actualmente hay muchos alineadores por frases disponibles, cada una con sus ventajas y desventajas. Algunos de estos alineadores son: hunalign – sentence aligner[60], Yet Another Sentence aligner-YASA[61], BlueAlign[62], por mencionar algunos. En este trabajo se utilizó el alineador YASA (Lamraoui y Langlais, 2013).

YASA es un programa bajo licencia *Apache License, Version 2.0*[63] que alinea dos archivos de diferentes lenguas para producir un bitexto. Maneja muchos formatos de archivos de entrada y salida. Este alineador opera un proceso de dos pasos a través de datos en paralelo. Primero identifica los cognados[64] para conseguir una primera alineación a nivel token que delimite un espacio de búsqueda fructífero. Posteriormente, la alineación de la oración se realiza en este espacio de búsqueda limitado. Asimismo, YASA, como la mayoría de los alineadores, asume una alineación monótona/uniforme.

La idea de la reducción del espacio viene de que una alineación de oraciones se puede lograr de manera eficiente utilizando una alineación rústica a nivel de palabra. Esta noción tiene muchas representaciones que son complejas de

[60] http://mokk.bme.hu/en/resources/hunalign/
[61] https://github.com/anoidgit/yasa
[62] https://github.com/rsennrich/Bleualign
[63] https://www.apache.org/licenses/LICENSE-2.0.html
[64] Los cognados son palabras de dos lenguas que comparten significado, ortografía y pronunciación. Ya sea que comparten un mismo origen etimológico o por préstamos lingüísticos.

ajustar. Sin embargo, el enfoque de YASA es simple, ya que implementa una representación gráfica de puntos de un corpus bilingüe, es decir, una matriz M, donde la i-ésima línea representa la i-ésima palabra del texto origen y la columna j representa la j-ésima palabra del texto destino. En este caso, una celda $M_{i,j}$ se fija en 1 si la i-ésima palabra origen y la j-ésima palabra destino son cognados. Entonces para un bitexto de origen I y destino J que entra en una relación análoga con palabras en el otro idioma, se busca la alineación que minimice $S(I, J)$, donde $S(i, j)$ se define como:

$$\begin{cases} 0 & si \quad i = j = 1 \\ \\ \underset{k=i-R}{\overset{i-1}{min}} \quad \underset{l|M_{k,l}=1}{min} \left(S(k,l) + c(k,l,i,j)\right) & Otro \quad caso \end{cases} \tag{2.58}$$

Con una función de costo

$$c(k,l,i,j) = \left| \frac{l-j}{k-i} - a \right| + (k-i-1) \times \mathcal{C}$$

donde R es una constante que establece la tolerancia máxima para discontinuidades en la alineación afín. $\mathcal{C}$ es una constante que penaliza una discontinuidad y a es la pendiente de la diagonal principal. Yasa usa los valores $R = 50$ y $\mathcal{C} = 5$. Después de realizar la alineación basada en afines, delimita una búsqueda de haz de oraciones como un número de oraciones de tamaño fijo centradas alrededor de los pares involucrados que se encuentran en el nivel cognado. En la implementación YASA tiene implementado un tamaño haz de 20 oraciones por defecto.

En la segunda etapa, el componente de base afín se agrega a la función minimizada mientras se alinea. La puntuación de coincidencia ($match$) se escribe con la siguiente ecuación:

$$S_{yasa} = -log \left[\left(\frac{P_T(c|n)}{P_R(c|n)} \right)^{\lambda_1} \times P(match)|\delta)^{\lambda_2} \right] \tag{2.59}$$

donde δ es una cantidad calculada directamente a partir de la longitud, contada en caracteres, de las oraciones fuente y candidatas a emparejar, asumiendo que sigue una distribución normal. Por otro lado, el primer término es una razón de verosimilitud que estima la probabilidad de que un emparejamien-

to que involucre n palabras en promedio comparta c cognados bajo el supuesto de que las oraciones están traducidas (numerador) o no (denominador). Asimismo, ambas distribuciones se modelan con binomios cuya probabilidad de éxito P_T y P_R se establecieron empíricamente con valores de 0.3 y 0.09, respectivamente.

Una vez desarrollado e introduciendo un nuevo coeficiente para la distribución a priori de coincidencias, finalmente la puntuación se convierte en:

$$S_{yasa} = -\lambda_1 \left(\left[c\, log \frac{P_T}{P_R} \right] - \left[(n-c)\, log \frac{1-P_T}{1-PR} \right] \right) \\ -\lambda_2\, logP(\delta|match) - \lambda_3\, logP(match) \tag{2.60}$$

2.4. Métrica de evaluación de resultados

Para determinar si una traducción automática es correcta o incorrecta, es necesario realizar evaluaciones de precisión. Las evaluaciones humanas serían largas y costosas. Para detectar los errores de traducción de una manera rápida y eficiente, son mediante métricas automáticas.

2.4.1. BLEU

Bilingual Evaluation Understudy (BLEU) es una métrica de evaluación automática propuesta por (Papineni et al., 2002). La idea principal de esta propuesta es que para evaluar la calidad de una traducción automática, se debe medir su cercanía a una o más traducciones hechas por humanos, donde estos servirán como referencia usando una métrica numérica, haciendo uso de la precisión modificada[65] que se basa en *n-gramas*.

Entonces, la tarea principal de BLEU es comparar los *n-gramas* de la traducción candidata con los *n-gramas* de la traducción de referencia y contar el número de coincidencias (ver Cuadro 2.3). Las coincidencias son independientes de la posición de las palabras de la frase. Cuantas más coincidencias, mejor

[65]La precisión modificada busca mejorar el cálculo de la presición, que para hacer este cálculo simplemente se cuenta el número de palabras de la traducción candidata que coinciden con una traducción de referencia y posteriormente dividirla con el total de palabras de la traducción candidata.

Candidato	It is a guide to action which ensures that the military always obeys the commands of the party.
Referencia 1	It is a guide to action that ensures that the military will forever heed Party commands.
Referencia 2	It is the practical guide for the army always to heed the directions of the party.

Cuadro 2.3: Traducción candidato generado por un traductor automático con dos traducciones de referencia hechas por humanos. Adaptado de (Papineni et al., 2002).

será la traducción candidata.

Por lo tanto, la precisión modificada P_n se calcula con la siguiente ecuación

$$P_n = \frac{\sum_{C \in candidatos} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in candidatos} \sum_{n-gram' \in C'} Count(n-gram')} \tag{2.61}$$

Para que una traducción candidata con una puntuación alta pueda coincidir con las traducciones de referencia en longitud, elección de palabras y orden de palabras; se le agrega un factor de penalización por brevedad BP, éste se calcula con la ecuación 2.62, donde c es la longitud de la oración candidata y r la longitud efectiva del corpus de referencia. Por lo tanto

$$BP = \begin{cases} 1 & si \quad c < r \\ \\ e^{\frac{1-r}{c}} & si \quad c \leq r \end{cases} \tag{2.62}$$

Finalmente, la ecuación del evaluador BLEU es

$$BLEU = BP * exp(\sum_{n=1}^{N} w_n log P_n) \tag{2.63}$$

Los resultados de BLEU se encuentran en un rango de entre 0 y 1, aunque en las publicaciones en el campo de la TA, es común multiplicarlo por 100. Si el valor es cercano a 1 (o 100 en el campo de la TA), es considerado como un resultado exitoso. Este es independiente de la lengua, ya que utiliza sus propias métricas para comparar las frases.

2.4.2. Perplejidad

Para medir la calidad de traducción de un modelo respecto a un lenguaje de referencia, es necesario mostrarle datos invisibles que no estaban en nuestro corpus de entrenamiento.

La *perplejidad* es una métrica de evaluación intrínseca[66] para los modelos de lenguaje (Jurafsky y Martin, 2019). Para ello se requiere un conjunto de pruebas o *test*.

Formalmente se define a la *perplejidad* de un modelo de lenguaje en un conjunto de pruebas, como la probabilidad inversa del conjunto de prueba normalizada por el número de palabras. Esto se calcula con la siguiente ecuación:

Dado un conjunto de pruebas $W = w_1 w_2 ... w_N$

$$PPL(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} \tag{2.64}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$$

Para expandir la ecuación de la probabilidad de W se recurre a la regla de la cadena, quedando expresado como

$$PPL(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 w_2 ... w_{i-1})}} \tag{2.65}$$

La *perplejidad* indica el número de posibles candidatos para la palabra w_i dado $w_1 w_2 ... w_{i-1}$. De la ecuación 2.65 cuanto mayor es la probabilidad condicional de la secuencia de palabras, menor es la perplejidad. Por consiguiente, minimizar la perplejidad equivale a maximizar la probabilidad del conjunto de pruebas según el modelo de lenguaje. Es decir, si los valores de la perplejidad son bajos, indican un mejor ajuste.

[66] Una métrica de evaluación intrínseca es aquella que mide la calidad de un modelo independientemente de las tareas que están destinadas.

2.5. Datos

Un modelo de NMT necesita ser alimentado de unos corpus que contiene miles o millones de frases de dos lenguas alineadas. Estos conjuntos de datos son de: entrenamiento (*train*), desarrollo (*dev*) y pruebas (*test*).

El corpus *train* es utilizado para entrenar el modelo y es el más grande de todos. Mientras que *dev* es el conjunto de datos utilizado para evitar el sobreentrenamiento o sobreajuste de un modelo, es decir, para que las traducciones puedan ser generalizadas a frases no vistas durante el entrenamiento. Por último, *test* es el corpus que se utiliza para la evaluación de desempeño del ajuste final del modelo.

Actualmente con el surgimiento de varios grupos de traductores en la región mixe y con la llegada del internet en algunas zonas; la escritura *Ayuuk* está viviendo su proceso de digitalización. Sin embargo, al empezar a clasificar las variantes de la lengua *Ayuuk*, se observa que cada variante cuenta con pocos recursos para la creación de los corpus. En esta tesis se está trabajando con la variante que se habla en el municipio de San Juan Güichicovi. Esta variante tiene escasos recursos digitalizados, por lo que se recurrió al trabajo de campo para la recolección de más textos. A continuación se muestran los datos disponibles de la lengua *Ayuuk* hasta el momento.

2.5.1. JW300

JW300 es un corpus paralelo de más de 300 lenguas del mundo, compiladas por (Agić y Vulić, 2019). Es el resultado de traducciones religiosas y no religiosas de diversas lenguas del mundo que la comunidad de los Testigos de Jehová ha estado traduciendo, estos datos están disponibles en OPUS[67].

En JW300 la lengua *Ayuuk* aparece con el identificador *mco* del ISO 639-

[67]El proyecto OPUS (Tiedemann, 2012) es una colección de textos extraídos de la web, convertidos y alineados para proporcionarle a la comunidad un corpus paralelo bajo licencia CC-BY-NC-SA (https://creativecommons.org/licenses/by-nc-sa/3.0/).

3[68], y de acuerdo a la clasificación de SIL International[69] pertenece al mixe de Coatlán. De igual manera, la lengua española aparece con el identificador *es* del ISO-639-1[70]. Actualmente el corpus español-*Ayuuk* de la región de Coatlán que proporciona JW300 tiene un total de 79,132 frases alineados y es el más extenso hasta el momento para la lengua *Ayuuk*. El proceso de descarga y procesamiento de textos se aborda en el Capítulo 3.

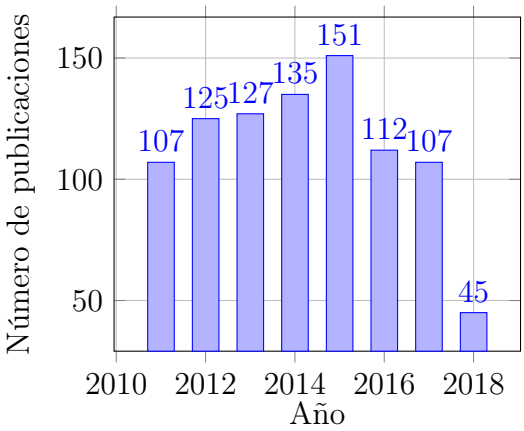


Figura 2.27: Estadística de publicaciones de la revista Atalaya en la lengua *Ayuuk* de la variante de la región de Coatlán, cuenta con 107 en el año 2011, 125 en 2012, 127 en 2013, 135 en 2014, 151 en 2015, 112 en 2016, 107 en 2017 y 45 en 2018; haciendo un total de 909 publicaciones. Datos tomados de JW300[71]. Elaboración propia.

La lengua *Ayuuk* de la variante del municipio de San Juan Güichicovi (*mir* en ISO-369-3) explorada en este trabajo, no se encuentra disponible en JW300. Por tal motivo, fue necesario el trabajo de campo para la recolección de textos de diferentes fuentes para las cuales había una traducción disponible entre español y *Ayuuk* de la variante mencionada.

[68]ISO 693-3 es un conjunto de códigos que define identificadores de tres letras para todos los lenguajes humanos conocidos. Las diferentes variantes de la lengua mixe aparecen con las siguiente claves: mixe de Coatlán *mco*, mixe del Istmo (San Juan Güichicovi) *mir*, mixe de Juquila *mxq*, mixe de Mazatlán *mzl*, mixe de Tlahuitoltepec *mxp*, mixe de Totontepec *mto*, mixe de Quetzaltepec *pxm* y mixe del este *neq*.

[69]https://iso639-3.sil.org/code_tables/639/data

[70]El español en el ISO-639-1 aparece con el identificador *es* y en el ISO-369-2 e ISO-369-3 con *esp* https://iso639-3.sil.org/code_tables/639/data.

[71]https://object.pouta.csc.fi/OPUS-JW300/v1/language_statistics.json (Última visita 18 de agosto 2020).

2.5.2. Traducciones recopiladas

Como se mencionó en la sección anterior, hasta el momento no hay ningún corpus disponible para la variante del *Ayuuk* de Güichicovi. Hasta donde se sabe, no ha habido una construcción de tal sistema para esta variante.

Los recursos que se pudieron recolectar en el trabajo de campo se pueden ver en el Cuadro 2.4, así como su estado abierto o no abierto.

Recursos	**es**	**mir**
[a]La Biblia evangelios de: Mateo, Juan, Lucas y Marcos.	Abierto	No abierto
[a]Constitución Política de los Estados Unidos Mexicanos.	Abierto	No abierto
[b]Cantos y poemas.	No abierto	No abierto
[c]Na ap na deedy kuentë. [c]El rey kondoy.	No abierto	No abierto
[d]Fabulas de Esopo.	Abierto	No abierto
[e]Archivo Nacional de lenguas indígenas de México.	No abierto	Abierto
[f]Extracciones en redes sociales.	Abierto	Abierto
[g]The dragon and the rabbit	Abierto	Abierto
[h]Frases traducidos por el autor.	Abierto	Abierto

Cuadro 2.4: Fuente de datos recopilados. Los datos abiertos se pueden encontrar en Github[72].

Donde:

a) Constitución Política de los Estados Unidos Mexicanos (UNTI-INALI); la Biblia (UNTI). Traducido por el Tr. Victoriano Santiago Cayetano, hablante nativo y traductor de la Unión Nacional de Traductores Indígenas (UNTI).

b) Cantos[73] y poemas[74].

c) Na ap na deedy kuentë; El rey kondoy. Colección personal y traducciones de Albino Pedro Juan, promotor de la lengua *Ayuuk*.

d) Esopo Mixe. Traducción del profesor Melchor Escobar Ocaña y profesora Gláfira Azcona Figueroa.

e) Lista de frases únicas del Archivo Nacional de Lenguas Indígenas de México (Lyon, 1980), traducidas por el autor de esta tesis.

[72] https://github.com/DelfinoAyuuk/corpora_ayuuk-spanish_nmt/

[73] Algunos cantos populares en *Ayuuk* fueron escuchados y escritos por el autor de esta tesis.

[74] Traducidos por el Tr. Victoriano Santiago Cayetano.

f) Recopilaciones en redes sociales[75].

g) The dragon and the rabbit[76]. Recopilación del lingüista del Instituto Lingüístico de Verano Norman W. Nordell durante su estancia en el municipio de San Juan Güichicovi. Este cuento es del inglés al *Ayuuk* y está bajo licencia CC BY-NC-SA[77].

h) Frases traducidas por el autor de esta tesis. Las frases en español son del proyecto Tatoeba[78].

Dado lo anterior, el corpus con el que se realizó este trabajo se compone de las siguientes categorías: religión, leyes, literatura, cantos, redes sociales y general (ver Cuadro 2.5).

Categorías	KB
Religión	935.3
Leyes	875.6
Literatura	143.2
Cantos y poemas	6.2
Redes sociales	15.4
General	83.8

Cuadro 2.5: Tamaño en KiloBytes de los archivos `.txt` de cada categoría que contiene el corpus creado.

Algunos de los archivos ya se encontraban alineados, otros no. Para aquellos que no estaban alineadas, se crearon las alineaciones de manera automática con la herramienta YASA (ver Figura 2.29), este proceso se describe en la Sección 3.3. Al final de todo el procedimiento se pudo generar un corpus de 8, 127 frases alineados de español al *Ayuuk* variante de Güichicovi, ver Capítulo 3.

[75]Es necesario aclarar que toda la escritura recopilada en redes sociales fueron corregidas por el autor de esta tesis, ya que en su mayoría presentaban errores de escritura.

[76]https://mexico.sil.org/es/resources/archives/55868 (Última visita 26 en marzo de 2021).

[77]https://creativecommons.org/licenses/by-nc-sa/4.0/

[78]https://www.manythings.org/anki/ (Última visita 26 de marzo de 2021).

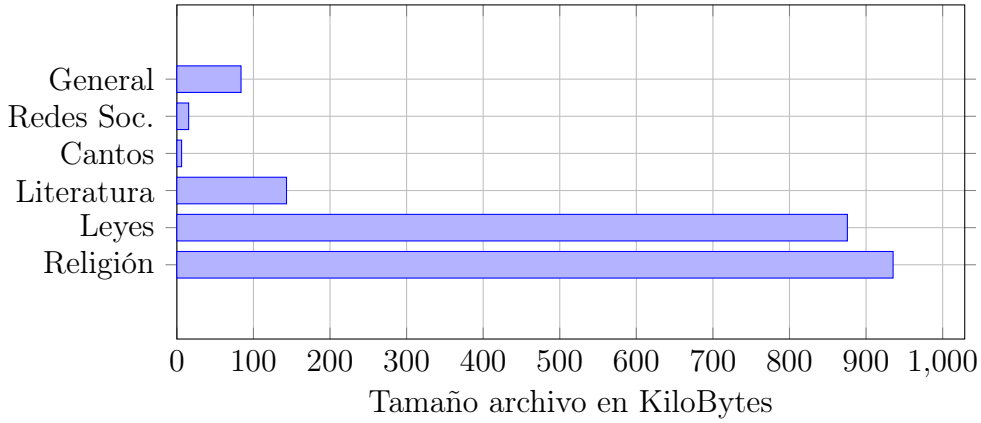


Figura 2.28: Representación gráfica del Cuadro 2.5.

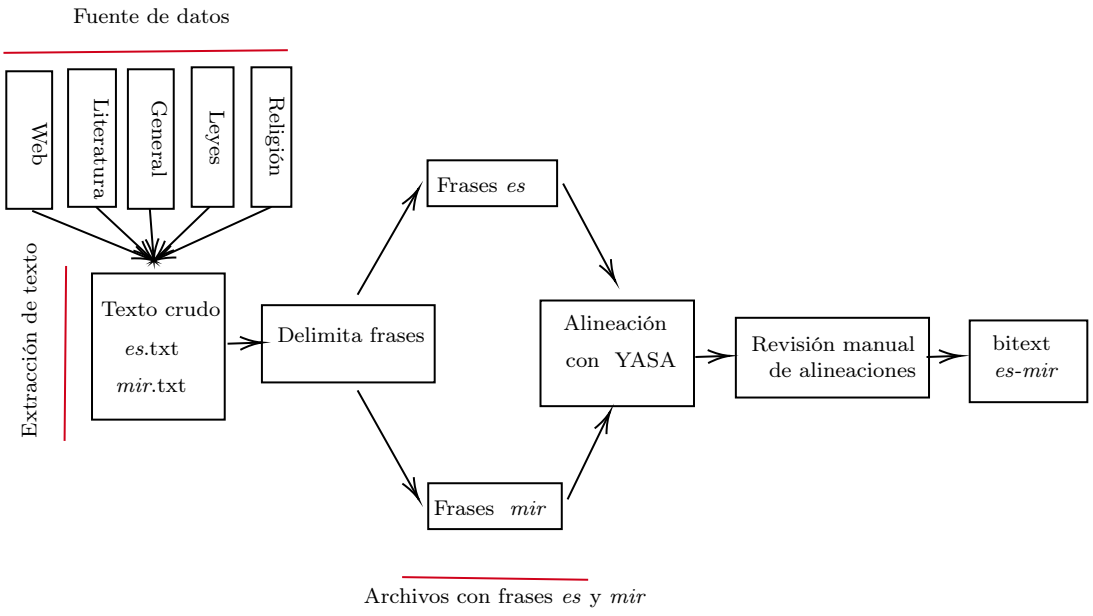


Figura 2.29: Proceso de construcción del bitexto *es-mir*.

Capítulo 3

Corpus

En este capítulo se presenta la implementación de los métodos del estado del arte para la creación del traductor automático neuronal. La metodología abarca desde la recolección de datos, preprocesamiento de textos, criterios de normalización, alineación automática con Yasa, definición del tamaño de los corpus, tokenización por subpalabras y configuración de experimentos.

3.1. Recolección de datos

La materia prima de un sistema de traducción automático neuronal son conjuntos de corpus bilingües alineados. Como se mencionó en la Sección 2.5, estos conjuntos de datos son: *train*, *dev* y *test*. Sin embargo, el *Ayuuk* de la variante de Güichicovi (*mir*) es una lengua de escasos recursos textuales digitalizados, por lo que hasta el momento no cuenta con ningún corpus de traducciones entre la lengua *Ayuuk* y español. En cambio, el *Ayuuk* de la variante de la región de Coatlán (*mco*) cuenta con un corpus de libre acceso en OPUS[79], estas traducciones han sido publicados por los Testigos de Jehová en sus revistas Atalaya que posteriormente fueron procesados y alineados por Agić y Vulić (2019), ver en Sección 2.5.1.

Debido a que este es el primer trabajo de NMT para la lengua *Ayuuk* de la variante de Güichicovi, el trabajo de campo fue necesario para recolectar traducciones y poder armar los corpus de entrenamiento para el modelo de traducción. En el trabajo de campo y en ciertos sitios de internet se pudo recolectar documentos en distintos formatos, otros se tradujeron directamente a

[79] https://opus.nlpl.eu/

un formato `.csv` y en algunos casos se recurrieron a aplicaciones que utilizan Optical Character Recognition (OCR) para escanear y extraer los textos de documentos que no están digitalizados.

Todos los datos recolectados se convirtieron a un formato `.txt`[80] para facilitar los trabajos de normalización y alineación.

3.2. Proceso de normalización

Como se mencionó en la Sección 1.1.2, la controversia por las diferentes formas de escritura en la que se encuentra el *Ayuuk* de la variante de Güichicovi y de otros municipios de la región mixe, hace que el procesamiento de los textos no sea tan sencillo.

Debido a que este trabajo no pretende imponer ningún sistema de escritura en específico, el proceso de normalización se basa en el artículo de Sagi-Vela González (2019). Este artículo reune toda la información relacionada con el camino que ha seguido el intento de unificación del alfabeto *mixe*, así como los acuerdos a lo que se ha llegado. Con la normalización se pretende recuperar el mayor número de palabras que fueron escritos con diferentes estilos y así evitar duplicados en el sistema de traducción automática (ver Figura 3.1). Además, en este proceso se define la delimitación de cada frase.

En el primer preprocesamiento se implementan las siguientes sustituciones en la escritura del *Ayuuk*: la *ch* por la *tsy* y la *ñ* por la *ny*, como se señala en Sagi-Vela González (2019). Asimismo, los acentos en las vocales de una palabra en *Ayuuk* son eliminados, los del *español* se mantienen (ver Figura 3.2); si un texto termina con un `?`, `!` y con un punto, se consideran como una frase (aplica para textos del *Ayuuk* y del *español*) (ver Figura 3.3).

Por último, para mantener las características de las frases, los caracteres como *punto y coma*, *coma*, *guiones* y *comillas*; no se quitan. Estos caracteres servirán en el proceso de alineación con YASA. El código completo del proceso de normalización se puede consultar en https://github.com/DelfinoAyuuk/corpora_ayuuk-spanish_nmt/

[80] Para la conversión de un pdf a txt desde la consola de Ubuntu, escribimos `pdftotext -layout "nombre-doc-origen".pdf "nombre-doc-destino".txt`

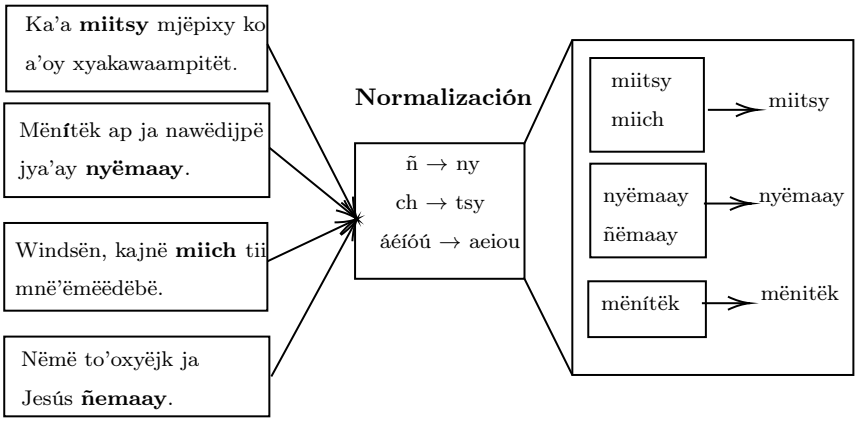


Figura 3.1: Ejemplo de normalización, las frases son extractos de las *Fábulas de Esopo*, *Na ap na deedy kuentë* y *Evangelio de San Juan*; se puede apreciar que los pares de palabras (**miitsy**, **miich**) y (**nyëmaay**, **ñëmaay**) tienen el mismo significado, además, se decide quitar los acentos en las vocales para la recuperación de palabras. Elaboración propia.

```
#sustituye ch por tsy; ~n por ny
def convenio(a):
        a = a.replace("~n","ny")
        a = a.replace("ch","tsy")
        return a
```

Figura 3.2: Función *convenio* para reemplazar la *ñ* por la *ny* y la *ch* por la *tsy*; extracto de código para normalización escrito en *Python*. Elaboración propia.

```
pre = pre.replace("!","!.")
pre = pre.replace("?","?.")
pre = pre.replace(".",".\n")

#Considera una frase hasta donde encuentre un punto
separaxpunto = pre.split("\n")
separaxpunto = [x.strip() for x in separaxpunto]
```

Figura 3.3: Código para la segmentación de oraciones; extracto de código para normalización escrito en *Python*. Elaboración propia.

3.3. Alineación con YASA

En algunas traducciones recolectadas las frases ya están alineadas por defecto, puesto que vienen ordenados por filas y columnas con sus respectivas traducciones, por ejemplo, las que están en un formato `.csv`. Sin embargo, hay otros textos muy extensos, como la *Biblia*, que necesitan de un alineador automático para que las frases tengan correspondencia.

Como ya se expuso en la Sección 2.3.2.1, YASA es un programa bajo licencia *Apache License 2.0*[81] que alinea dos traducciones de un texto, los archivos que contienen los textos traducidos deben ser del mismo formato. La alineación se realiza frase por frase para generar un bitexto[82].

Con el segmentador de oraciones que se presentó en la sección anterior (ver Figura 3.3), se pudo realizar la separación de frases siguiendo los criterios mencionados en la Sección 3.2. Las frases fueron acomodadas por filas en un formato `.txt`. A continuación se muestra el proceso de alineación utilizando dos pequeños párrafos de *Na ap na deedy kuentë*; español (izquierda) y *Ayuuk* variante de Güichicovi (derecha).

-Estoy deteniendo este cerro, si lo suelto, caerá sobre mí. ¿Me podrías hacer un favor de ayudarme a detener este cerro? Ya estoy muy cansado y necesito ir a comer, porque llevo días sin probar alimento.	*-Joom adá m'ëjk tunëtsy ndij'ijpy, koodsy adá n'ëxmádsët, xynya ka'abedëbëtsy ayá. ¿Nej kap mayájt mdunët koodsy xypyubedët, ok tij'ijpë miidsy adá tun? Tyámëtsy madyëjk nja wi kyay nja wi y'u'uga'any, je'egëxp koodsy jegyáy jey'u'ugy maxyëë të ndij'ity; tëëdsy anaxyë nëgë anuu'xë'ëy.*

Los textos presentados son normalizados y segmentados, generando dos archivos `.txt`; `es.txt` (Figura 3.4) y `mir.txt` (Figura 3.5). Donde cada línea corresponde a una frase.

Una vez preparado el par de textos *origen* y *destino*, el siguiente paso es ejecutar YASA dentro del directorio donde se encuentran los pares de textos.

[81] https://www.apache.org/licenses/LICENSE-2.0
[82] Un bitexto es un par de textos origen y destino que se corresponden entre sí.

```
1 - estoy deteniendo este cerro, si lo suelto, caerá sobre mí .
2 ¿me podrías hacer un favor de ayudarme a detener este cerro? .
3 ya estoy muy cansado y necesito ir a comer, porque llevo días
sin probar alimento .
```

Figura 3.4: Archivo `es.txt` normalizado y donde cada frase termina con un punto.

```
1 -joom ada m'ëjk tunëtsy ndij'ijpy, koodsy ada n'ëxmadsët,
xynya ka'abedëbëtsy aya .
2 ¿nej kap mayajt mdunët koodsy xypyubedët, ok tij'ijpë miidsy
ada tun? .
3 tyamëtsy madyëjk nja wi kyay nja wi y'u'uga'any, je'egëxp
koodsy jegyay jey'u'ugy maxyëë të ndij'ity; tëëdsy anaxyë nëgë
anuu'xë'ëy .
```

Figura 3.5: Archivo `mir.txt` normalizado y donde cada frase termina con un punto.

Con este paso se genera un archivo con extensión `.xml` que contiene las alineaciones de las frases (ver Figura 3.6). Todo este proceso se ejecuta dentro de una terminal linux Ubuntu con el siguiente comando[83].

```
yasa -i o -e c -b 20 -o c es.txt mir.txt >esmir.xml
```

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE CESALIGN PUBLIC "-//CES//DTD cesAlign//EN" "">
3  <cesAlign VERSION="1.14"  type="sent" fromDoc="es.txt"
       toDoc="mir.txt">
4  <linkList>
5  <linkGrp>
6  <link xtargets="1;1" certainty="1.46127"></link>
7  <link xtargets="2;2" certainty="1.518"></link>
8  <link xtargets="3;3" certainty="3.31622"></link>
9  </linkGrp>
10 </linkList>
11 </cesAlign>
```

Figura 3.6: Formato `.xml` de la alineación un texto origen `es.txt` con un texto destino `mir.txt` utilizando YASA.

[83]Donde `-i o` es la entrada de una sentencia por cada línea; `-e c` alinea cognados para encontrar puntos de paso; `-b n` = 20 establece un radio entre puntos de paso; `-o c` establece el formato de salida de la alineación.

En la Figura 3.6 con `xtargets="1;1"` se puede interpretar que la primera frase del *español* (Figura 3.4) tiene correspondencia con la primera frase del *Ayuuk* (Figura 3.5), de manera similar la segunda y tercera frase tienen el mismo patrón.

En las alineaciones que contienen cientos o miles de frases, estas no son tan obvias, como se puede ver en la Figura 3.7. La frase 202 del *español* está alineada con la frase 206 del *Ayuuk*, pero la frase 207 del *Ayuuk* no tiene correspondencia con ninguna frase del *español*. Asimismo, la frase 211 del *Ayuuk* tiene correspondencia con dos frases del *español*; 206 y 207.

```
<link xtargets="202;206" certainty="-1.50749"></link>
<link xtargets=";207" certainty="2.48883"></link>
<link xtargets="203;208" certainty="2.81229"></link>
<link xtargets="204;209" certainty="13.8585"></link>
<link xtargets="205;210" certainty="4.99044"></link>
<link xtargets="206 207;211" certainty="7.92825"></link>
<link xtargets="208;212" certainty="-1.50749"></link>
```

Figura 3.7: Formato `.xml` de una alineación con muchas frases. Extracto de la alineación de la *Constitución Política de los Estados Unidos Mexicanos*.

En este trabajo se decide excluir los `xtargets` origen y `xtargets` destino que no cuentan con ninguna o tengan doble alineación. Para este proceso, el archivo `.xml` es parseado con el lenguaje Python para convertirlo en un `dataframe` y eliminar los `xtargets` que tengan las características antes mencionadas (ver Figura 3.8).

	xtargets_es	xtargets_mir
0	1	1
1	2	2
2	3	3
3	5	6
4	8	8
...	...	...
112	129	127
113	130	128
114	131	129
115	132	130
116	133	131

117 rows × 2 columns

Figura 3.8: Dataframe de un `.xml` parseado; alineación de *Na ap na deedy kuentë*.

Los archivos `.txt` del lenguaje origen y destino que contienen las frases también son convertidos a un dataframe, a estos se les asigna un `id` que posteriormente con la función `merge` de Python se juntan con los `xtargets` correspondientes a cada lengua, ver código en la Figura 3.9, y el resultado en la Figura 3.10.

```
#Lista para los Ids
ID = []
for i in range(1,len(frases_es)+1):
    ID.append(i)
#Covierte lista id a dataframe
id_ = pd.DataFrame(ID,columns=['id_es'])
#Concatena id con su respectiva frase
id_frases = pd.concat([id_,frases_es], axis = 1)
#Une dos dataframes con el metodo merge de pandas
df_es = pd.merge(es, id_frases, left_on='xtargets_es',
   right_on='id_es')
```

Figura 3.9: Código para unir los dataframes que contienen los `xtargets` y las frases del español, mediante el método `merge` de Python. Extracto de código de parseo y unión de dataframes.

Después, se genera un nuevo dataframe que contiene la concatenación de las frases del español con las frases del *Ayuuk*; siendo ésta la alineación final de frases de un documento generado con YASA (ver Figura 3.11).

	xtargets_es	id_es	0
0	1	1	kong'oy recibe su territorio y su pueblo 1
1	2	2	y se dice, pues, que de verdad, llego el tiemp...
2	3	3	2
3	5	5	que de dos huevos que fueron encontrados alla ...
4	8	8	desde entonces dicen que cada uno de nosotros ...
...	...	...	...
111	128	128	así es como los mixes creemos que es ella: jad...
112	129	129	así es como se dice, que es ella misma la que ...
113	130	130	37
114	131	131	de esta manera se cuenta que kong'oy enseñó a ...
115	132	132	dicen también que cuando nuestros antepasados ...

116 rows × 4 columns

Español

	xtargets_mir	id_mir	0
0	1	1	ko kong'oy nyaaxpaaty kyajtpaaty 1
1	2	2	jadu'un mëdya'agy y'ity ko ja xëë diempë y'aba...
2	3	3	2
3	6	6	jadu'un mëdya'agy y'ity ko ja animaltu'udy ajx...
4	8	8	jadu'un ajxy myëdya'agy, jekyën jadu'un y'awij...
...	...	...	...
111	126	126	nëm magunaax ajxy myëna'any ko ja tajëëw jyëds...
112	127	127	ja tajëëw kaayëëk uukeek ajxy xymyooyëmb, paad...
113	128	128	37
114	129	129	jadu'unëk kong'oy ja n'ap ndeedy nëë duung jek...
115	130	130	ko ajxy tyuung abëjky, amdsoo ajxy je'e jyugya...

116 rows × 4 columns

ayuuk

Figura 3.10: Unión de dataframes; frases con xtargets.

Posteriormente, todas las alineaciones generadas así como las que ya estaban alineadas se almacenan en un solo dataframe, y a cada columna se le

aplica la función `drop_duplicates` para eliminar duplicados, tanto en la primera como en la segunda columna.

Finalmente se logró juntar un bitext con 6,717 frases alineadas con extensión `.txt` y, es el archivo que se manipulará para generar los corpus de *train*, *dev* y *test*. Es necesario mencionar que las alineaciones fueron extraídas de manera aleatoria por el autor de este trabajo para revisar de forma manual si efectivamente las alineaciones tienen correspondencia[84]. El Código completo de parseo y unión de dataframes se puede consultar en https://github.com/DelfinoAyuuk/corpora_ayuuk-spanish_nmt/.

	es	mir
0	kong'oy recibe su territorio y su pueblo 1	ko kong'oy nyaaxpaaty kyajtpaaty 1
1	y se dice, pues, que de verdad, llego el tiemp...	jadu'un mëdya'agy y'ity ko ja xëë diempë y'aba...
2	2	2
3	que de dos huevos que fueron encontrados alla ...	jadu'un mëdya'agy y'ity ko ja animaltu'udy ajx...
4	desde entonces dicen que cada uno de nosotros ...	jadu'un ajxy myëdya'agy, jekyën jadu'un y'awij...
...	...	...
111	así es como los mixes creemos que es ella: jad...	nëm magunaax ajxy myëna'any ko ja tajëëw jyëds...
112	así es como se dice, que es ella misma la que ...	ja tajëëw kaayëëk uukeek ajxy xymyooyëmb, paad...
113	37	37
114	de esta manera se cuenta que kong'oy enseñó a ...	jadu'unëk kong'oy ja n'ap ndeedy nëë duung jek...
115	dicen también que cuando nuestros antepasados ...	ko ajxy tyuung abëjky, amdsoo ajxy je'e jyugya...

116 rows × 2 columns

Figura 3.11: Alineación o bitexto de frases *es-mir* que corresponden al documento *Na ap na deedy kuentë*.

3.4. Obtención de los corpus *train*, *dev* y *test*

Uno de los factores que definen el rendimiento del modelo de traducción son el tamaño y el contenido del corpus *train*, *dev* y *test*. Para los experimentos de este trabajo se proponen dos divisiones de conjuntos de corpus: *estricto* y *aleatorio*. El tamaño de cada corpus dependerá de la división.

- Para la división *estricta*, el bitext con las 912 frases especializadas del *Archivo Nacional de Lenguas Indígenas* (Lyon, 1980) se reservaron para

[84]El autor de esta tesis conoce la lecto-escritura *Ayuuk* y es un hablante nativo de la lengua *Ayuuk* de la variante del municipio de San Juan Güichicovi, por lo que la revisión de la alineación automática fue realizada por el mismo autor.

ser el corpus *test*, para la obtención de los corpus *train* y *dev*, al bitext con las 6,717 frases únicas se le aplicó parte del código[85] del Proyecto Masakhane de lenguas Africanas. Donde el conjunto del *test* fue comparado por cada fila de las 6,717 frases alineadas del bitext para eliminar posibles duplicados, posteriormente se aplicó una función para eliminar frases que tenían alguna semejanza; la función `fuzzfilter`.[86] Finalmente, el bitext que contenía las 6,717 frases, fue reducido a 6,547; donde 5,847 son para *train* y 700 para *dev* (ver Cuadro 3.1).

- Para la división *aleatoria*, se juntaron las 912 frases especializadas del *Archivo Nacional de Lenguas Indígenas* con las 6,717 frases únicas, después se realizó un `random` para extraer nuevamente 912 frases que sirvieron como *test*[87]. Posteriormente se siguió el mismo procedimiento como para la división *estricta*. Finalmente, el bitext que contenía las 6,717 frases, fue reducido a 6,941; donde 5,941 son para *train* y 700 para *dev* (ver Cuadro 3.1).

3.4.1. Obtención de corpus en JW300 para el *Ayuuk* de la variante de la región de Coatlán

Para el caso del *Ayuuk* de la variante de la región de Coatlán. Esta variante está disponible en el portal de *JW300*[88] y ya se encuentra lista para su descarga. Esto se puede hacer instalando `opus-tools` desde la terminal de Linux con `pip install opustools-pkg`. Para descargar el corpus disponible se emplea el siguiente comando:

- `opus_read -d JW300 -s es -t mco -wm moses -w jw300.es jw300.mco -q`

Para descomprimir los datos se emplea el comando.

- `gunzip JW300_latest_xml_es-mco.xml.gz`

Una vez descargados los archivos que contienen las frases alineadas, se hace un procesamiento de textos[89] para eliminar caracteres especiales, duplicados y

[85] https://github.com/masakhane-io/masakhane-mt/blob/master/starter_notebook_into_English_training.ipynb

[86] Esta función se puede escribir gracias a la biblioteca `fuzzywuzzy`, que se basa en la distancia de Levenshtein para calcular una relación de semejanza entre dos secuencias para devolver un porcentaje de similitud.

[87] https://github.com/DelfinoAyuuk/corpora_ayuuk-spanish_nmt/

[88] https://opus.nlpl.eu/JW300-v1.php

[89] https://github.com/DelfinoAyuuk/corpora_ayuuk-spanish_nmt/

	Conjunto estricto		Conjunto aleatorio	
	Español	**Ayuuk Güichicovi**	**Español**	**Ayuuk Güichicovi**
Pares de frases	7,215			
Frases únicas	6,717			
Pares de frases train	5,847		5,941	
Palabras	102,753	116,916	104,504	117,289
Palabras únicas	12,990	18,399	13,150	18,775
Longitud promedio de frases	17.6	20	17.6	19.7
Longitud máxima de frases	93	109	142	171
Pares de frases dev	700		700	
Palabras	12,180	13,758	11,987	13,789
Palabras únicas	3,463	4,015	3,529	4,104
Longitud promedio de frases	17.4	19.7	17.1	19.7
Longitud máxima de frases	66	106	127	158
Pares de frases test	912		912	
Palabras	4,513	4,099	15,883	17,711
Palabras únicas	904	1,323	4,150	4,823
Longitud promedio de frases	5	4.5	17.4	19.4
Longitud máxima de frases	13	10	219	315

Cuadro 3.1: Descripción general del corpus para el traductor automático *es-mir-es*.

alineaciones vacías.

Para la obtención de los corpus *train*/*dev*/*test* se siguió el mismo procedimiento aleatorio mencionado en la Sección 3.4, debido a que JW300 no proporciona los corpus para realizar el *test*.

3.5. Proceso de tokenización

Después de crear los corpus necesarios para el entrenamiento del sistema de traducción y de realizar la limpieza de datos[90], la tokenización fue el siguiente paso antes de entrenar el modelo de traducción, ya que las entradas que recibe una red neuronal son tokens representados en forma de vectores. Como se vió en la Sección 2.1.2.1, la Traducción Automática presenta un problema de vocabulario abierto al tratar con palabras desconocidas, sin embargo, de acuerdo con Sennrich et al. (2016), esto se resuelve segmentando las palabras en subpalabras para generar combinaciones de palabras no vistas en el corpus.

Los corpus tanto para el conjunto *aleatorio* como para el conjunto *estricto* que se manipularon durante los experimentos de este trabajo son: *train.es*,

[90]Eliminando caracteres especiales en las frases.

	Español	Ayuuk Coatlán
Pares de frases	79,132	
Frases únicas	67,197	
Conjunto aleatorio		
Pares de frases train	65,071	
Palabras	1,102,333	918,525
Palabras únicas	28,976	42,506
Longitud promedio de frases	16.9	14.1
Longitud máxima de frases	75	78
Pares de frases dev	1000	
Palabras	17,276	14,505
Palabras únicas	3,537	3,541
Longitud promedio de frases	17.2	14.5
Longitud máxima de frases	66	52
Pares de frases test	1000	
Palabras	16,971	14,354
Palabras únicas	3,561	3,469
Longitud promedio de frases	16.9	14.3
Longitud máxima de frases	68	63

Cuadro 3.2: Descripción general del corpus para el *Ayuuk* de Coatlán (*mco*).

train.mir, *dev.es*, *dev.mir*, *test.es* y *test.mir* [91].

Para el proceso de tokenización, el procesamiento por Byte Pair Encoding (BPE) se aplicó sobre el corpus de entrenamiento para extraer un número determinado de códigos BPE[92], y posteriormente generar unidades de subpalabras aprendidas automáticamente a partir de los datos de entrenamiento. Para realizar dicha acción, se instaló la biblioteca `subword-nmt`[93].

```
pip install subword-nmt
```

Para extraer el vocabulario subword BPE del *train*, se ejecuta el siguiente comando desde la notebook de Jupyter Python.

```
1 ! subword-nmt learn-joint-bpe-and-vocab --input train.$src
```

[91]Donde *es* es el código ISO-369-1 de la lengua Español, *mir* el código ISO-369-3 de la lengua *Ayuuk* de Güichicovi. Para la prueba del *Ayuuk* de Coatlán se utilizaron las extensiones *es* y *mco* respectivamente.

[92]En este trabajo se probó con un vocabulario BPE de 4,000 y 2,000.

[93]El proceso de instalación se puede consultar en el repositorio https://github.com/rsennrich/subword-nmt.

```
    train.$tgt -s nvoca -o bpe.codes.nvoca --write-
    vocabulary vocab.$src vocab.$tgt
```

Aplicado el comando anterior, se generan los archivos *bpe.codes.nvoca*, *vocab.src* y *vocab.tgt* [94]. Posteriormente se aplican los siguientes comandos sobre los corpus *train, dev* y *test*[95].

```
1 ! subword-nmt apply-bpe -c bpe.codes.nvoca --vocabulary
    vocab.$src < train.$src > train.bpe.$src
2 ! subword-nmt apply-bpe -c bpe.codes.nvoca --vocabulary
    vocab.$tgt < train.$tgt > train.bpe.$tgt
3 ! subword-nmt apply-bpe -c bpe.codes.nvoca --vocabulary
    vocab.$src < dev.$src > dev.bpe.$src
4 ! subword-nmt apply-bpe -c bpe.codes.nvoca --vocabulary
    vocab.$tgt < dev.$tgt > dev.bpe.$tgt
5 ! subword-nmt apply-bpe -c bpe.codes.nvoca --vocabulary
    vocab.$src < test.$src > test.bpe.$src
6 ! subword-nmt apply-bpe -c bpe.codes.nvoca --vocabulary
    vocab.$tgt < test.$tgt > test.bpe.$tgt
```

Finalmente se generan los archivos con los textos segmentados en subpalabras, estos son: *train.bpe.es*, *train.bpe.mir*, *dev.bpe.es*, *dev.bpe.mir*, *test.bpe.es* y *test.bpe.mir*. El contenido de cada uno de estos serán los tokens de entrada para el entrenamiento, desarrollo y prueba del modelo de traducción.

3.6. Instalación de JoeyNMT

Para entrenar el modelo de traducción se necesita de una herramienta que soporte arquitecturas como RNNs y Transformers. JoeyNMT es la herramienta que se utilizó en este trabajo para realizar los experimentos con modelos Transformer. JoeyNMT se instala[96] con el siguiente comando

[94] Donde *nvoca* = (*2000 o 4000*), *src* es la lengua origen y *tgt* la lengua objetivo.

[95] El notebook completo se puede ver en el proyecto original de lenguas africanas Masakhane https://github.com/masakhane-io/masakhane-mt/blob/master/starter_notebook_into_English_training.ipynb.

[96] El proceso de instalación y los requerimientos se pueden consultar en el repositorio https://github.com/joeynmt/joeynmt.

```
pip install joeynmt
```

O clonando el repositorio con el siguiente comando

```
git clone https://github.com/joeynmt/joeynmt.git
```

3.6.1. Entrenamiento

La configuración de los experimentos en JoeyNMT se deben realizar en un formato `.yaml`, en este trabajo se llamó `transformer.yaml`. La configuración contiene la descripción de la arquitectura del modelo (número de capas ocultas, número de cabezales, dimensión de embeddings,...); ruta a los datos de *train.bpe.es*, *train.bpe.mir*, *dev.bpe.es*, *dev.bpe.mir*, *test.bpe.es* y *test.bpe.mir*; número de épocas y lotes; logitud máxima de frases; tasa de aprendizaje, entre otros parámetros[97].

Después de realizar la configuración necesaria, inicia el entrenamiento ejecutando el siguiente comando

```
! cd joeynmt; /ve/bin/python3 -m joeynmt train configs/
   transformer.yaml
```

Una vez finalizado el entrenamiento, si se desea conocer el puntaje BLEU del *dev* y *test* del experimento, se ejecuta el siguiente comando

```
! cd joeynmt; /ve/bin/python3 -m joeynmt test configs/
   transformer.yaml
```

Los resultados de algunos experimentos del modelo Transformer usando la herramienta JoeyNMT con los datos disponibles hasta el momento, se muestran en el siguiente capítulo.

[97]La configuración del formato `.yaml` de los experimentos para el traductor neuronal *Ayuuk*-Español se pueden consultar en https://github.com/DelfinoAyuuk/corpora_ayuuk-spanish_nmt.

Capítulo 4

Experimentos y resultados

En este capítulo se presentan, evalúan y comparan los resultados de varios experimentos realizados utilizando la arquitectura de traducción automática del estado del arte basado en redes neuronales; la arquitectura Transformer (Vaswani et al., 2017). Para probar la hipótesis y responder a las preguntas de investigación de este trabajo se entrenaron dos variantes del *Ayuuk*[98]; variante del municipio de San Juan Güichicovi[99] y variante de la región de Coatlán[100].

4.1. Configuración de arquitectura neuronal

Para los experimentos de este trabajo se realizaron cuatro configuraciones *codificador-decodificador* del modelo basada en la arquitectura Transformer. Las primeras tres configuraciones son para el traductor español al *Ayuuk* de Güichicovi y viceversa (*es-mir-es*):

- **Configuración *A***. Número de capas: 3, número de cabezales: 4, dimensión de embedding de entrada: 64, dimensión embedding: 64, tamaño de lote: 128.
- **Configuración *B***. Número de capas: 6, número de cabezales: 4, dimensión de embedding de entrada: 256, dimensión de embedding: 256, tamaño de lote: 128.

[98]En este capítulo las abreviaciones para el *Ayuuk* de la variante del municipio de San Juan Güichicovi, será mediante su identificador ISO-639-3 *mir*; variante del *Ayuuk* de la región de Coatlán con su identificador ISO-639-3 *mco*; y para el español mediante su identificador ISO-639-1 *es*.

[99]Traductor *es-mir* y *mir-es*.

[100]Traductor *es-mco* y *mco-es*.

- **Configuración *C***. Número de capas: 6, número de cabezales: 4, dimensión de embedding de entrada: 256, dimensión de embedding: 256, tamaño de lote: 32.

La cuarta configuración *codificador-decodificador* corresponde al traductor español al *Ayuuk* variante de la región de Coatlán y viceversa (*es-mco-es*):

- **Configuración *D***. Número de capas: 6, número de cabezales: 4, dimensión de embedding de entrada: 256, dimensión de embedding: 256, tamaño de lote: 512.

Estos modelos se entrenaron en un servidor con dos GPU Tesla V100. Para obtener el resultado de los modelos *es-mir* y *mir-es*, usualmente tomó alrededor de $2h$ por épocas de 100, $5h$ por 250 y $6h$ por 150 épocas. Por otro lado, para obtener el resultado de los modelos *es-mco* y *mco-es*, se tomó alrededor de $24h$ por 100 épocas. También se pudieron reproducir los experimentos en la plataforma *Colaboratory* (https://colab.research.google.com).

4.2. Evaluaciones BLEU y perplejidad

En este trabajo se consideró realizar dos divisiones en el conjunto de corpus, ver Sección 3.4. Los tamaños de *train*/*dev*/*test* del modelo de traducción español-*Ayuuk* de Güichicovi-español, difieren dependiendo de cada división:

- División estricta: *train* $5,847$ / *dev* 700 / test 912
- División aleatoria: *train* $5,941$ / *dev* 700 / test 912

Por cada división se realizaron cinco experimentos (ver Figuras 4.1 y 4.2), dos para la configuración con menos capas (A) y tres para la configuración con más capas (B). De igual manera se modificó:

a) La longitud máxima de frases (50 o 70).

b) El vocabulario del algoritmo de subpalabras BPE (con 2000 o 4000).

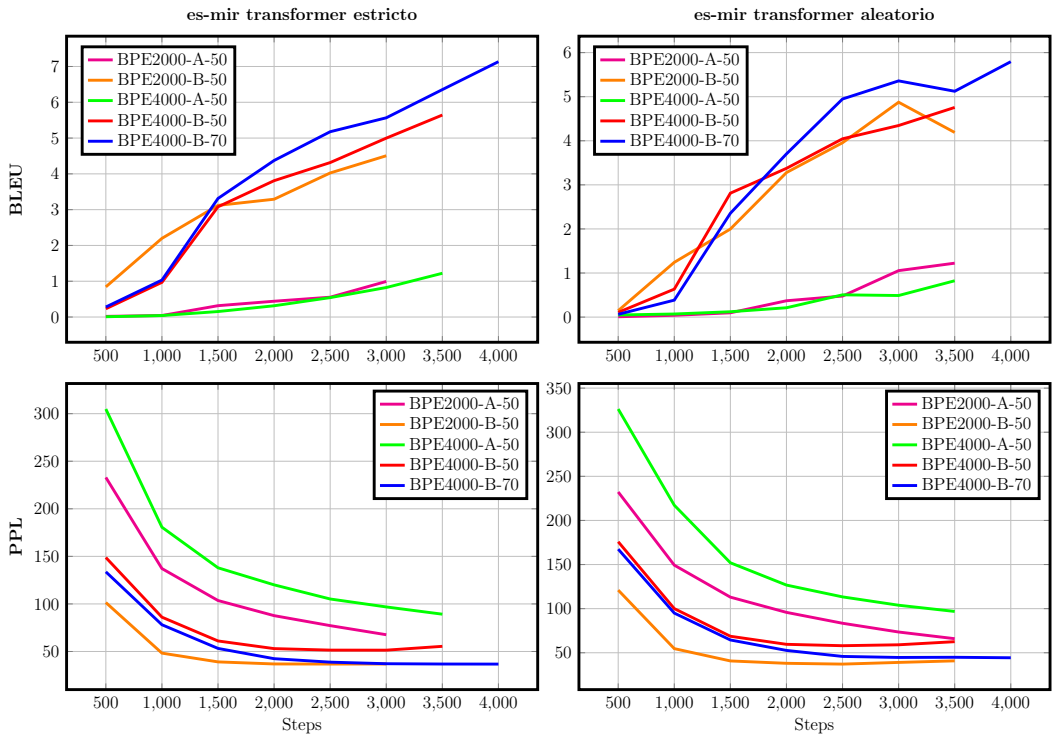


Figura 4.1: Perplejidad y BLEU en el conjunto de desarrollo durante el entrenamiento con 100 épocas en dirección *es-mir*.

4.2.1. Experimentos para el traductor *es-mir-es*

En las siguientes figuras se presentan los resultados de los experimentos para el traductor *es-mir-es*.

La Figura 4.1 muestra la perplejidad y puntuación BLEU en el conjunto de desarrollo durante el entrenamiento para la dirección del español (*es*) al *Ayuuk* (*mir*). La primera parte del Cuadro 4.1, las columnas dos al cinco, presentan los resultados de los conjuntos de desarrollo (*dev*) y prueba (*test*).

La Figura 4.2 muestra la curva de aprendizaje en la dirección de traducción *Ayuuk* (*mir*) al español (*es*). La segunda parte del Cuadro 4.1, las columnas del seis al nueve, presentan los resultados sobre el desarrollo (*dev*) y prueba (*test*) para esta dirección de traducción.

Como se puede observar, estos conjuntos de experimentos muestran que la traducción entre la lengua *Ayuuk* y español es posible, ver Sección 4.3. El modelo con la configuración (*A*) presenta un rendimiento muy bajo, en cambio

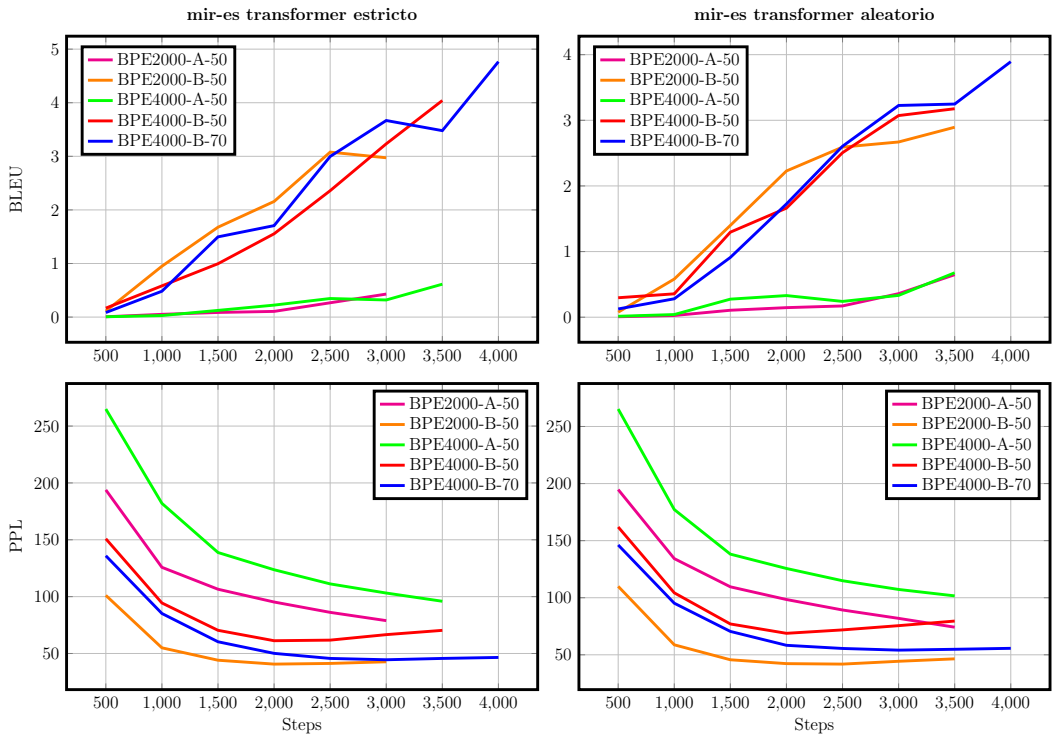


Figura 4.2: Perplejidad y BLEU en el conjunto de desarrollo durante el entrenamiento con 100 épocas en dirección *mir-es*.

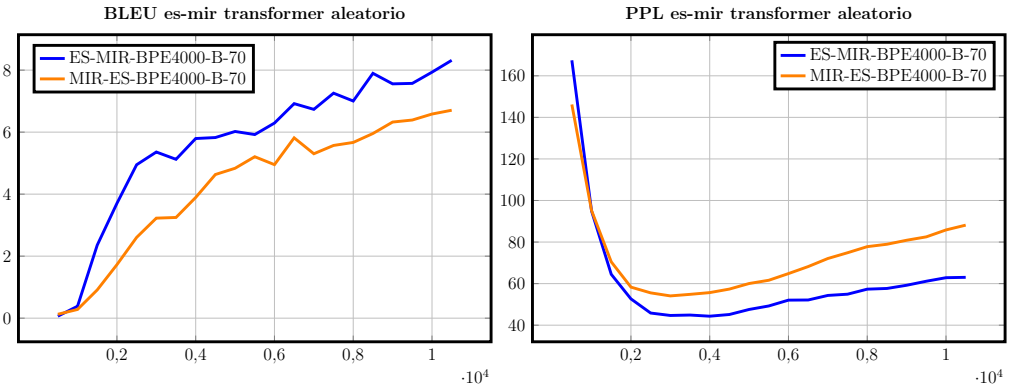


Figura 4.3: Perplejidad y BLEU en el conjunto de desarrollo del entrenamiento *es-mir* y *mir-es* utilizando la configuración (B) con 250 épocas.

se tiene más ganancia en el modelo con más capas (B), esto no es trivial ya que se cuenta con una pequeña cantidad de datos de entrenamiento. Por otro lado, la división *estricta* como se esperaba muestra que es muy difícil de traducir, las puntuaciones BLEU son mínimas. Sin embargo, con las divisiones aleatorias, las puntuaciones BLEU son más prometedoras. También se observó que con la configuración (B) es más "fácil" traducir del español al *Ayuuk* que en la otra

Configuración A 100 **épocas**	**Estricto** *es-mir*		**Aleatorio** *es-mir*		**Estricto** *mir-es*		**Aleatorio** *mir-es*	
BLEU	**dev**	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**
Longitud máxima 50 BPE 2000	1.72	0.05	1.66	1.71	0.64	0.10	0.91	0.66
Longitud máxima 50 BPE 4000	2.03	0.10	1.21	1.24	1.02	0.16	0.93	0.83
Configuración B 100 **épocas**	**Estricto es-mir**		**Aleatorio** *es-mir*		**Estricto** *mir-es*		**Aleatorio** *mir-es*	
BLEU	**dev**	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**
Max lenght 50 BPE 2000	3.91	0.10	3.59	3.70	2.21	0.41	2.49	2.72
Longitud máxima 50 BPE 4000	5.02	0.13	4.17	4.20	2.33	0.28	2.13	2.23
Longitud máxima 70 BPE 4000	7.58	0.10	5.83	5.56	4.03	0.27	3.64	3.52
Configuración B 250 **épocas**	**Aleatorio** *es-mir*				**Aleatorio** *mir-es*			
BLEU	**dev**		**test**		**dev**		**test**	
Longitud máxima 70 BPE 4000	5.83		5.56		3.64		3.52	

Cuadro 4.1: Puntuaciones BLEU de los entrenamientos con dirección *es-mir* y *mir-es*.

dirección.

Dado los resultados prometedores de la configuración (B) en la división aleatoria, se realizó un experimento más grande con 250 épocas, siguiendo la intuición de que aún no se ha alcanzado el rendimiento correcto con 100 épocas. En la Figura 4.3 se muestra la curva de aprendizaje en el conjunto de desarrollo del entrenamiento en ambas direcciones (*es-mir*, *mir-es*), la parte inferior del Cuadro 4.1 muestra los resultados finales. De acuerdo con los resultados, el entrenamiento con 250 épocas tiene el mismo puntaje BLEU que con 100 épocas, destaca una elevación de perplejidad en el step $4,000$ lo que indica un sobreentrenamiento del sistema.

Finalmente, después de observar el comportamiento del modelo experimentando con las configuraciones (A) y (B), donde el conjunto aleatorio mostró resultados exitosos en los puntajes BLEU (*dev* y *test*), se realizó otro experimento en ambas direcciones con este conjunto bajo la configuración C, donde se reduce a 32 el número de lotes, se mantiene la BPE en $4,000$ y se aumenta a 100 la longitud máxima de frases. Este experimento da como resultado una disminución de perplejidad y un aumento en el puntaje BLEU, pasando de 5.56 a 7.29 para la dirección del español (*es*) al *Ayuuk* (*mir*) y 3.52 a 5.82 para la dirección *Ayuuk* (*mir*) al español (*es*), en la Figura 4.4 y en el Cuadro 4.2 se muestran los resultados.

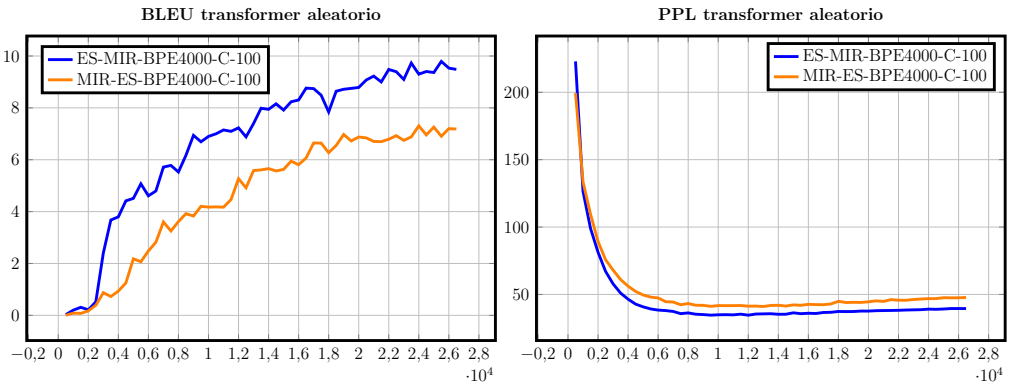


Figura 4.4: Perplejidad y BLEU en el conjunto de desarrollo del entrenamiento *es-mir* y *mir-es* utilizando la configuración (C) con 150 épocas.

Configuración C 150 épocas	**Aleatorio *es-mir***		**Aleatorio *mir-es***	
BLEU	dev	test	dev	test
Longitud max 100 BPE 4000	7.34	7.29	6.18	5.82

Cuadro 4.2: Puntajes BLEU de entrenamientos con la configuración C en dirección *es-mir* y *mir-es*.

4.2.2. Experimento para el traductor *es-mco-es*

Como se menciona en la sección anterior, la configuración D corresponde al traductor para la variante del *Ayuuk* de la región de Coatlán (*mco*). Esta variante solo cuenta con la versión aleatoria en el conjunto de corpus, el tamaño de cada corpus es:

- División aleatoria: *train* 65,071 / *dev* 1000 / *test* 1000

La longitud máxima de frases se mantuvo en 70 y los vocabularios de subpalabras BPE en 4000. Por cada dirección de traducción solo se realizó un experimento con 100 épocas cada uno, en la Figura 4.5 se muestra la curva de aprendizaje de ambas direcciones de traducción.

Como se puede apreciar, los experimentos muestran que la traducción español-*Ayuuk* de Coatlán-español es mejor que para la variante de Güichicovi. El modelo con la configuración (D) presenta un rendimiento muy aceptable, esto se debe principalmente por el tamaño considerable que tiene el corpus de entrenamiento, por lo que las puntuaciones BLEU son muy prometedoras, ver

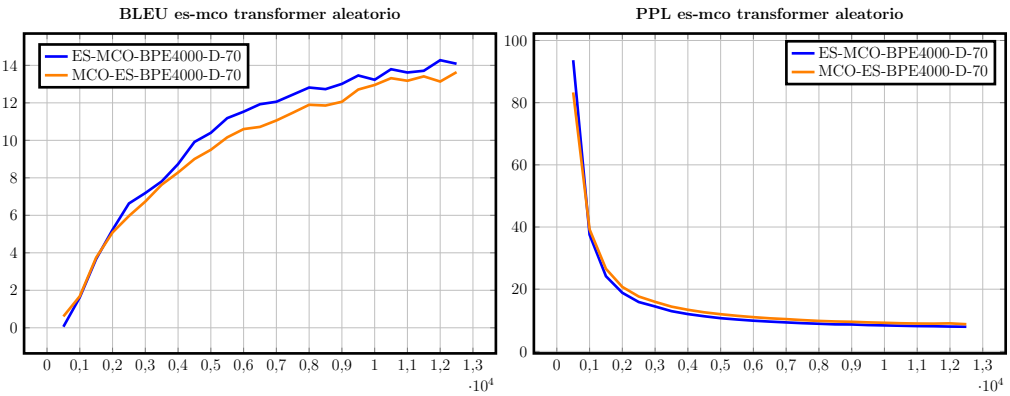


Figura 4.5: Perplejidad y BLEU en el conjunto de desarrollo del entrenamiento *es-mco* y *mco-es* con 100 épocas.

Configuración D 100 épocas	**Aleatorio *es-mco***		**Aleatorio *mco-es***	
BLEU	dev	test	dev	test
Longitud max 70 BPE 4000	15.28	14.64	15.00	13.38

Cuadro 4.3: Puntajes BLEU de entrenamientos con dirección *es-mco* y *mco-es*.

resultados en el Cuadro 4.3.

Por otro lado, las traducciones en ambas direcciones (*es-mco* y *mco-es*) son casi iguales a comparación de las traducciones para la variante del *Ayuuk* de Güichicovi, lo más probable es que se deba a que los corpus de la variante de Coatlán vienen de una sola fuente; es decir, traducciones de los Testigos de Jehová. El modelo para esta variante todavía se puede mejorar pero no se encuentra dentro del alcance de esta tesis, por lo que se deja para trabajos futuros. Sin embargo, los resultados que arroja esta variante entrenada con más de 65,071 frases alineadas sirve de referencia para la variante del *Ayuuk* de Güichicovi, entrenada con menos de 6,000 frases alineadas.

4.3. Traducciones generadas por el modelo

Los puntajes BLEU y perplejidad dan una idea de cómo pueden ser las traducciones candidatas que proporcionan el mejor modelo de traducción generado hasta el momento.

En esta sección se muestran algunos resultados de traducción que genera el modelo entrenado con la configuración (C). Las frases de entrada se escogieron de manera aleatoria dentro del corpus *test*. Asimismo, en cada cuadro de resultados la fila *origen* contiene la frase que se quiere traducir, la fila *objetivo* la traducción correcta en la lengua destino y la fila *candidato* la traducción generada por el traductor automático.

Traducción Español - *Ayuuk* de Güichicovi	
Origen	todos decían que soy malo
Objetivo	age nëm ajxy myana'any ko ëëtsy n'ëxëëgya'ayë
Candidato	a nëje'e ajxy ënajty myënaambë te'emjyëdu'un kyë'ëxë'ëky
Origen	entonces el presidente municipal
Objetivo	ja jëyëjpë mëj kuduunk kopk'ajpë
Candidato	mënit ja jëyëjpë mëj kuduunk kopk'ajpë
Origen	ustedes me vieron ayer en el mercado
Objetivo	mijts axëëy xyijx jim ma too'ktaaktën
Candidato	xyijxëtsy mijts axëëy ma too'ktaaktën
Origen	le dijeron señor, ven y ve
Objetivo	nëm ajxy nyëmaay mëj windsën, jam ukte'emy'ix
Candidato	nëm ajxy y'adsooy mëj windsën, weenëtsy n'ijxë'ëky
Origen	los estados de la federación conservan la extensión y límites que hasta hoy han tenido, siempre que no haya dificultad en cuanto a éstos
Objetivo	mëdu'untyë estados ajxy yajpaady ma ja mexico naax jootyën, je'enë ja naax tsëwa'and ajxy myëëdë neby ajxy jëda'aty nyë'ëgëdëmëëdën, nëgoo mëbëjnëbë ajxy kya'ayajtsip kya'ayajma'adë'ëwët
Candidato	ja kuduunk kugapxy ajxy mexico naax jooty, ëdaa ajxy je'e këxyë wiinë tsobaatp ku ja tyuunk kyapxy ënajty oy oyë të kya'atuny

Cuadro 4.4: Algunas traducciones candidatas *Ayuuk* (*mir*) generadas por el traductor automático.

Los cuadros 4.4 y 4.5 muestran las traducciones candidatas generadas por el modelo de traducción automática de textos para el *Ayuuk* de Güichicovi. El Cuadro 4.6 muestra las traducciones candidatas generadas por el modelo de traducción para el *Ayuuk* variante de la región de Coatlán.

Traducción *Ayuuk* de Güichicovi - Español	
Origen	nëm ajxy nyëmaay mëj windsën, jam ukte'emy'ix
Objetivo	le dijeron señor, ven y ve
Candidato	y ellos le dijeron señor,
Origen	jim jaa koy y'aame'naay
Objetivo	el conejo estaba escondido
Candidato	estaba el conejo
Origen	nëm ja ya'ay ajxy y'adsooy ku ëdaa ya'eay ëxyëp kya'aku'uwandëyjëya'ayë, kap ëjts miitsy ëxyëp të nyajkë'ëdëgë'ëy
Objetivo	respondieron y le dijeron si éste no fuera malhechor, no te lo habríamos entregado
Candidato	ellos le respondieron si fuere necesario que no hagas
Origen	ja jëyëjpë mëj kuduunk kopk'ajpë
Objetivo	entonces el presidente municipal
Candidato	al presidente de la república
Origen	nëmë ja ya'ay nyëmaagyumbë ko oy yajpaady
Objetivo	me dijo que se encuentra bien
Candidato	y él dijo hombre, que el hijo estaba enfermo

Cuadro 4.5: Algunas traducciones candidatas español (*es*) generadas por el traductor automático.

Traducción Español - *Ayuuk* de la variante de Coatlán	
Origen	¿ qué esperanza tenemos ?
Objetivo	¿ ti di'ib ni'amukë n'awijx njëjp'ijxëm ?
Candidato	¿ ti n'awijx njëjp'ijxëm ?
Origen	también tenemos el ejemplo del escritor del salmo
Objetivo	nan xymyo'oyëmë oybyë ijxpajtënë ja dios mëduumbë di'ib jyaayë salmo
Candidato	nan nmëdäjtëmë oybyë ijxpajtën mä salmo
Traducción *Ayuuk* de la variante de Coatlán - Español	
Origen	pes xëmë xymyo'oyëm di'ib meerë nyajtëgoy'äjtëm mä tu'uk tu'ugë jotmay
Objetivo	siempre nos da justo lo que necesitamos para superar cada problema
Candidato	él siempre nos da lo que es justo en cada situación
Origen	ets kom oyjya'ay , ta tpudëkë ënä'ktëjk ets pënaty näämnëm tëjkëp ëxpëjkpë
Objetivo	como es hospitalario , anima a los jóvenes y a los nuevos
Candidato	como es bueno , los jóvenes ayudan a los jóvenes y a los que están dispuestos

Cuadro 4.6: Traducciones español(*es*) y *Ayuuk* de Coatlán (*mco*) generadas por el traductor automático.

4.4. Discusión

En la sección anterior se vió el proceso de experimentación para encontrar un modelo de traducción que se ajuste a los recursos con los que cuenta la lengua *Ayuuk* de Güichicovi, así como un breve experimento para el *Ayuuk* de la variante de Coatlán.

Con los resultados obtenidos se prueba la hipótesis y se responden las pre-

guntas de investigación de este trabajo. Los experimentos demostraron que los sistemas de TA del estado del arte, basados en redes neuronales, son lo suficientemente adecuados para crear traducciones entre la lengua *Ayuuk* de Güichicovi al español, y viceversa. Estos modelos del estado del arte se pueden adaptar de acuerdo a los recursos que se tienen para tener un desempeño exitoso. Además, con los puntajes BLEU y perplejidad obtenidos, se generó un resultado base (`Baseline`) que será referencia para trabajos futuros para esta variante del *Ayuuk*.

De acuerdo con el INALI (2009), el *Ayuuk* de la variante del Güichicovi y el *Ayuuk* de la variante de Coatlán están clasificados dentro del *mixe bajo* (ver Cuadro 1.2). Por esta razón se probó el traductor *mir-es*, pasándole como entrada frases que corresponden al *Ayuuk* de la variante de Coatlán, el resultado de la traducción se puede ver en el Cuadro 4.7. De igual manera, se realizó el mismo experimento para el traductor *mco-es* pasándole como entrada frases del *Ayuuk* de la variante de Güichicovi.

TraducciÓn *mco* dentro del sistema de traducción *mir-es*	
Origen	¿ ti di'ib ni'amukë n'awijx njëjp'ijxëm ?
Objetivo	¿ qué esperanza tenemos ?
Candidato	¿qué nueva esa casa?
Origen	ok, ta jyam'äjty tu'ugë jotmay mä naymyujkën
Objetivo	posteriormente surgió un problema
Candidato	justo en la hora
Traducción *mir* dentro del sistema de traducción *mco-es*	
Origen	age nëm ajxy myana'any ko ëëtsy n'ëxëëgya'ayë
Objetivo	todos decían que soy malo
Candidato	mantenga la paz
Objetivo	ja tsyiit tyats ja uk ëyë'p ajxy xim
Destino	el gato y el perro están jugando
Candidato	el cargo de la ciudad

Cuadro 4.7: Prueba de traducción del *Ayuuk*-español, introduciendo una variante distinta al destinado.

Los resultados muestran que por el momento, no es posible traducir una variante diferente para la que ha sido entrenado el traductor, ni aunque pertenezcan (según el INALI), a la misma variante lingüística, como el caso del *Ayuuk* de Güichicovi y Coatlán, que están clasificadas dentro del *mixe bajo*. Las principales razones que justifican lo anterior, remiten a la falta de datos de

entrenamiento y diferencias en estilo de escritura en la variante de Güichicovi; aunque en la variante de Coatlán hay más datos, falta recopilar textos de otras categorías, el número de vocales es otra barrera, puesto que, como se puede ver en el Cuadro 4.7, la escritura de la variante de Coatlán hace uso de la vocal *ä*, ausente en la variante de Güichicovi.

Por otro lado, la calidad de la mayoría de las traducciones de frases en el traductor automático creado son erróneas, esto tiene que ver con los pocos datos de entrenamiento que se tienen actualmente y la escritura no estandarizada. Sin embargo, en un futuro donde exista un número mayor de datos, se podrá realizar un análisis morfológico más profundo, lo que llevará a crear nuevas herramientas para mejorar el modelo de traducción en la lengua *Ayuuk* de la variante de Güichicovi.

Conclusiones y trabajos futuros

En este trabajo se creó el primer sistema de traducción automática de textos para la lengua *Ayuuk*, variante del municipio de San Juan Güichicovi, al español y viceversa. Donde se sostiene como hipótesis, que es posible adecuar los sistemas de traducción automática del estado del arte, basados en redes neuronales; para crear traducciones entre las lenguas antes mencionadas. Sin embargo, construir por primera vez un sistema de TA en un escenario con escasos recursos y sin antecedente de trabajo relacionado con dicha lengua, implica distintas y complicadas etapas de desarrollo, desde la recolección de datos hasta el entrenamiento del sistema, dando lugar a conclusiones específicas en cada etapa.

La etapa de trabajo de campo para la recolección de textos fue muy fructífera, debido a que hubo participación de personas que están inmersas en la traducción y preservación de la lengua *Ayuuk* de la variante trabajada en esta tesis. Con la buena respuesta por parte de los traductores se logró juntar archivos de diferentes categorías (religión, leyes, literatura,...), ver Cuadro 2.5.

Sin embargo, la mayoría de lo recolectado está protegido por derechos de autor y solamente algunos tienen licencia libre, a pesar de eso, es un buen punto de partida para promover que la creación de obras futuras sean abiertas y con ello lograr que personas hablantes y no hablantes interesados en la lengua, tengan acceso a estos materiales. Liberar estos recursos permitirá que la comunidad pueda aportar mejoras, que no solo beneficiarían la creación de un corpus de libre acceso para trabajos futuros en PLN en esta lengua, sino también en otros ámbitos como el educativo, con la generación de materiales didácticos para el aprendizaje.

Encontrar textos alineados a nivel de frases entre una lengua aglutinante como el *Ayuuk* y un lengua fusionante como el español, es difícil. En lo que

respecta a la alineación automática de textos extraídos de las traducciones recolectadas entre el *Ayuuk* y español, se obtuvieron resultados satisfactorios con la herramienta de alineación *Yasa* (Lamraoui y Langlais, 2013), no obstante, fue necesaria y primordial la verificación manual de las alineaciones para corroborar que las frases entre ambas lenguas tuvieran correspondencia. En este sentido, a través del trabajo realizado se reitera la importancia de que las alineaciones automáticas deben ser verificadas por hablantes que tengan conocimientos de lecto-escritura de la lengua que se está trabajando, a fin de tener mayor certeza de las correspondencias entre frases, ya que los puntajes ofrecidos por *Yasa* no son suficientes y es necesaria una revisión manual para lograr una adecuada alineación. Para futuras alineaciones automáticas se seguirá trabajando en esta dirección.

El corpus utilizado tuvo que ser creado desde cero porque antes de este trabajo no se tenía registro de un corpus para la variante del *Ayuuk* de Güichicovi, aunque en la plataforma JW300 hay un corpus de libre acceso, éste corresponde al *Ayuuk* de la región de Coatlán. Uno de los problemas que persiste en los textos de la lengua *Ayuuk* de la variante de Güichicovi, es la escasez de recursos y el limitado acceso a ellos, así como la variación en la escritura ("petakera" y "bodeguera") a pesar de pertenecer a la misma variante, lo que dificultó el proceso de normalización.

La forma en que se atacó este problema fue retomando el artículo de Sagi-Vela González (2019), donde se rescatan los acuerdos a los que han logrado llegar ambas posiciones para tratar de unificar la escritura, con base en esa información se logró crear una función para normalizar la escritura e intentar recuperar el mayor número de palabras que variaban en la forma en que estaban escritas. Es conveniente mencionar que para el proceso de normalización durante la construcción de futuros corpus, se seguirán respetando las posiciones del estilo de escritura.

La tokenización por subpalabras utilizando el algoritmo BPE tiene la ventaja de generar combinaciones no vistas en el corpus de entrenamiento, en Sennrich y Zhang (2019) se menciona que este algoritmo puede mejorar significativamente resultados de traducción para lenguas de escasos recursos, por ese motivo, la tokenización por subpalabras fue implementada sobre los corpus utilizados en este trabajo, sin embargo, en un futuro donde existan más

datos y con ello un análisis más profundo de la lengua, podría elaborarse algún segmentador morfológico que sin duda alguna, beneficiaría el modelo de traducción.

En experiencias anteriores no había sido prometedor el empleo de modelos de traducción automática basadas en arquitecturas de aprendizaje profundo, particularmente con la configuración *seq2seq*, para las lenguas nativas de las Américas (Mager y Meza, 2018); particularmente porque habían pocos o ningún dato de entrenamiento. Sin embargo, este trabajo muestra que un modelo estándar basado en la arquitectura Transformer y aún con una configuración limitada en recursos, puede producir resultados prometedores. Estos resultados todavía son bajos comparados con los estándares normales en el campo de la TA[101], pero a pesar de ello, los puntajes BLEU obtenidos son buenos indicadores para un futuro en el que se cuente con más datos para la realización de este trabajo.

En lo que refiere a los resultados obtenidos en los experimentos, se puede ver que es más fácil traducir del español al *Ayuuk* de Güichicovi, que de manera inversa, debido a que el español ya cuenta con una escritura estandariza; mientras que los resultados con la lengua *Ayuuk* derivan de los pocos datos de entrenamiento que se tiene hasta el momento, así como de las marcadas diferencias de escritura entre los "bodegueros" y "petakeros" utilizado en el corpus de entrenamiento.

Además, se observó que las frases especializadas del *Archivo Nacional de lenguas indígenas de México* podrían establecer una buena referencia para la evaluación del progreso y rendimiento del sistema de traducción en trabajos futuros. Por último, en un futuro no se descarta la posibilidad de experimentar con traducciones sintéticas generadas a partir de un modelo de traducción con exitosos puntajes BLEU y perplejidad.

Las principales contribuciones de esta tesis son las siguientes:

- Recolección, digitalización y alineación de textos de diferentes fuentes para las cuales había una traducción entre *Ayuuk* de la variante de Güi-

[101] Algunas puntuaciones BLEU de experimentos de traducción de lenguas africanas se pueden consultar en la siguiente liga https://github.com/masakhane-io/masakhane-mt/tree/master/benchmarks

chicovi y *español*.

- Creación de corpus para el entrenamiento y evaluación del sistema de traducción *mir-es-mir*.
- Creación del primer traductor automático de textos *mir-es-mir*, así como la generación de un resultado base (*baseline*) que será la referencia para trabajos futuros.
- Creación de un repositorio[102] donde se puede consultar el código base de este trabajo, resultados de entrenamiento y la parte del corpus que está disponible bajo licencia libre.
- Código para descargar, procesar y obtener los corpus *train/dev/test* de los datos disponibles en JW300 del *Ayuuk* de la variante de la región de Coatlán (*mco*).

Asimismo, con la finalidad de enriquecer este trabajo, se han publicado y presentado en congresos, los siguientes articulos en torno al tema:

- Artículo
  Zacarías, D. y Meza, I. : Ayuuk-Spanish Neural Machine Translator. *Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the America*. pp. 168-172. 2021. (Zacarías Márquez y Meza Ruiz, 2021)
- Artículo
  Zacarías, D. y Meza, I. : Traductor Automático Neuronal Ayuuk-Español. *Research in Computer Science special issue: Proceedings of COMIA*. pp. 10. 2021.
- Tutorial/Plática
  Traducción automática para lenguas originarias de México. *Encuentro Nacional de Computación 2021* realizado el 09 de agosto de 2021.

En resumen, para mejorar el rendimiento del sistema de traducción, el trabajo futuro se centrará en:

[102] https://github.com/DelfinoAyuuk/corpora_ayuuk-spanish_nmt

1. Recopilar más datos, especialmente teniendo en cuenta las diferentes variantes de la lengua *Ayuuk*. Hasta ahora en este trabajo se abordó una variante específica, pero existen múltiples variantes que también carecen de una ortografía estandarizada.

2. Aunque la configuración *estricta* penaliza fuertemente al sistema, las frases especializadas podrían establecer una buena referencia para evaluar el progreso y el rendimiento del sistema de traducción automática, por lo que se seguirá evaluando bajo esta configuración.

3. En este trabajo la tokenización fue basada en subpalabras, sin embargo, un enfoque basado en un análisis morfológico más profundo podría llegar a beneficiar el modelo de traducción (Kann, Mager Hois, Meza-Ruiz, y Schütze, 2018).

4. El proceso de normalización seguirá respetando las posiciones del estilo de escritura de los "petakeros" y "bodegueros", así como el número de vocales que tiene cada variante.

5. La escasez de datos es un poco alarmante, por lo que una generación sintética de corpus para aumentar el conjunto de datos podría beneficiar al sistema de traducción, siempre y cuando el conjunto de datos de entrenamiento del sistema que genere las traducciones tenga un tamaño considerable[103].

Finalmente, es necesario mencionar que para hacer que este proyecto crezca y tenga alcance con otras variantes del *Ayuuk*, es indispensable que más personas nativas hablantes de la lengua se involucren en este tipo de proyectos, no necesariamente en TA, pero sí desde sus ámbitos posibles como lo es la traducción de frases en texto o audio, creando material didáctico para fomentar la lecto-escritura, páginas web para dar a conocer la cultura y la lengua, digitalizar y crear obras bajo licencia libre para que sean más accesibles a la comunidad de hablantes y no hablantes que esten interesados en la lengua, entre otros. Sin olvidar que actualmente la llegada y el rápido crecimiento de las TIC's ha impactado muchas áreas de la sociedad, y es urgente incorporarlas y aprovecharlas en el trabajo de rescate y revitalización de las lenguas indígenas.

[103]Un tamaño considerable para lenguas de escasos recursos sería aproximadamente 500,000 frases emparejadas.

Bibliografía

3Blue1BrownEspañol. (2020). *Los Cálculos de la Retropropagación | Aprendizaje Profundo. Capítulo 4.* https://www.youtube.com/watch?v=CyPjDPKtycM&list=LL&index=3.

Agić, Ž., y Vulić, I. (2019). JW300: A Wide-Coverage Parallel Corpus for Low-Resource Languages. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3204–3210). Florence, Italy: Association for Computational Linguistics. https://www.aclweb.org/anthology/P19-1310. doi: 10.18653/v1/P19-1310

Álvarez, S. M., Paoletti, M. E., Hurtado, J. M. H., Gallego, J. A., Plaza, J., y Martín, J. D. (2019). *Evaluación de Rendimiento del Entrenamiento Distribuido de Redes Neuronales Profundas en Plataformas Heterogéneas.*

Bahdanau, D., Cho, K., y Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate.* https://arxiv.org/abs/1409.0473. doi: 10.48550/ARXIV.1409.0473

Bengio, Y., Ducharme, R., Vincent, P., y Janvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Berger, A., Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Gillett, J. R., Lafferty, J., ... Ures, L. (1994). The Candide system for machine translation. En *Human language technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994.*

Bhattacharyya, P. (2015). *Machine translation.* CRC Press. doi: https://doi.org/10.1201/b18004

Bojanowski, P., Grave, E., Joulin, A., y Mikolov, T. (2017). *Enriching Word Vectors with Subword Information.* https://arxiv.org/abs/1607.04606. doi: 10.48550/ARXIV.1607.04606

Bragagnini Mendizabal, C. M. (2019). *Traducción Automática del Español al Inglés usando Redes Neuronales Profundas con Información Conceptual de Sentencias.* Tesis de Maestría. Universidad Católica San Pablo, Perú.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., ... Roossin, P. S. (1990). *A Statistical Approach to Machine Translation* (n.º 2). Vol. 16. (pp. 79–85). Computational Linguistics.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., y Mercer, R. L. (1993). *The Mathematics of Statistical Machine Translation: Parameter Estimation* (n.º 2). Vol. 19. (pp. 263–311). Computational Linguistics. https://www.aclweb.org/anthology/J93-2003.

Brunning, J. J. J. (2010). *Alignment models and algorithms for statistical machine translation.* Doctoral thesis. University of Cambridge.

Canziani, A. (2020). *Atención y el Transformador.* Aprendizaje Profundo. Consultado el 20 de julio de 2021. https://atcold.github.io/pytorch-Deep-Learning/es/week12/12-3/.

Cavar, M., Cavar, D., y Cruz, H. (2016, may). Endangered Language Documentation: Bootstrapping a Chatino Speech Corpus, Forced Aligner, ASR. En *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016).* Paris, France: European Language Resources Association (ELRA).

Cho, K., van Merrienboer, B., Bahdanau, D., y Bengio, Y. (2014). *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.* https://arxiv.org/abs/1409.1259. doi: 10.48550/ARXIV.1409.1259

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., y Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* https://arxiv.org/abs/1406.1078. doi: 10.48550/ARXIV.1406.1078

Dash, N. S. (2008). Corpus Linguistics: An introduction. *Encyclopedia of Life Support System (EOLSS).*

Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* https://arxiv.org/abs/1810.04805. doi: 10.48550/ARXIV.1810.04805

Dot-CSV. (2018). *¿Qué es una Red Neuronal? Parte 3.5 : Las Matemáticas de Backpropagation | DotCSV.* https://www.youtube.com/watch?v=M5QHwkkHgAA&t=912s.

EAMT. (2008). *What is Machine Translation?* European Association for Machine Translation. Consultado el 26 de octubre de 2020. http://www.eamt.org/mt.php.

Feist, T., y Palancar, E. (2015). *Oto-Manguean Inflectional Class Database.* University of Surrey.

Gelbukh, A. (2007). Estado de la investigación. En *Memoria del I Simposio Internacional sobre Organización del Conocimiento: Bibliotecología y Terminología* (p. 337).

Ghojogh, B., y Ghodsi, A. (2020, 12). *Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey.* doi: 10.31219/osf.io/m6gcn

Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. *Synthesis Lectures on Human Language Technologies*, 309. doi: 10.2200/S00762ED1V01Y201703HLT037

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep Learning.* MIT Press. http://www.deeplearningbook.org.

Gutierrez-Vasques, X. (2015, junio). Bilingual lexicon extraction for a distant language pair using a small parallel corpus. En *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop* (pp. 154–160). Denver, Colorado: Association for Computational Linguistics. doi: 10.3115/v1/N15-2021

Gutierrez-Vasques, X., Sierra, G., y Hernandez, I. (2016, 05). *Axolotl: a Web Accessible Parallel Corpus for Spanish-Nahuatl.* LREC.

Helcl, J., y Libovický, J. (2017). *Neural Monkey: An Open-source Tool for Sequence Learning* (n.º 107). Prague, Czech Republic: The Prague Bulletin of Mathematical Linguistics. http://ufal.mff.cuni.cz/pbml/107/art-helcl-libovicky.pdf. doi: 10.1515/pralin-2017-0001

Hochreiter, S., y Schmidhuber, J. (1997, 12). Long Short-Term Memory. *Neural computation*, *9*, 1735-80. doi: 10.1162/neco.1997.9.8.1735

Hutchins, J. (1982). The evolution of machine translation systems/w. john hutchins. *Practical experience of machine translation, V. Lawson (ed.) North-Holland Publishing Company/© ASLIB*.

Hutchins, J. (2006). Machine Translation: History. *Encyclopedia of Language and Linguistics*, *7*, 375-383.

Hutchins, W. J. (2001, January). Machine Translation over fifty years. *Histoire Épistémologie Langage*, *23*, 7-31. doi: 10.3406/hel.2001.2815

Hutchins, W. J., y Somers, H. L. (1992). *An introduction to machine translation* (Vol. 362). Academic Press London.

INALI. (2009). *Catálogo de las lenguas indígenas nacionales: Variantes lingüísticas de México con sus autodenominaciones y referencias geoestadísticas.* (Primera ed.). México, D.F.: Instituto Nacional de Lenguas Indígenas.

INEGI. (2020). *Censo de Población y Vivienda 2020.* Instituto Nacional de Estadística y Geografía. Consultado el 10 de febrero de 2021. https://www.inegi.org.mx/programas/ccpv/2020/default.html#Tabulados.

INPI. (2017). *Etnografía del pueblo mixe de Oaxaca (ayuukjä'äy).* Instituto Nacional de los Pueblos Indígenas. Consultado el 12 de marzo de 2020. https://www.gob.mx/inpi/articulos/etnografia-del-pueblo-mixe-ayuukja-ay.

Joulin, A., Grave, E., Bojanowski, P., y Mikolov, T. (2016). *Bag of Tricks for Efficient Text Classification.* Vol. abs/1607.01759. CoRR. http://arxiv.org/abs/1607.01759.

Juan Ampuero, A. (2008). *Lengua Española Morfología: Los monemas.* http://mimosa.pntic.mec.es/ajuan3/lengua/monemas.htm.

Jurafsky, D., y Martin, J. (2019). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Draft. https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

Kann, K., Mager Hois, J. M., Meza Ruiz, I. V., y Schütze, H. (2018, junio). Fortification of Neural Morphological Segmentation Models for Polysynthetic Minimal-Resource Languages. En *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 47–57). New Orleans, Louisiana: Association for Computational Linguistics. doi: 10.18653/v1/N18-1005

Kann, K., Mager Hois, J. M., Meza-Ruiz, I. V., y Schütze, H. (2018, junio). Fortification of Neural Morphological Segmentation Models for Polysynthetic Minimal-Resource Languages. En *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 47–57). New Orleans, Louisiana: Association for Computational Linguistics. https://www.aclweb.org/anthology/N18-1005. doi: 10.18653/v1/N18-1005

Klein, G., Kim, Y., Deng, Y., Nguyen, V., Senellart, J., y Rush, A. M. (2018). *OpenNMT: Neural Machine Translation Toolkit.* https://arxiv.org/abs/1805.11462. doi: 10.48550/ARXIV.1805.11462

Koehn, P. (2010). *Statistical Machine Translation.* Cambridge University Press. http://statmt.org/book/.

Koehn, P. (2017). *Neural Machine Translation.* Vol. abs/1709.07809. CoRR. http://arxiv.org/abs/1709.07809.

Koehn, P., y Knight, K. (2003). *(slides) What's New in Statistical Machine Translation.* http://www.cs.jhu.edu/~phi/.

Koehn, P., Och, F. J., y Marcu, D. (2003). Statistical Phrase-Based Translation. En *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 127–133).

Kreutzer, J., Bastings, J., y Riezler, S. (2019, noviembre). Joey NMT: A Minimalist NMT Toolkit for Novices. En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations* (pp. 109–114). Hong Kong, China: Association for Computational Linguistics. https://www.aclweb.org/anthology/D19-3019. doi: 10.18653/v1/D19-3019

Lamraoui, F., y Langlais, P. (2013). Yet another fast, robust and open source sentence aligner. time to reconsider sentence alignment. *XIV Machine Translation Summit*.

Li, P. (2013, 07). A Survey of Machine Translation Methods. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, *11*. doi: 10.11591/telkomnika .v11i12.2780

Lyon, D. D. (1980). *Mixe de Tlahuitoltepec, Oaxaca, Archivo de Lenguas Indígenas de México*. México: Colegio de México.

Mager, M., Carrillo, D., y Meza, I. (2018). Probabilistic Finite-State morphological segmenter for Wixarika (huichol) languague. *J. Intell. Fuzzy Syst.*, *34*, 3081-3087.

Mager, M., Gutierrez-Vasques, X., Sierra, G., y Meza-Ruiz, I. (2018, agosto). Challenges of language technologies for the indigenous languages of the Americas. En *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 55–69). Santa Fe, New Mexico, USA: Association for Computational Linguistics.

Mager, M., y Meza, I. (2018). Hacia la traducción automática de las lenguas indıgenas de méxico. *Proceedings of the DH*.

Mager Hois, J. M. (2017). *Traductor híbrido wixarika - español con escasos recursos bilingües*. Tesis de Maestría, Universidad Autónoma Metropolitana-Azcapotzalco.

Matiss, R. (2019). *Hybrid Machine Translation by Combining Output from Multiple Machine Translation Systems*. Doctoral thesis, University of Latvia.

Maxwell, M., y Bills, A. (2017, marzo). Endangered Data for Endangered Languages: Digitizing Print dictionaries. En *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages* (pp. 85–91). Honolulu: Association for Computational Linguistics. doi: 10.18653/v1/W17-0112

Medina-Urrea, A. (2007). Affix Discovery by Means of Corpora: Experiments for Spanish, Czech, Ralámuli and chuj. En *Aspects of Automatic Text Analysis* (pp. 277–299). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-540-37522-7_13

Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space.* https://arxiv.org/abs/1301.3781. doi: 10.48550/ARXIV.1301.3781

Moreno Cabrera, J. C. (2003). *Síntesis y análisis en las lenguas: crítica de la tipología morfológica clásica y de algunas de sus aplicaciones sincrónicas y diacrónicas.* ELUA. Estudios de Lingüística. N. 17 (2003). ISSN 0212-7636, pp. 465-504.

Mota García, E. (2001). *Segmentación de Imágenes Utilizando Redes Neuronales Recurrentes.* Tesis de Maestría, Centro de Investigación en Matemáticas.

Nekoto, W., Marivate, V., Matsila, T., Fasubaa, T., Fagbohungbe, T., Akinola, S. O., ... Bashir, A. (2020, noviembre). Participatory Research for Low-resourced Machine Translation: A Case Study in African Languages. En *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 2144–2160). Online: Association for Computational Linguistics. https://www.aclweb.org/anthology/2020.findings-emnlp.195. doi: 10.18653/v1/2020.findings-emnlp.195

Och, F., y Ney, H. (2004, 12). The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, *30*, 417-449. doi: 10.1162/0891201042544884

Orozco Camacho, A. M. (2018). *Generación automática de memes de Internet a través de una red neuronal profunda.* Tesis de Licenciatura, Facultad de Ciencias, Universidad Nacional Autónoma de México.

Papineni, K., Roukos, S., Ward, T., y Zhu, W.-J. (2002, julio). Bleu: a method for automatic evaluation of machine translation. En *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (acl)* (p. 311-318). Philadelphia. doi: 10.18653/v1/P19-1310

Pennington, J., Socher, R., y Manning, C. (2014, octubre). GloVe: Global Vectors for Word Representation. En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics. https://www.aclweb.org/anthology/D14-1162. doi: 10.3115/v1/D14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., y Zettlemoyer, L. (2018). *Deep contextualized word representations.* Vol. abs/1802.05365. CoRR. http://arxiv.org/abs/1802.05365.

Poibeau, T. (2017). *Machine translation.* MIT Press.

Reyes Gómez, J. C. (2005). *Aportes al proceso de enseñanza aprendizaje de la lectura y la escritura de la lengua Ayuuk.* Oaxaca, México: Centro de Estudios Ayuuk–Universidad Indígena Intercultural Ayuuk.

Ruder, S. (2016). *An overview of gradient descent optimization algorithms.* Vol. abs/1609.04747. CoRR. http://arxiv.org/abs/1609.04747.

Sagi-Vela González, A. (2019). El mixe escrito y el espejismo del buen alfabeto. *Revista de Llengua i Dret*(71).

Santiago Cayetano, V. (s.f.). *Letrë ajxy yajtuunkpaatypë ku ajxy y'ayuuk jaay: Alfabeto usado para escritura en el mixe del Istmo.* Guía de escritura del mixe del Istmo. Consultado el 15 de marzo de 2020. https://ayuuk.net/es/alfabeto-usado-para-la-escritura-mixe.

Schwartz, L. (2018). *The history and promise of machine translation* (Vol. 18). doi: 10.1075/ata.18.08sch

Sennrich, R., Haddow, B., y Birch, A. (2016, agosto). Neural Machine Translation of Rare Words with Subword Units. En *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (volume 1: Long papers)* (pp. 1715–1725). Berlin, Germany: Association for Computational Linguistics. https://www.aclweb.org/anthology/P16-1162. doi: 10.18653/v1/P16-1162

Sennrich, R., y Zhang, B. (2019, julio). Revisiting Low-Resource Neural Machine Translation: A Case Study. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 211–221). Florence, Italy: Association for Computational Linguistics. https://aclanthology.org/P19-1021. doi: 10.18653/v1/P19-1021

Siebecke, M., y Arvidson, T. (2016). *IBM Model 4 Alignment Comparison: An evaluation of how the size of training data affects the interpretation accuracy and training time for two alignment models that translates natural language.* Stockholm, Sweden: Degree project, KTH Royal Institute of Technology

School of Computer Sciecie and Communication. http://kth.diva-portal.org/smash/get/diva2:931496/FULLTEXT01.

Soriano García, M. S. (2018). *Traductor automático español - purépecha mediante OpenNMT*. Tesis de Licenciatura, Universidad de Guadalajara.

Sutskever, I., Vinyals, O., y Le, Q. V. (2014). *Sequence to Sequence Learning with Neural Networks.* Vol. abs/1409.3215. CoRR. http://arxiv.org/abs/1409.3215.

Thouvenot, M. (2011). *Chachalaca en cen, juntamente: Compendio Enciclopedico del Nahuatl, DVD.* INAH.

Tiedemann, J. (2012, mayo). Parallel Data, Tools and Interfaces in OPUS. En *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)* (Vol. 2012, pp. 2214–2218). European Language Resources Association (ELRA).

Torres Cisneros, G. (2004). *MIXES: Pueblos Indígenas del México Contemporáneo.* (Primera ed.). México, D.F.: CDI:PNUD.

UNAM. (2012). *Gran Diccionario Náhuatl.* Universidad Nacional Autónoma de México. Consultado el 17 de octubre de 2020. http://www.gdn.unam.mx/.

Urrea, A. M., Camacho, J. A. H., y García, M. (2009). *Towards the Speech Synthesis of Raramuri: A Unit Selection Approach based on Unsupervised Extraction of Suffix Sequences.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). *Attention Is All You Need.* doi: 10.48550/ARXIV.1706.03762

Wang, S. (s.f.). *(slides) Transformer Model (1/2): Attention without RNN.* https://github.com/wangshusen/DeepLearning.

Willett, T., Graham, S., Hillman, V., Williams, J., Bautista, M. B., Luría, M. P., ... Eugenio, M. D. C. (2018). *Breve diccionario del mixe del Istmo Mogoñé Viejo, Oaxaca* (Primera ed.). México, D.F.: Instituto Linguistico de Verano.

Zacarías Márquez, D., y Meza Ruiz, I. V. (2021, junio). Ayuuk-Spanish Neural Machine Translator. En *Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas* (pp. 168–172). Online: Association for Computational Linguistics. https://aclanthology.org/2021.americasnlp-1.19. doi: 10.18653/v1/2021.americasnlp-1.19

Zens, R., Och, F., y Ney, H. (2002, 09). *Phrase-Based Statistical Machine Translation.* (p. 18-32). doi: 10.1007/3-540-45751-8_2