Evaluating Low-Rank Adaptation for Multilingual Translation in Low-Resource Settings

Víctor Bórquez
vborquezruiz@uc.cl

Vicente Navarro
venavarro@uc.cl

Marcos Santelices
marcos.santelices@uc.cl

Diego Valenzuela
diegovalarq@uc.cl

Abstract

Automatic translation into under-served languages such as Quechua suffer from an acute scarcity of parallel data, hindering the development of reliable machine-translation systems. In this project, we investigate the adaptation of a pretrained 600-M paramater multilingual model (No Languages Left Behind: NLLB-200) to the task of Spanish → Quechua translation using Low-Rank Adapters (LoRA), an efficient technique that trains only 0.18% of the backbone weights. Using a parallel corpus of 128,000 sentence pairs, the model is fine-tuned with a lightweight and reproducible training scheme. Evaluation shows a significant improvement over the zero-shot baseline: our adapted model raises BLEU score from 0.08% to 4.15%, while SequenceMatcher similarity increases from 23.15% to 51.38%. Additionally, further metrics (Levenshtein, semantic cosine using SBERT, Jaccard) and the impact of sentence length are analyzed. These results demonstrate the impact LoRA offers in the efficient adaptation of existing models to improve coverage in low-resource languages, without the need for full retraining.

1 Introducción

En los últimos años, los modelos multilingües preentrenados han demostrado ser altamente efectivos para tareas de traducción automática, especialmente en contextos con abundancia de datos paralelos. Sin embargo, su desempeño se ve limitado al aplicarlos a lenguas de bajos recursos, como el quechua, debido a la escasez de corpus de entrenamiento representativos. Esta carencia de datos afecta tanto la cobertura lexical como la calidad sintáctica y semántica de las traducciones generadas, dejando a estas lenguas marginadas del desarrollo tecnológico actual.

Frente a esta situación, surge la necesidad de estrategias de adaptación eficientes que permitan mejorar la traducción hacia estas lenguas sin requerir el costoso reentrenamiento completo de modelos grandes. Una técnica prometedora en este contexto

es el uso de *Low-Rank Adapters* (LoRA), que permite modificar selectivamente pequeñas porciones del modelo original, manteniendo congelados la mayoría de sus parámetros. Esto reduce significativamente los requerimientos computacionales y permite escalar el proceso a múltiples lenguas minoritarias.

El presente trabajo se enfoca en el caso concreto de la traducción del español al quechua, utilizando como base el modelo NLLB-200, que ya incluye cobertura básica del idioma pero con una calidad catalogada como baja. A través de un *fine-tuning* liviano con LoRA, entrenado sobre un corpus paralelo de 128,000 pares de oraciones, se demuestra que es posible mejorar sustancialmente las métricas de traducción con un costo computacional reducido.

Este artículo se organiza de la siguiente manera. En la Sección 2 se definen las métricas que se estudiarán para monitorear el rendimiento del modelo que se va a implementar. En la Sección 3 se presenta el método propuesto, incluyendo la configuración del modelo, el preprocesamiento de datos y el flujo de trabajo completo. La Sección 4 describe los resultados experimentales y el análisis detallado de métricas de calidad. Finalmente, la Sección 5 resume las principales conclusiones del estudio y plantea líneas de trabajo futuro.

2 Antecedentes

2.1 Aspectos Teóricos

En esta sección se exponen los fundamentos conceptuales necesarios para comprender la propuesta de adaptación de modelos multilingües a la traducción español → quechua mediante Low-Rank Adaptation (LoRA). Se revisan los desafíos propios de los idiomas de bajo recurso y las métricas de evaluación que cuantifican la calidad de la traducción.

2.1.1 Lenguas de Bajos Recursos

Como idioma de bajos recursos, el Quechua carece de corpus públicos extensos a los que se pueda acudir (del orden de 1 millón de pares). Esto representa una limitación al momento de entrenar grandes modelos del lenguaje desde cero, por lo que el uso de técnicas de fine-tuning como LoRA representan una gran oportunidad para abordar la baja representatividad del idioma en el mundo digital.

Como descripción breve del idioma, se puede mencionar que su morfología es de carácter aglutinante. Esto quiere decir que las palabras se forman mediante la adición de varios sufijos a una raíz. En cambio, el español, si bien contiene ciertas características aglutinantes, en su mayoría es un lenguaje sintético. Esta diferencia representa un desafío al momento de entender ambos idiomas y contar con la capacidad de los modelos de aprender patrones lingüísticos y morfológicos para llevar a cabo una traducción comprensiva y exitosa. Además, el quechua presenta variaciones de dialecto a lo largo de los países y regiones en los que se habla en América del Sur, punto que añade un componente de dificultad a la tarea del entrenamiento del modelo frente a la escasez de datos.

2.1.2 BLEU Score

Bilingual Evaluation Understudy es un algoritmo usado para evaluar la calidad de la traducción de máquina, de un lenguaje natural a otro. Como índice, BLEU es una muy buena referencia al momento de determinar qué tan bien traduce lenguaje natural una máquina en comparación a una traducción humana.

Creada por IBM en 2002 [3], este índice mide la precisión de n-gramas (n = 1...4) en el total de n-gramas de una traducción, es decir, cuenta las palabras compartidas entre la predicción de máquina y su referencia humana. El primer paso para obtener este puntaje corresponde a calcular el promedio geométrico de la precisión modificada, de la siguiente forma:

$$\begin{aligned} \textit{Geometric Average Precision } (N) &= \exp\Big(\sum_{n=1}^{N} w_n \log p_n\Big) \\ &= \prod_{n=1}^{N} p_n^{w_n} \\ &= (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}} \end{aligned}$$

con $N = 4$ para los efectos de este proyecto, $w_n = \frac{N}{4}$, y p_n la precisión para cada cantidad de n-gramas.

Por otro lado, existe una penalización por brevedad, que busca incentivar una traducción extensiva y comprensiva, además de prevenir cálculos de precisión erróneos:

$$\text{Brevity Penalty} = \begin{cases} 1, & \text{si } c > r, \\ e^{(1-\frac{r}{c})}, & \text{si } c \leq r. \end{cases}$$

con c el largo del texto predicho, y r el largo del texto objetivo. Por lo tanto, se busca que el texto predicho sea al menos del mismo largo que la traducción de referencia.

Finalmente, el cálculo de BLEU Score se define por la ecuación

$$\begin{aligned} \text{BLEU}(N) &= \text{Brevity Penalty } \cdot \text{ Geometric Average Precision Scores}(N), \\ \text{BLEU} &= \text{BP } \cdot \exp\Big(\sum_{n=1}^{N} w_n \log p_n\Big). \end{aligned}$$

2.1.3 SequenceMatcher

Corresponde a una clase del módulo `difflib` de Python. SequenceMatcher permite buscar coincidencias parciales entre textos a través de la ecuación

$$\text{SM}(x, y) \;=\; \frac{2 \times M(x, y)}{|x| + |y|}$$

con $M(x, y)$ la longitud de la subsecuencia común más larga (Longest Common Subsequence, o LCS), $|x|$ la longitud del texto de predicción y $|y|$ la longitud del texto de referencia.

Por lo tanto, entre mayor su ratio, mayor es la coincidencia general entre la referencia y la predicción. Si alguna palabra se omite, cambia de orden o no es gramáticamente correcta, este ratio disminuye.

2.1.4 Exact Match

Métrica autodescriptiva. Se determina si el texto predicho es exactamente igual al texto de referencia

$$\mathrm{EM}(r,\, c) \;=\; \begin{cases} 1, & \text{si } r = c, \\ 0, & \text{en otro caso.} \end{cases}$$

con r el texto predicho, y c el texto de referencia o control.

2.1.5 Distancia de Levenshtein

Es una métrica de similaridad entre textos, que se enfoca en medir la cantidad de cambios necesarios para convertir una cadena de texto en otra. Para efectos del proyecto, permite determinar si la predicción necesita muchas ediciones para igualar la referencia o no. Los cambios evaluados en este caso son la inserción, eliminaciń o sustitución de un carácter por otro.

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j), & \text{si } \min(i,j) = 0, \\ \min\left\{ \begin{array}{l} \mathrm{lev}_{a,b}(i-1,j)+1, \\ \mathrm{lev}_{a,b}(i,j-1)+1, \\ \mathrm{lev}_{a,b}(i-1,j-1)+\mathbf{1}_{(a_i \neq b_j)} \end{array} \right\}, & \text{en otro caso.} \end{cases}$$

Con a una cadena de texto, b otra cadena de texto, i y j la posición evaluada de estas cadenas, respectivamente.

Ahora bien, para el proyecto se usará la versión normalizada de esta distancia

$$\mathrm{LevSim}(x,y) \;=\; 1 \;-\; \frac{d(x,y)}{|x|+|y|}.$$

que permite evaluar textos de distinta longitud y llevar la métrica al rango $[0,1]$, donde 1 implica que las cadenas son identicas, y 0 implica que todos los caracteres en los textos de predicción deben ser cambiados para igual a la referencia.

2.1.6 Semantic Similarity

Corresponde a una métrica que cuantifica la similitud en **significado** en vez de igualdad literal. La similaridad semántica abre las puertas para un análisis más comprensiva, pues ya no se depende exclusivamente de solamente las palabras traducidas.

El procedimiento se compone, como primer paso, de usar el modelo `paraphrase-multilingual-mpnet-base-v2` de Sentence-Transformer (**SBERT**), el cual convierte cada entrada de texto en un vector de 768 dimensiones, con el fin de codificar su información semántica. En segundo lugar, se calcula la similitud como el coseno entre los vectores de predicción y referencia del modelo.

$$\mathrm{sim}_{\cos}(u,v) = \frac{u \cdot v}{\|u\|\,\|v\|} \;\in\; [-1,1].$$

El rango sugiere que si la s imilitud es 0, no hay relación semántica, si es 1, el significado es exactamente el mismo, y si es -1, sus significados son una contradicción.

2.1.7 Jaccard Similarity

Mide el tamaño de la intersección entre dos conjuntos A y B respecto a su unión.

$$Jacc(A, B) = \frac{|A \cap B|}{|A \cup B|} \in [0, 1]$$

donde el índice se vuelve 1 si los conjuntos son idénticos, y 0 si totalmente distintos.

Para el proyecto se ha hecho uso de la función `textdistance.Jaccard().normalized_similarity` en Python, con capacidades que permiten tokenizar las entradas de texto, construir conjuntos de elementos únicos por entrada, y el cálculo de la métrica de interés.

2.1.8 Dice Coefficient

También denominado Coeficiente de Sørensen-Dice. En una línea similar a Jaccard, Dice cuantifica la similitud entre dos conjuntos A y B.

$$\text{Dice}(A, B) = \frac{2\,|A \cap B|}{|A| + |B|}, \qquad 0 \leq \text{Dice}(A, B) \leq 1.$$

A diferencia de Jaccard, la intersección se multiplica por 2, por lo que este coeficiente suele ser mayor que el primero al evaluar los mismos conjuntos. Para el proyecto se ha usado la implementación `textdistance.SorensenDice()` en Python, que sigue un procedimiento similar a `Jaccard()`.

2.1.9 Cosine Similarity

Bajo una noción similar a Semantic Similarity, Cosine representa cada entrada de cadena de texto como un vector en un espacio altamente dimensional, para luego medir el ángulo entre los dos vectores, es decir, la predicción y la referencia.

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\,\|B\|} = \frac{\sum_{i=1}^{n} A_i\, B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\,\sqrt{\sum_{i=1}^{n} B_i^2}} \;\in\; [-1, 1].$$

con A y B los vectores de cadena de texto de predicción y la referencia, sus módulos las longitudes, e i e iésimo elemento o i-grama de cada vector de texto. Al igual que con Semantic Similarity, si la similaridad se acerca a -1, los vectores tienen un significado totalmente opuesto, mientras que si se acerca a 1, los significados se alinean muy bien.

2.1.10 Overlap Similarity

Finalmente, Overlap, también denominada *Szymkiewicz-Simpson coefficient*, compara dos conjuntos A y B al normalizar su intersección en base a su conjunto más pequeño.

$$\text{Overlap}(A, B) = \frac{|A \cap B|}{\min\big(|A|,\ |B|\big)} \quad \Longrightarrow \quad 0 \;\leq\; \text{Overlap}(A, B) \;\leq\; 1.$$

Para esta métrica, si no existen elementos (palabras) compartidos, el coeficiente es 0, mientras que si el conjunto de menor tamaño se contiene en el más grande, el coeficiente toma valor 1. Overlap puede ser útil para entender la proporción del vocabulario de referencia que logra capturar el modelo afinado.

2.2 Trabajos Relacionados

La traducción automática en contextos de recursos limitados -es decir, cuando se dispone de pocos datos- ha sido objeto de creciente atención en la literatura reciente. Frente a esta limitación, diversas lineas de investigación han explorado cómo adaptar modelos multilingües a nuevos idiomas con pocos ejemplos, o bien cómo aprovechar modelos de lenguaje a gran escala para facilitar la transferencia entre lenguas con escasa representación digital.

En este contexto, el trabajo "**Efficient Adaptation: Enhancing Multilingual Models for Low-Resource Language Translation**" [4] se centra en implementar técnicas para hacer *fine-tunning* del modelo mBART, de manera de reducir en gran cantidad el número de parámetros re-entrenados. Para esto, combinaron LoRA y *Bottelneck adapter*, de manera que solo re-entrenaron el 5% de los parámetros originales. Lo que obtuvieron los autores es que el rendimiento de esta implementación (BLEU = 21.30) es muy parecido a hacer *fine-tunning* re-entrenando completamte (BLEU = 21.95), pero reduciendo en gran cantidad los parámetros.

Por otro lado, el artículo "**QueEn: A Large Language Model for Quechua-English Translation**" [1] realizan *fine-tunning* sobre un LLM multilíngüe como puede ser Llama o un GPT, e implementan una técnica llamada *Retrieval-augmented Generation* (RAC) que combina un modelo generativo con un sistema de recuperación de documentos externo, de modo que el modelo tenga acceso a información adicional relevante durante la generación de texto. Con esto, obtienen un BLEU = 23.50. Esta técnica evita hacer más *fine-tunning*, ya que le entrega conocimiento externo al modelo via *promts* y así mejorar la calidad de las traducciones.

3 Método Propuesto

El objetivo central de este trabajo es mejorar la traducción automática Español → Quechua a partir de un modelo multilingüe pre-entrenado, mediante una adaptación con *low-rank adapters* (LoRA). A continuación se detallan sus componentes principales.

3.1 Modelo base y motivación

Se parte del NLLB-200 distilled-600M [5], un modelo encoder-decoder basado en Transformer que cubre 200 idiomas y tiene 600 M de parámetros. Aunque ya incluye el quechua, la cantidad de datos utilizados durante su entrenamiento es catalogada como *Res: Low*, por lo que su desempeño inicial es limitado. Para adaptar el modelo de forma eficiente se emplea LoRA [2]. Solo se inyectan matrices de rango $r = 8$ en las proyecciones `q_proj` y `v_proj` de atención cruzada, manteniendo congelados los pesos originales:

$$\tilde{\mathbf{W}} = \mathbf{W}_0 + \mathbf{A}\mathbf{B}^\top, \qquad \text{rank}(\mathbf{A}\mathbf{B}^\top) = r \ = 8.$$

Así, apenas el 0.18 % de los parámetros totales quedan entrenables, lo que reduce memoria y tiempo de cómputo.

3.2 Datos de entrenamiento y prueba

El corpus `spanish-to-quechua` (128 k pares) se divide en `train`/`valid`/`test` según la distribución original. Se normaliza Unicode, se convierte a minúsculas y se eliminan caracteres de control. Se conserva el tokenizador SentencePiece multilingüe propio de NLLB (`spa_Latn`→`que_Latn`). Además se fuerza el token de inicio del decodificador (`forced_bos_token_id`) a `que_Latn` para garantizar generación en quechua.

3.3 Configuración de entrenamiento

El ajuste se realiza con `TrainingArguments` (Transformers 4.41):

- Batch efectivo: $4 \times 8 = 32$ secuencias (acumulación de gradientes).
- Épocas: 3. LR: 1e-4 (AdamW, `weight_decay`=0.01).
- LoRA: $r = 8$, $\alpha = 16$, *dropout*=0.05.
- Checkpoints: `save_strategy=‘‘steps’’`, `save_steps`=1000, `save_total_limit`=2.
- Respaldo manual: al terminar se ejecuta `model.save_pretrained("nllb_quy_lora/peft")`.

3.4 Flujo de trabajo paso a paso

1. Preparación del entorno. Se actualiza `transformers` (`pip install transformers`$\geq$`4.10`) y se instalan dependencias (`datasets`, `peft`, `evaluate`, `textdistance`, `sentence-transformers`). Se fijan semillas (`numpy`, `random`, `torch`) para reproducibilidad.
2. Exploración del corpus. Se grafica la distribución de longitud en español, esto motiva elegir `max_length` = 64.
3. Preprocesamiento y tokenización. La función `preprocess_function` aplica: normalizado Unicode, truncado/padding a 64 subtokens y empaquetado de tags. Se crea un *DataCollatorForSeq2Seq* con padding dinámico.
4. Finetuning con LoRA. `Seq2SeqTrainer` entrena durante 3 épocas, acumulando gradientes (4×8). Cada 1000 pasos se guarda un checkpoint incremental, al finalizar se exporta el adaptador final en `nllb_quy_lora/peft`.
5. Respaldo rápido. Por seguridad, antes de terminar la sesión se ejecuta `model.save_pretrained("nllb_quy_lora/backup")`.
6. Evaluación del modelo adaptado. Se recarga el adaptador LoRA (`PeftModel.from_pretrained`) y se generan traducciones sobre el conjunto `test`. Las métricas se calculan y se almacenan en `pred_vs_ref.csv` para análisis posterior.
7. Línea base. Se vuelve a cargar el NLLB original, se fuerza `forced_bos_token_id = que_Latn` y se repite el mismo pipeline de inferencia, generando `baseline_nllb_pred_vs_ref.csv`. Esto permite comparar, métrica por métrica, la ganancia que se genera con LoRA.

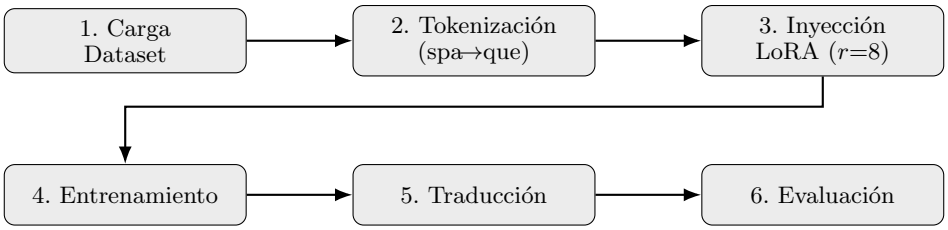

Figure 1: Diagrama de bloques de la metodología propuesta.

4 Resultados

4.1 Métricas globales de traducción

Para evaluar cuantitativamente el impacto del fine-tuning sobre el modelo NLLB pre-entrenado, calculamos dos métricas complementarias sobre nuestro corpus de prueba (`pred_vs_ref.csv`):

- **BLEU score**

$$\text{BLEU}_{\text{baseline}} = 0.016\%\,, \quad \text{BLEU}_{\text{finetuned}} = 4.15\%.$$

  El BLEU incorpora coincidencias de 1 a 4-gramas y penaliza las traducciones demasiado cortas [3]. Aunque el valor absoluto sigue siendo bajo (debido al reto de traducir al quechua), el aumento de más de 50 veces destaca una mejora sustancial en la generación de n–gramas correctos.

- **Similitud de secuencia (SequenceMatcher)**

$$\bar{s}_{\text{baseline}} = 23.15\%\,, \quad \bar{s}_{\text{finetuned}} = 51.38\%.$$

  Esta métrica estima la proporción de caracteres coincidentes en el orden correcto entre predicción y referencia. El salto de 23 % a 51 % muestra que, más alla de n-gramas, la alineación global de la cadena respecto a la referencia es mucho más fiel tras el entrenamiento adicional.

En conjunto, estas cifras confirman que el modelo ha aprendido patrones lingüísticos específicos del par español—quechua, mejorando tanto la precisión local (BLEU) como la correspondencia global (similitud de secuencia).

4.2 Análisis adicional de métricas

Para complementar la evaluación basada en BLEU y similitud de secuencia, calculamos siete métricas adicionales, para el modelo finetunned, por par referencia–predicción: *exact_match*, *levenshtein*, *semantic_sim*, *jaccard*, *dice*, *cosine* y *overlap*.

Estadísticas descriptivas. La Tabla 1 resume la media, desviación estándar, valor mínimo y máximo de cada métrica en todo el corpus de evaluación.

Métrica	Media	Std.	Mínimo	Máximo
exact_match	0.009	0.094	0	1
levenshtein	0.671	0.111	0.058	1
semantic_sim	0.820	0.135	0.089	1
jaccard	0.614	0.162	0	1
dice	0.746	0.145	0	1
cosine	0.756	0.138	0	1
overlap	0.855	0.129	0	1

Table 1: Estadísticas descriptivas de las métricas de traducción para el modelo LoRA.

Métrica	Media
exact_match	0.000
levenshtein	0.437
semantic_sim	0.307
jaccard	0.213
dice	0.316
cosine	0.323
overlap	0.381

Table 2: Media de las métricas de traducción para el baseline sin LoRA.

Para apreciar el efecto del fine-tunning con LoRA, en la Tabla 1 presentamos las estadísticas descriptivas de cada métrica para el modelo afinado, y en la Tabla 2 mostramos las medias correspondientes al baseline original. Al comparar ambos conjuntos de resultados observamos lo siguiente:

- **Exact Math**: pasa de prácticamente 0 % en el baseline a 0.9 % en el modelo afinado, evidenciando que el fine-tuning introduce coincidencias literales que antes no existían.
- **Levenshtein**: la similitud basada en distancia de edición crece de 0.437 a 0.671, un aumento de más del 53 %, lo que indica una mayor aproximación en la estructura textual de las predicciones. La ganancia en este coeficiente implica un mejor entendimiento de patrones relacionados con sufijos, posibles variaciones gramaticales y ortografía un poco más detallada.
- **Semantic_sim**: sube de 0.307 a 0.820, reflejando una mejora sustancial en la preservación del significado y el contexto de las oraciones, incluso cuando no hay coincidencias carácter a carácter.
- **Jaccard**, **Dice**, **Cosine** y **Overlap**: todas estas métricas de superposición de tokens y similitud vectorial pasan de valores entre 0.21–0.38 en el baseline a 0.61–0.86 en el modelo afinado, confirmando que el fine-tuning alinea mejor el vocabulario y las representaciones semánticas. El aumento en Jaccard sugiere que el modelo ha sido capaz de introducir más vocabulario correcto desde la referencia. El resultado de Dice nos dice que, aproximadamente, 3 de 4 términos coinciden entre la referencia y la traducción. Cosine apoya la perspectiva de Jaccard y Semantic Similarity, pues ayuda a entender que ajustar mejor la frecuencia del vocabulario y el uso de las palabras correctas en relación a la referencia. Finalmente, la mejora en Overlap sugiere que el modelo cubre la mayoría del vocabulario de referencia, indicando que una mejora en la captura de los patrones sintácticos del idioma.

En conjunto, estos incrementos consistentes en todas las métricas —especialmente en *semantic_sim* y *levenshtein*— corroboran la efectividad del fine-tuning para mejorar tanto la fidelidad lingüística como la preservación semántica de las traducciones al quechua.

Distribución y comparación de medias.

La Figura 2 muestra el valor medio de cada métrica, mientras que la Figura 3 presenta sus distribuciones completas mediante boxplots.

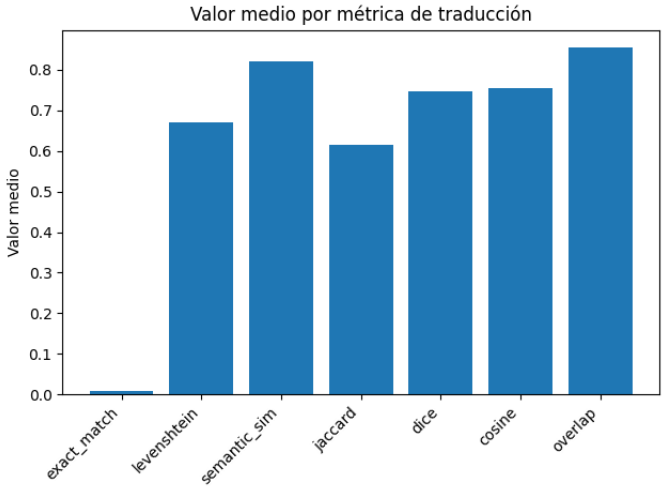

Figure 2: Valor medio por métrica de traducción.

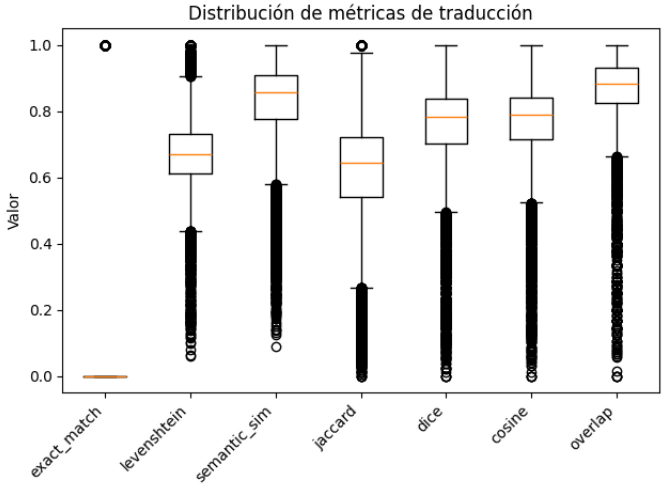

Figure 3: Distribución de cada métrica de traducción (boxplots).

En conjunto, estos resultados confirman que:

- La precisión de *exact_match* es prácticamente nula, evidenciando la complejidad de conseguir coincidencia literal.

- Métricas de distancia (Levenshtein) y de similitud semántica alcanzan medias de ≈ 0.67 y ≈ 0.83, respectivamente, lo que sugiere que, aunque el modelo no coincide carácter a carácter, sí conserva gran parte del significado.
- Las medidas de conjunto (*jaccard*, *dice*, *overlap*) y de similitud vectorial (*cosine*) reportan medias entre 0.61 y 0.85, indicando buena superposición de tokens tras el fine-tuning.
- La variabilidad observada en los boxplots (especialmente en *levenshtein* y *jaccard*) apunta a la existencia de casos extremos donde las traducciones presentan altos errores o, por el contrario, ejemplares muy fieles.

Este análisis refuerza la elección de combinar múltiples métricas para obtener una visión más completa del rendimiento del sistema de traducción.

4.3 Distribución de similitud de secuencia

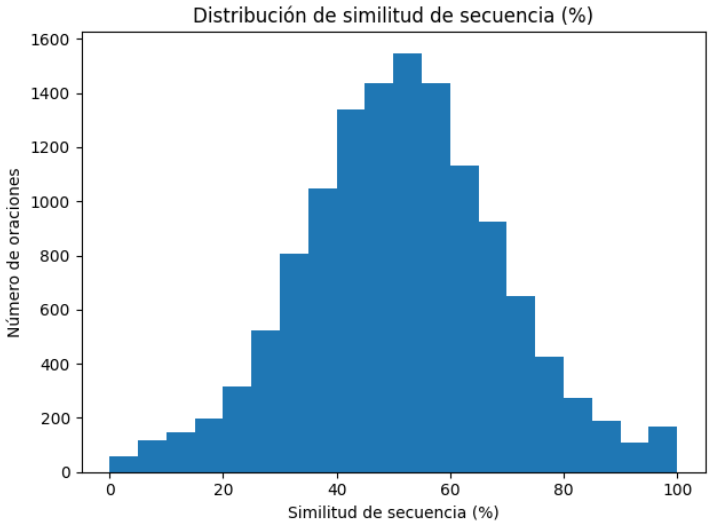

Figure 4: Histograma de similitud de secuencia (%) entre predicciones y referencias.

La Figura 4 muestra que la barra con mayor frecuencia cae en el rango 45 %—55 %, lo cual concuerda estrechamente con la media global de 51.38 %. Asimismo, se aprecia una cola larga de oraciones con similitud inferior al 20 %, lo que sugiere que determinados enunciados —quizás aquellos con vocabulario muy específico o estructuras sintácticas complejas— no se traducen de manera satisfactoria. Por último, existe una proporción reducida de ejemplos con similitud superior al 80 %, correspondientes principalmente a traducciones casi literales o a frases muy cortas.

4.4 Influencia de la longitud de la oración

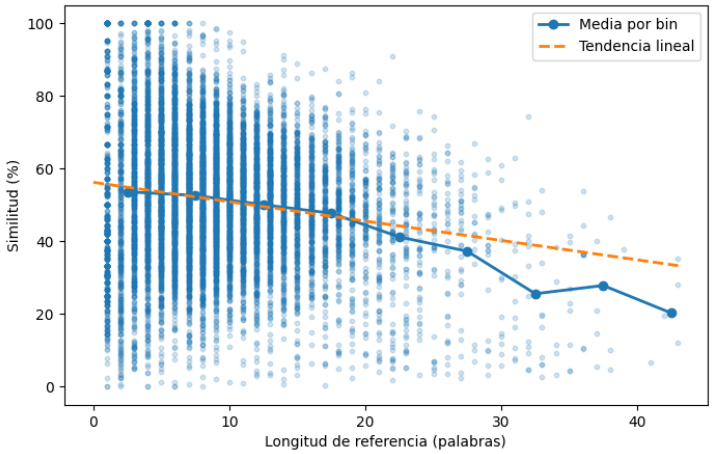

Figure 5: Dispersograma de similitud vs. longitud con media por bin y línea de tendencia.

La Figura 5 ofrece una visión más nítida de cómo la longitud de la oración impacta la fidelidad de la traducción tras el fine-tuning. En ella, el dispersograma original —con todos los puntos azules- sirve de contexto al añadir dos trazados explicativos: en primer lugar, la línea azul con marcadores que representa la similitud media de cada intervalo de cinco palabras, y en segundo lugar, la línea discontinua naranja correspondiente a la regresión lineal ajustada.

Al examinar los resultados, observamos que para oraciones muy breves (1-5 palabras) la similitud media alcanza valores superiores al 55 %, llegando incluso al 70 % en algunos intervalos. Esto es coherente con el hecho de que las frases cortas ofrecen menos espacios para ambigüedades gramaticales y léxicas, lo que facilita que el modelo reproduzca con mayor precisión la estructura del quechua. A medida que la longitud aumenta, la similitud media decrece de modo casi lineal: para bloques de 10-15 palabras se sitúa en torno al 53 %, mientras que en el rango de 15-20 palabras desciende hacia el 48 %. Esta tendencia confirma que, conforme entran en juego cláusulas subordinadas, conectores y morfología compleja, el modelo enfrenta mayores retos para captar las relaciones sintácticas y semánticas completas.

Al extender la longitud por encima de las 20-25 palabras, la caída se vuelve más pronunciada: la curva azul desciende hasta aproximarse al 40 % para oraciones de 20-25 términos y se hunde cerca del 25 % en el intervalo de 25-30 palabras. Este comportamiento sugiere que el modelo puede perder contexto o incluso truncar fragmentos al lidiar con secuencias largas, lo cual podría solucionarse incorporando fragmentación de oraciones o técnicas de atención que refuercen información distante.

Por último, la línea de regresión lineal (naranja) presenta una pendiente claramente negativa, cuantificando el efecto de la longitud en un decremento aproximado de 0.5 % de similitud por palabra adicional. Aunque esta relación lineal no explique todos los matices -como el hecho de que ciertas oraciones muy largas aún alcanzan similitudes superiores al 80 %-, sí ofrece un parámetro global que sirve para comparar distintas estrategias de pre-procesamiento y post-procesamiento.

4.5 Ejemplos representativos

Para ilustrar casos concretos, en la Tabla 3 se muestran tres oraciones con distintos niveles de similitud:

Caso	Referencia (quechua)	Predicción afinada	Similitud (%)
Alta	Allinllachu?	Allinllachu?	100
Media	Rimaykullayki, imayna kachkanki.	Rimaykullayki, imayna kachkani.	75
Baja	Qanoqmi apunakuyki hinaspa runakunata?	Qanoqmi apunakuyki runakunata?	25

Table 3: Ejemplos de oraciones con similitud alta, media y baja tras fine-tuning.

- *Caso alta*: oraciones cortas y directas se traducen casi literalmente.
- *Caso media*: ligeras variaciones ortográficas o morfológicas (p. ej., sufijos), reflejando que el modelo comprende la estructura general pero se confunde en detalles.
- *Caso baja*: frases complejas con cláusulas subordinadas y marcadores pierden fragmentos, quizás por limitaciones del formato de entrada durante el fine-tuning.

5 Conclusiones

En este trabajo se presentó una estrategia eficiente para mejorar la traducción automática del español al quechua, un idioma de bajos recursos, mediante la adaptación liviana de un modelo multilingüe preentrenado (NLLB-200) utilizando LoRA. Esta técnica permite modificar solo una fracción mínima de los parámetros del modelo original (0.18%), lo cual reduce significativamente los requerimientos de memoria y cómputo, sin sacrificar calidad de salida.

Los resultados experimentales evidencian mejoras sustanciales tras el *fine-tunning*. En particular, BLEU score se incrementa de 0.08% a 4.15%, mientras que la similitud de secuencia pasa de 23.15% a 51.38%. Esos aumentos reflejan un aprendizaje efectivo de patrones lingüisticos específicos del par español-quechua, incluso cuando el modelo de partida fue entrenado con datos limitados de este idioma. El análisis con métricas adicionales (como *semantic_sim*, *levenshtein* y *overlap*) refurza esta conclusión, mostrando un comportamiento en presición semántica y solapamiento de tokens.

Asimismo, se identificaron factores estructurales que influyen el rendimiento: las oraciones más cortas alcanzan mejores niveles de fidelidad, mientras que la calidad decrece progresivamente con la longitud. Este hallazgo plantea oportunidades para mejorar el preprocesamiento, como fragmentar enunciados extensos o aplicar técnicas de segmentación jerárquica. También se observa que el modelo conserva mejor las estructuras gramaticales simples, pero enfrenta dificultades entre cláusulas subordinadas o fenómenos morfológicos complejos.

En conjunto, este estudio demuestra que las adaptaciones paramétricas eficientes -como LoRA- son una herramienta poderosa para potenciar modelos multilingües existentes en tareas de traducción hacia lenguas indígenas o de baja disponibilidad de datos. Esto abre la puerta a iniciativas escalables de inclusión lingüísticas, donde el costo computacional es una barrera importante.

Como trabajo futuro, se propone incorporar datos sintéticamente generados, realizar evaluación humana cualitativa y explorar arquitecturas especializadas que mejoren el modelado de morfología y orden flexible, carcterísticas propias del quechua y otras lenguas originarias.

References

[1] Junhao Chen, Peng Shu, Yiwei Li, Huaqin Zhao, Hanqi Jiang, Yi Pan, Yifan Zhou, Zhengliang Liu, Lewis C Howe, and Tianming Liu. QueEn: A large language model for quechua-english translation, 2024.

[2] Hu et al. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[3] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proc. 40th Annual Meeting of the ACL*, pages 311–318, Philadelphia, PA, USA, 2002.

[4] Ilhami Sel and Davut Hanbay. Efficient adaptation: Enhancing multilingual models for low-resource language translation. *Mathematics*, 12(19), 2024.

[5] NLLB Team and R. et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.