Traductor Español – Otomí

1st Lázaro Martínez Abraham Josué
Universidad Nacional Autónoma de México
abrahamlazaro@comunidad.unam.mx

2nd Guerra Silva Erick Iván
Universidad Nacional Autónoma de México
erickivanguerra0.0@gmail.com

3rd Martínez Gutiérrez Carlos Giovanni
Universidad Nacional Autónoma de México
cgiovanni@comunidad.unam.mx

4th Oropeza Castañeda Ángel Eduardo
Universidad Nacional Autónoma de México
angelorocasfi@gmail.com

5th Gamiño González José Luis
Universidad Nacional Autónoma de México
ronluis0905@gmail.com

***Abstract*—**Este documento presenta los resultados obtenidos de los experimentos realizados con la arquitectura de transformer para el procesamiento del lenguaje natural, en específico, para la traducción al otomí. Se utiliza la métrica de BLEU para el análisis de los resultados.

***Índice de Términos*—**Byte-Pair Encoding, traducción, transformer, lenguas de bajos recursos, lenguas indígenas de México, entropía.

I. INTRODUCCIÓN

Para nuestro trabajo, realizamos un traductor basado en tokens con BPE. Para ello, realizamos un path para una tarea de traducción simple, siguiendo los pasos de recolección de datos, limpieza del corpus, segmentación en tokens utilizando BPE usando la entropía como medida de tokenización, a su vez, utilizamos la arquitectura transformer planteada por Vaswani [1], siguiendo la implementación proporcionada por los desarrolladores de OpenNMT [2].

A la arquitectura original se le modificaron distintos parámetros buscando un desempeño bueno de nuestro sistema. También se realizó la generación de Embeddings contextualizados a partir del entrenamiento del transformer, aunque estos ya no se presentan en los resultados.

El trabajo se divide en diferentes secciones para su comprensión. En el marco teórico encontramos información acerca de la tecnología y herramientas que utilizamos, así como información sobre la dificultad de traducir lenguas de bajos recursos. En la sección de diseño experimental hablamos del número de iteraciones para BPE, la métrica de entropía usada, así como los hiper parámetros seleccionados para la arquitectura de transformer. En el análisis de los resultados presentamos y explicamos los resultados obtenidos. En la sección de conclusiones hablamos de las medidas que se pueden utilizar para mejorar nuestro sistema.

II. MARCO TEÓRICO

Traducción para Lenguas de bajos recursos

En los últimos años, el campo de la traducción automática, o Machine Translation (MT) se ha mejorado considerablemente para lenguas con muchos recursos. [3] Entiéndase por recursos como material lingüístico, como, por ejemplo, un conjunto de documentos o corpus, que permite realizar sistemas de procesamiento de lenguaje natural de cierta lengua.

Dada la cantidad de recursos, y el avance en modelos de redes neuronales profundos, se han alcanzado desempeños muy altos. Por ejemplo, para el par alemán-inglés, se alcanza un BLEU de 31.46[4], y para el par francés-inglés, un BLEU de 46.4. [5]

Sin embargo, estas tecnologías se ven alejadas para su uso con lenguas de bajos recursos, ya que los modelos neuronales actuales requieren de muchos datos para ser entrenados. Esto abre el campo de la investigación para que científicos de datos e ingenieros exploren alternativas para poder encontrar métodos que les permitan entrenar un buen traductor con MT contando con pocos recursos. Estos métodos, como la transferencia de conocimiento, no solo permiten que las lenguas de bajos recursos puedan acercarse a la tecnología de traducción, si no, que también permite que otros sistemas mejoren su precisión utilizando el poder de dichas técnicas y todos los recursos que tienen.

Problemas en la Traducción para Lenguas de originarias de México

Un problema para la traducción de lenguas originarias de México es que la mayoría de las lenguas son puramente orales. Al tratar de recolectar los documentos o

corpus de una lengua originaria, nos enfrentamos a la falta de estandarización de dichas lenguas.

La falta de estandarización se debe a muchos problemas sociales, incluso políticos y económicos[6], [7], los cuales no pueden ser resueltos de manera sencilla.

A su vez, la oralidad de las lenguas permite y fomenta una gran cantidad de variaciones de una misma lengua, distribuidas por todo el territorio nacional. En México existen 11 familias lingüísticas, 68 lenguas originarias oficialmente reconocidas con más de 300 variantes[8].

Otro problema también es la intra-variabilidad lingüística entre las lenguas. Por ejemplo, algunas lenguas son aglutinantes y otras son orales.

Problemas en la Traducción para Lenguas de originarias de México

Existen pocos antecedentes de Machine Translation que involucren lenguas originarias de México. Tenemos como primer ejemplo el proyecto Microsoft Translator Community Partners[9] con el Otomí de Querétaro y el Maya de Yucatán. Dicho sistema se puede acceder a través del servicio Bing[10].

Otro ejemplo es la traducción de Wixárika-Español[11], el cual fue desarrollado con un esquema estadístico con la herramienta de Moses. Recientemente se han generado otros ejemplos utilizando redes neuronales profundas, con lenguas como: Mexicanero, Náhuatl, Purépecha, Yorm Nokki, Ayuuk[12] y Quechua[13].

Otros artículos recientes han mostrado como la implementación de transferencia de conocimiento puede resultar beneficioso para la traducción, integrando sistemas más complejos como Speech Translation (ST) para el Náhualt de la zona alta de Puebla[14].

También se han mostrado resultados contundentes aumentando la cantidad de datos para sistemas que trabajan con Náhuatl, pero requiriendo de un gran esfuerzo para la recolección del corpus[15].

Corpus Tsunkua Elotl

La comunidad de Elotl pone a nuestra disposición un corpus paralelo de Otomí – español para tareas del procesamiento del lenguaje natural. Dicho corpus cuenta con poco más de 4 mil oraciones, un número limitado, pero que nos permitirá entender los retos de traducción con bajos recursos[16].

Byte-Pair Encoding

Para la realización de nuestro sistema, generamos la tokenización de nuestro corpus con el algoritmo Byte-Pair Encoding (BPE). Este sencillo algoritmo permite comprimir la información, remplazando los pares de bytes consecutivos más comunes por un byte diferente que no aparece en el corpus. El algoritmo de BPE aproxima de manera muy buena a la codificación Shannon-Fano.

BPE es un algoritmo sencillo que no tiene una métrica establecida para determinar cuantas iteraciones generan una buena representación comprimida del corpus. Para nuestro proyecto, utilizamos la función de entropía:

$$H(X) = -\frac{1}{N}\sum_{i=1}^{|X|} log_2 p(X = x_i)$$

A su vez, también graficamos las curvas generadas de la Lez de Zipf para la comparación de las frecuencias en los corpus (español y Otomí), ya que al tener corpus que tengan una distribución distinta, puede afectar al desempeño del sistema.

Embeddings

La representación vectorial de las palabras es una herramienta que apoya en gran medida a los sistemas de NLP. Los Embeddings son representaciones vectoriales de las palabras que buscan representar a las palabras con menos dimensiones de las que se ocupan al codificar palabras o sub-words con la codificación one hot [17], [18].

A su vez, si los Embeddings se entran al mismo tiempo que el transformer, lo que obtendremos serán representaciones que dependen del lugar donde aparecen las palabras, en este caso, representaciones contextualizadas[19].

Transformer

La arquitectura transformer[1] presentada en 2017 en el paper “Attention is all you need”, presentó una innovadora forma de trabajar con la atención que se desarrolló en redes neuronales recurrentes (RNN) y las redes Long-Short Term Memory (LSTM).

Dicha arquitectura presenta un sistema encoder-decoder multicapa que incorporaba un bloque de atención el cual se podía calcular de manera matricial, haciendo al sistema más rápido y que su entrenamiento fuera más rápido (aunque con mucho procesamiento).[1]

La arquitectura transformer ha revolucionado el mundo de la inteligencia artificial, no solo en el campo del NLP. Su arquitectura es muy complicada para explicar aquí, por lo que se refiere al paper original para saber más del modelo.

Beam Search

Beam Search es un algoritmo para decodificación que se utiliza en procesamiento del lenguaje natural. La intuición del algoritmo es no hacer una búsqueda greedy

tomando solo el elemento con mayor probabilidad en cada iteración o “desenmascaramiento”, si no, tomar múltiples alternativas para realizar una búsqueda óptima[20].

BLEU

BLEU (Bilingual Evaluation Understudy), que traducido es suplente de evaluación bilingüe es una métrica que permite medir las diferencias entre la traducción automática y la traducción humana. El algoritmo detrás de esta métrica compara expresiones consecutivas de la traducción automática con las expresiones consecutivas de la fuente de referencia (traducción humana, por ejemplo) y cuenta las coincidencias entre éstas. A mayor grado de coincidencia implica una mayor similitud con la traducción de referencia y por lo tanto una mejor traducción[21].

III. Diseño Experimental

Byte-Pair Encoding

Para la parte de tokenizar, se utilizaron 70 iteraciones de BPE tanto para Español como para Otomí.

Transformer

La arquitectura transformer que utilizamos consta de 3 capas de encoder y 3 capas de decoder. A su vez, se definió la dimensión de los Embeddings como 64, aunque lo natural es que sean de dimensión 128, se optó por disminuir el tamaño de los Embeddings dada la cantidad reducida de datos con los que se cuenta.

El transformer también implementa una capa feed-forward dentro de cada encoder y decoder. Dichas capas se asignaron con una dimensión de 128.

El número de cabezales de atención para cada capa atencional se estableció en 4. A su vez, se determinaron otros hiper parámetros como el batch size en 30, learning rate en 0.005. Se utilizó un optimizador Noam con un coeficiente de warmup de 4000, un factor de 1, una épsilon de 1E-9, y también como optimizador base se utilizó un optimizador Adam con betas 0.9 y 0.999 respectivamente.

Como función de costo se utilizó una función de entropía cruzada, de la paquetería PyTorch[22].

Todo el sistema fue entrenado utilizando PyTorch, para poder entrenarlo de manera más rápida utilizando la GPU que permite utilizar de manera gratuita el servicio de Google Colab.

El sistema se entrenó con 100 iteraciones, utilizando padding para estandarizar el tamaño de la entrada (oraciones). También, se utilizó una decodificación utilizando el algoritmo de Beam Search con un tamaño de k (seleccionados) de 3.

El sistema fue entrenado con 4500 oraciones, y fue evaluado con 200 oraciones siguiendo la métrica de BLEU implementada por la paquetería sacrebleu.

IV. Resultados

Byte-Pair Encoding

En la generación de los tokens, determinamos que 70 eran las iteraciones necesarias para alcanzar un mínimo de entropía en ambos corpus. En la figura 1, 2 y 3 podemos observar la evolución de entropía para ambos corpus a través del BPE.

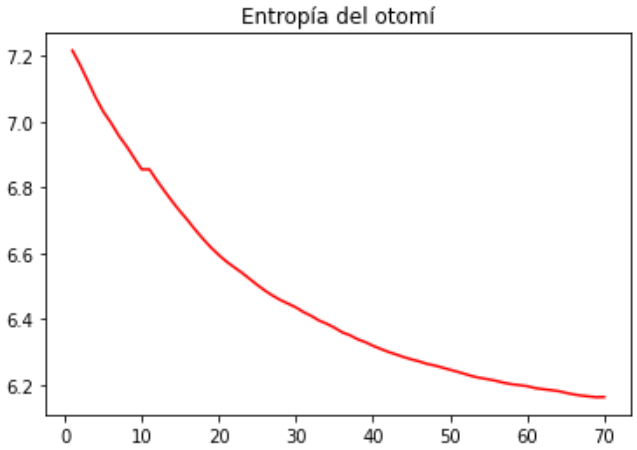

Figura 1. Evolución de la entropía para el Otomí

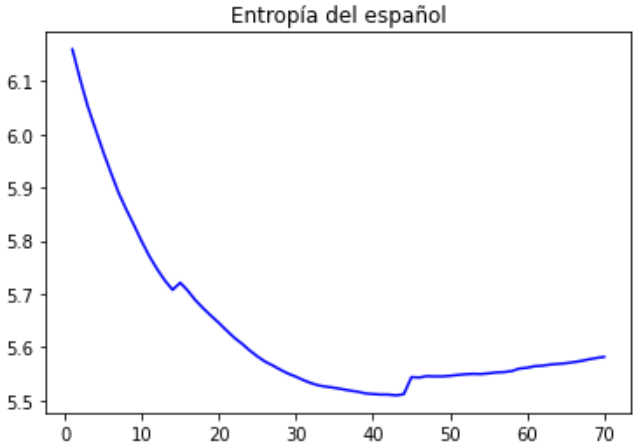

Figura 2. Evolución de la entropía del Español

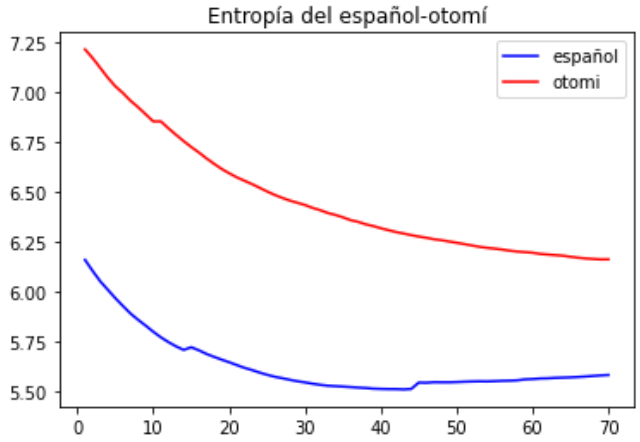

Figura 3. Comparación de la evolución de la entropía de ambos corpus

A su vez, comparamos la distribución de frecuencias de los tokens entre ambos corpus utilizando las curvas generadas a través de la frecuencia de tokens, basados en la Ley de Zipf. En la figura 4 podemos observad dichas curvas.

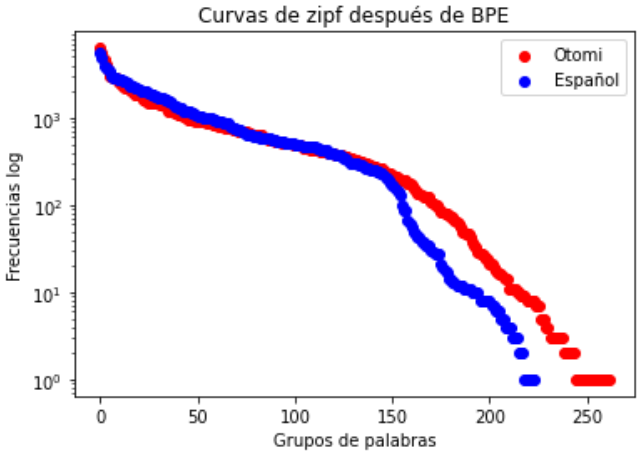

Figura 4. Comparación de distribuciones con la Ley de Zipf

También comparamos nuestra propia implementación con la herramienta Subword-NMT[23], obteniendo valores más bajos para la entropía de los corpus. En la figura 5 podemos observar como el valor de la entropía no disminuye incluso con más iteraciones para los resultados de Subword-NMT.

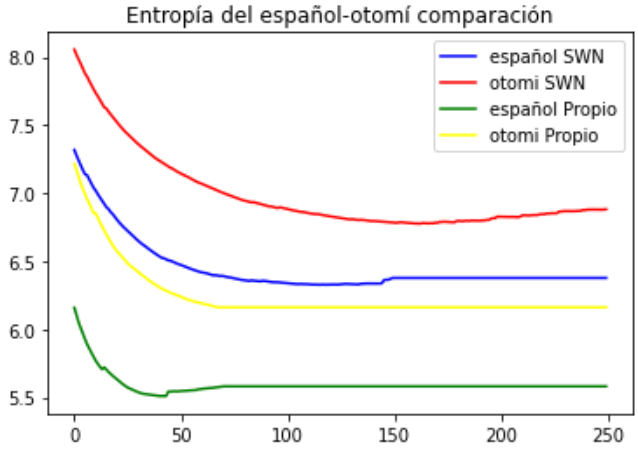

Figura 5. Comparación de nuestra implementación con Subword-NMT

Transformer

Con lo que respecta al modelo de traducción, el sistema tardó 2 horas en entrenarse utilizando un GPU proporcionada por Google Colab. En la figura 6 podemos observar como evolucionó la perdida del sistema conforme se fue entrenando.

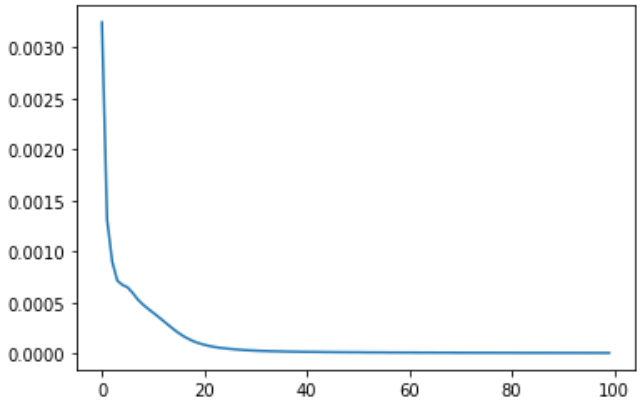

Figura 6. Perdida durante el entrenamiento del modelo

Para la parte final de evaluación, utilizando la métrica de BLEU y 200 oraciones de prueba, el sistema arrojó un valor de BLEU de 0.99, teniendo un desempeño muy pobre respecto al estado del arte de traducción.

Esto era de esperarse, ya que contamos con muy pocos datos, incluso, podemos observar en la gráfica de perdida que la curva indica una convergencia muy temprana, lo cual puede evitar que el modelo aprenda adecuadamente e incluso da indicios de un sobre ajuste en el entrenamiento.

V. CONCLUSIONES

De nuestro trabajo podemos inferir y aseverar varios hechos. Si bien la arquitectura transformer es muy poderosa, al tratar de utilizarla para traducción de bajos recursos, debemos buscar una forma de evitar el sobre ajuste y la

convergencia temprana del modelo, no solo modificando la arquitectura del sistema, si no también tratando de remediar el problema de utilizar tan pocos datos.

Otra cosa importante por destacar es la evidencia de que la arquitectura transformer es delicada, y debe ser tratada con cuidado ya que sobre ajusta muy fácil.

Trabajo a futuro

Ahora comprendiendo la arquitectura transformer, el siguiente paso es modificar nuestro sistema para que pueda tener mejore resultados. Esto se puede lograr de diferentes maneras.

La primera y más trivial, es con la ayuda de más datos. Esto es complicado, ya que existen trabajos para generar corpus de traducción que llegan a tardar años, y tomando en cuenta los problemas descritos en el marco teórico, es una solución trivial con el desarrollo más complicado de las propuestas.

La siguiente propuesta es la de utilizar datos no paralelos para entrenar en una tarea distinta, como predicción, un sistema transformer y después ajustar el modelo para aprender traducción. Esto no es trivial, ya que recientemente se han mostrado muchos estudios en los que la transferencia de conocimiento por medio de entrenamiento en diferentes tareas puede mejorar el performance de un sistema de traducción. En este caso, ya que los datos paralelos para traducción son muy pocos, podemos utilizar recursos no paralelos para entrenar a un transformer a predecir la siguiente palabra en Español y en Otomí, y con el sistema resultante, en específico, con el encoder del sistema resultante, armar una arquitectura de transformer y entrenar para traducción.

Otra posibilidad es aumentar nuestro dataset utilizando una técnica de ventaneo, pero cabe la posibilidad de que el sistema pueda no estar contemplando todo el contexto de una palabra al procesarla, por lo cual debe realizarse un buen estudio antes de realizarlo.

REFERENCIAS

[1] A. Vaswani *et al.*, “Attention Is All You Need.” Google, p. 15, 2017, [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf.

[2] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, “OpenNMT: Open-Source Toolkit for Neural Machine Translation.” 2017, doi: 10.18653/v1/P17-4012.

[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nat. Publ. Gr.*, vol. 521, pp. 436–444, 2015, [Online]. Available: https://s3.us-east-2.amazonaws.com/hkg-website-assets/static/pages/files/DeepLearning.pdf.

[4] S. Takase and S. Kiyono, “Lessons on parameter sharing across layers in transformers.” 2021, doi: https://arxiv.org/abs/2104.06022.

[5] M. Xu, L. Li, D. F. Wong, Q. Liu, and L. S. Chao, “Document Graph for Neural Machine Translation.” p. 14, 2020, doi: https://arxiv.org/abs/2012.03477.

[6] M. Mager, X. Gutierrez-Vasques, G. Sierra, and I. Meza, “Challenges of language technologies for the indigenous languages of the Americas.” p. 15, 2018, doi: https://arxiv.org/abs/1806.04291.

[7] D. Zacarías and I. V. Meza, “Ayuuk-Spanish Neural Machine Translator.” Association for Computational Linguistics, pp. 168–172, 2021, doi: 10.18653/v1/2021.americasnlp-1.19.

[8] INLI, *Catálogo de las Lenguas Indígenas Nacionales*, 1st ed. México, 2009.

[9] Microsoft, “Translator Community Partners.” https://www.microsoft.com/en-us/translator/business/community/ (accessed Dec. 13, 2021).

[10] Microsoft, “Microsoft Bing.” https://www.bing.com/translator (accessed Dec. 13, 2021).

[11] J. M. Mager, C. Barrón, and I. V. Meza, “Traductor estadístico wixarika - español usando descomposición morfológica.” Universidad Inca Garcilaso de la Vega, p. 6, 2016, doi: http://hdl.handle.net/20.500.11818/614.

[12] M. Mager, A. Oncevay, A. Rios, A. Meza Ruiz, Ivan Vladimir Palmer, G. Neubig, and K. Kann, “Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas.” University of Zurich, p. 287, 2021, [Online]. Available: https://www.zora.uzh.ch/id/eprint/203436/1/2021.americasnlp-1.pdf.

[13] G. Uchamaco, H. Calderón, and F. Cárdenas, “Incubación de Sistema de Traducción Automática Español a Quechua, Basado en la Plataforma Libre y Código Abierto Apertium.” p. 9, 2013, [Online]. Available: https://journal.ceprosimad.com/index.php/ceprosimad/article/view/7.

[14] J. Shi, J. D. Amith, X. Chang, S. Dalmia, B. Yan, and S. Watanabe, “Highland Puebla Nahuatl–Spanish Speech Translation Corpus for Endangered Language Documentation.” p. 11, 2021, [Online]. Available: https://aclanthology.org/2021.americasnlp-1.7.pdf.

[15] S. Bello, E. Sánchez, E. Bonilla, J. Hernández, J. Ramírez, and B. Pedroza, “Nahuatl Neural Machine Translation Using Attention Based Architectures: A Comparative Analysis for RNNs and Transformers as a Mobile Application Service.” pringer Nature Switzerland, p. 20, 2021, doi: https://doi.org/10.1007/978-3-030-89820-5_10.

[16] Elotl, “TSU̱NKUA,” *Acerca del corpus*, 2021. https://tsunkua.elotl.mx/about/ (accessed Dec. 13, 2021).

[17] P. Battle, “Clasificación de Texto Financiero: un Análisis de los diferentes Métodos de Embedding de Palabras,” *PROCESAMIENTO DE DATOS*, 2018. https://www.bbvaaifactory.com/es/4062-2/ (accessed Sep. 10, 2021).

[18] J. Collis, “Glossary of Deep Learning: Word Embedding,” *Glossary of Deep Learning*, 2017. https://medium.com/deeper-learning/glossary-of-deep-learning-word-embedding-f90c3cec34ca (accessed Sep. 10, 2021).

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” Google, p. 16, 2019, [Online]. Available: https://arxiv.org/pdf/1810.04805.pdf.

[20] R. Khandelwal, “An intuitive explanation of Beam Search,” *Towards data science*, p. 1, 2020.

[21] Microsoft, “¿Qué es una puntuación BLEU?” microsoft, p. 1, 2021, [Online]. Available: https://docs.microsoft.com/es-es/azure/cognitive-services/translator/custom-translator/what-is-bleu-score.

[22] “PyTorch.” [Online]. Available: https://pytorch.org/.

[23] R. Sennrich, B. Haddow, and A. Birch, “Subword Neural Machine Translation.” 2016, [Online]. Available: https://github.com/rsennrich/subword-nmt.