# Capítulo 5. Pipeline de preprocesamiento, aumentación y vectorización

En este capítulo se describen las etapas de preprocesamiento, aumentación y vectorización por el que los datos del corpus atravesarán. Se espera que estas operaciones consigan transformar a los datos a un formato que pueda ser mejor aprovechado por los modelos a entrenar. En la Figura 6. Resumen del pipeline de preprocesamiento, aumentación y vectorizaciónse puede apreciar un resumen gráfico del este flujo de transformaciones. En las siguientes secciones, se describe cada una de las etapas, las operaciones realizadas y las razones que llevaron a la elección de determinada decisión. Luego, se prueba la eficacia de este pipeline mediante una tarea de clusterización simple. Se ofrece finalmente una discusión en donde se evalúa el impacto en el proyecto de lo conseguido.

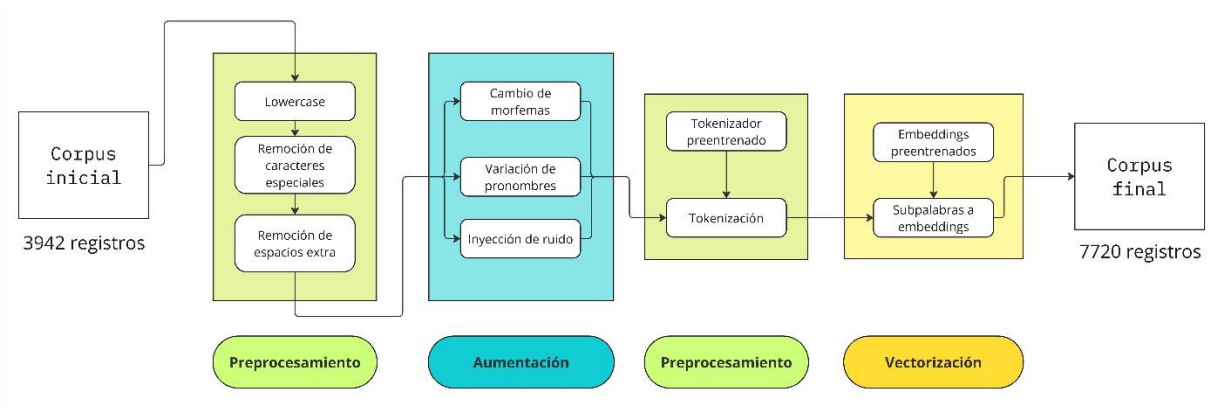



*Figura 6.* Resumen del pipeline de preprocesamiento, aumentación y vectorización

## 5.1. Resultados alcanzados

### 5.1.1. Preprocesamiento

Para el tratamiento previo de los datos, el primer paso fue aplicar las siguientes operaciones básicas a la columna *transcription*:

- Convertir todos los caracteres alfabéticos a minúsculas
- Eliminar caracteres especiales (no alfabéticos ni numéricos)
- Eliminar espacios en blanco extra entre palabras y al inicio y al final de la oración

#### 5.1.1.1. Elección del tokenizador

A continuación, se decidió tokenizar las oraciones. Es decir, separar las oraciones en partes más pequeñas bajo cierto criterio. Existen distintos enfoques para este proceso: separación a nivel de palabra, a nivel de carácter, a nivel de morfema, a nivel de sílaba o a nivel de otras subpalabras. Como se describió en el marco conceptual, el Iskonawa es una lengua aglutinante.

Es decir, se forman palabras mediante el añadido de diversos morfemas. En ese sentido, se procedió a identificar, con ayuda de la revisión sistemática, qué método sería el más apropiado para una lengua de esas características.

Un estudio realizó pruebas con otra lengua aglutinante, el turco, para identificar qué método de tokenización traía más beneficios (Toraman et al., 2023). Sus resultados mostraron que tokenizadores a nivel de carácter o palabra tenían un performance deficiente en una serie de tareas específicas. En contraste, los métodos considerados como estado del arte (Byte-pair encoding y WordPiece) fueron los que mejor se desenvolvieron. Junto a ellos, el tokenizador a nivel morfológico tuvo un desempeño similar.

En estudios más cercanos, Zevallos (2022) utiliza también un tokenizador Byte-pair encoding (BPE), pues la considera el estándar de facto para lenguas aglutinantes. Asimismo, realizó experimentos con una variación de dicho BPE y otra técnica que intenta separar a las palabras en prefijos, raíces y sufijos (Prefix-Root-Postfix-Encoding).

En ese sentido, se decide que los tokenizadores a utilizar deben operar a nivel de subpalabra, pues permite capturar de mejor manera la información de la lengua dada su naturaleza aglutinante. Además, considerando que el Iskonawa usa exclusivamente sufijos y el desarrollo de un segmentador morfológico escapa al alcance de este proyecto, se decide no trabajar con el método PRPE ni tokenizadores morfológicos. De esta forma, se optó por tokenizar las oraciones en Iskonawa con ayuda de un tokenizador BPE entrenado en nuestro corpus.

#### 5.1.1.2. Entrenamiento del tokenizador

Para entrenar el tokenizador seleccionado, se optó por utilizar la implementación de BPE en SentencePiece (Kudo y Richardson, 2018). Esta herramienta, más allá de ser fácilmente accesible mediante un módulo de Python, permite entrenar un tokenizador de manera no supervisada directamente con el texto en crudo; es decir, sin una "pre-tokenización". Además, implementa distintos algoritmos como BPE, Unigrama, a nivel de carácter y a nivel de palabra.

Una vez instalado el módulo, se guardó el contenido de la columna *transcription* (luego de la limpieza inicial) en un archivo *txt* fila por fila, pues es el formato que la herramienta esperaba. Luego de crear este archivo, se entrenó al tokenizador SentencePiece con un tamaño de vocabulario de 2048 y con el mencionado algoritmo BPE. Esta herramienta guarda lo aprendido por el tokenizador en dos archivos: un .model y un .vocab. El primero es de tipo binario, mientras que en el segundo se pueden visualizar las subpalabras aprendidas. En la

Figura 7 se aprecian dichas subpalabras con el código asignado a cada una por el tokenizador. Ambos archivos fueron almacenados en el repositorio remoto.

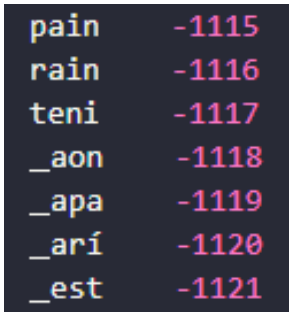



*Figura 7.* Extracto del archivo .vocab del tokenizador en Iskonawa

### 5.1.2. Aumentación

Para la etapa de aumentación, se aplicaron fundamentalmente tres operaciones:

- **Cambio de morfemas de determinada categoría gramatical:** Una considerable cantidad de oraciones en el corpus estaban acompañadas de anotaciones respecto a su categoría (por ejemplo, verbo, sustantivo, adjetivo, entre otras). Se decidió aprovechar este conocimiento para aumentar el número de oraciones. Siguiendo una de las técnicas propuestas en el conjunto de métodos Easy Data Augmentation (Wei y Zou, 2019), se realizó una operación parecida al cambio de sinónimos.

  Dado que el Iskonawa es aglutinante, cambiar o reemplazar una palabra entera significaría perder información o cambiar totalmente el significado de la oración. Por ello, este reemplazo fue realizado a nivel de morfema. Este cambio fue realizado a los morfemas con las siguientes categorías: adjetivos, cuantificadores, números, demostrativos, determinantes, ideófonos, interrogativos, interjecciones, sustantivos, onomatopeyas, posposiciones y verbos.

  Entonces, si un morfema era de la categoría adjetivo, se reemplazaba por otro aleatorio de esta misma categoría. Previamente se procesó todo el corpus para tener un diccionario con las categorías presentes en el corpus y los morfemas encontrados para cada una de ellas. Esta operación fue aplicada con un 50% de probabilidad a cualquier morfema que cayese en una de las categorías anteriormente descritas. En la Tabla *12* se muestran algunos ejemplos de cómo se realizó esta alteración.

Tabla 13

*Ejemplos de aumentación mediante reemplazo de morfema*

| Categoría | Oración original | Oración nueva |
|---|---|---|
| Adjetivo | kapa chopa **biri**bi | kapa chopa **niborewan**bi |
| Sustantivo | nena non **epa**bo yosibo iketian | nena non **mapo**bo yosibo iketian |
| Verbo | **oin**yamabobi | **rowa**yamabobi |

- **Variación de pronombres:** Esta operación fue similar a la anterior; sin embargo, se restringió a solo los pronombres, pues existían características adicionales que considerar. De esta forma, por cada registro, se identificaron los morfemas que hayan sido etiquetados como pronombres. Luego, con un 30% de probabilidad se reemplazó este morfema por uno con distinta persona y número, pero del mismo rol sintáctico. Esto se puede entender de mejor manera consultando la Tabla 14. Si se encontró un pronombre etiquetado como 1PL:S, este se podría reemplazar por cualquiera de los pronombres de la columna S en la tabla para crear la nueva oración.

Tabla 14

*Pronombres personales del Iskonawa. Tomado de (Zariquiey, 2015)*

| Persona | **A** (sujeto de verbo transitivo) | **S** (sujeto de verbo intransitivo) | **P** (objeto de verbo transitivo) |
|---|---|---|---|
| **Primera singular** | ena | eah | ea |
| **Segunda singular** | mi | mi | mia |
| **Tercera singular** | oanton | oa | oa |
| **Primera plural** | | | |
| **Inclusivo** | non(bo) | no(bo) | no(bo) |
| **Exclusivo** | enabo | eahbo | eabo |
| **Segunda plural** | mibo | mibo | miabo |
| **Tercera plural** | aboton | abo | abo |

- **Inyección de ruido:** Dentro del set de técnicas de aumentación para tareas generativas denominado GenAug (Feng et al., 2020), se encuentra el uso de la inserción de ruido sintético. En el artículo se proponen distintos métodos de aumentación (como el reemplazo de hipónimos, hiperónimos, sinónimos, entre otras); sin embargo, concluyen que uno de los métodos que mejor resultado obtiene es la inserción de ruido a nivel de carácter.

  Por ello, se implementó también este mecanismo en la etapa de aumentación. Por cada oración, se recorrieron cada una de sus palabras y, con un 10% de probabilidad, se aplicó alguna de estas 3 formas de ruido a dicha palabra: intercambio entre dos caracteres vecinos, inserción de un carácter aleatorio o eliminación de un carácter aleatorio.

Además, también se intentó aumentar los datos mediante el manipulado de árboles de dependencia, como se planteó en (Şahin y Steedman, 2018). Sin embargo, está operación requería tener anotaciones respecto a las dependencias dentro de la oración; lamentablemente, no se contaba con dichos datos en ninguna de las oraciones. Luego de aplicar los tres métodos descritos anteriormente, se obtuvieron em promedio 7720 oraciones (3932 más que en el corpus original).

### 5.1.3. Vectorización

Para la etapa de vectorización, se consideraron las siguientes alternativas encontradas en la literatura para el entrenamiento de embeddings en un contexto de lenguas de bajos recursos:

- **Anchored bi- and multilingual word embeddings (Eder et al., 2021):** Este artículo propone la creación de embeddings bilingües para lenguas de bajos recursos mediante el uso del espacio vectorial de los embeddings de una lengua con muchos recursos. Es decir, aprovecha el conocimiento adquirido durante el entrenamiento de embeddings con grandes cantidades de datos en cierta lengua y transfiere ese aprendizaje a una lengua de más bajos recursos mediante el uso de sus vectores como puntos de anclaje.

  Para este proyecto, se utilizó el código publicado[5] por los autores para replicar los experimentos con nuestro set de datos. Además, se utilizaron embeddings[6] en español

[5] https://github.com/hangyav/anchor-embeddings
[6] https://github.com/dccuchile/spanish-word-embeddings

entrenados con el método Word2Vec en el corpus Spanish Billion Word Corpus. De esta forma, se utilizó dicho espacio vectorial como anclaje para obtener embeddings en Iskonawa.

- **Contextualised cross-lingual word embeddings (Wada et al., 2020):**
  Esta técnica busca obtener embeddings interlingüísticos a partir de pequeños corpus paralelos mediante el entrenamiento de un modelo encoder-decoder LSTM. Además, explora una opción para utilizar esta técnica en conjunto con otros embeddings pre-entrenados.
  En el contexto del proyecto, se utilizó también el código[7] publicado por los autores para replicar esta técnica con nuestros datos. Se obtuvieron dos archivos con embeddings en Iskonawa: el primero fue conseguido entrenando el modelo solo con la data paralela de nuestro corpus; el segundo, se obtuvo de entrenar también con esta data paralela, pero a partir de los embeddings conseguidos por el método anterior de anclaje.

- **FastText:** Se entrenaron también embeddings mediante este método originalmente propuesto en (Bojanowski et al., 2017) que busca enriquecer a los embeddings mediante el uso de subpalabras. Se tomó el algoritmo base sin ninguna modificación con el fin de que funcionara como línea base para comparar las otras técnicas antes descritas.

Para todos los casos antes descritos, los embeddings fueron entrenados durante 150 épocas. Además, el tamaño de los vectores fue de 300 y en todos los casos se utilizó SkipGram debido a su mejor desempeño en corpus pequeños (Mikolov et al., 2013).

### 5.1.4. Prueba en tarea de clusterización

Como se definió en los objetivos, una vez definido y desarrollado el pipeline, se sometió el corpus de datos en Iskonawa a esta serie de operaciones para ser usados en una tarea de clusterización con el fin de determinar su utilidad. Un cluster es esencialmente un grupo de

[7] https://github.com/twadada/multilingual-nlm

elementos; ambas palabras serán usadas intercambiablemente en adelante. En ese sentido, “clusterizar” se refiere a agrupar elementos en conjuntos con algunas características en común.

La tarea consistió en extraer 3000 registros aleatorios del corpus luego de atravesar por las etapas de preprocesamiento y aumentación, pero antes de la etapa de vectorización. Luego, se vectorizaron estos 3000 registros mediante los cuatro distintos embeddings mencionados en el capítulo anterior.

Los objetos a clusterizar son los embeddings de las palabras encontradas en estos 5000 registros. Dado que los embeddings fueron entrenados a nivel de subpalabras, el vector de cada palabra se calculó promediando los vectores de todas las subpalabras que la componen. Una vez obtenidas estas representaciones, se entrenó un modelo de clusterización KNN para cada uno de los 4 grupos. Los resultados se pueden visualizar en la Figura 8 , donde, tras aplicar una reducción de dimensionalidad, los vectores se redujeron a solo dos componentes con el fin de identificar visualmente cómo se agruparon las palabras.

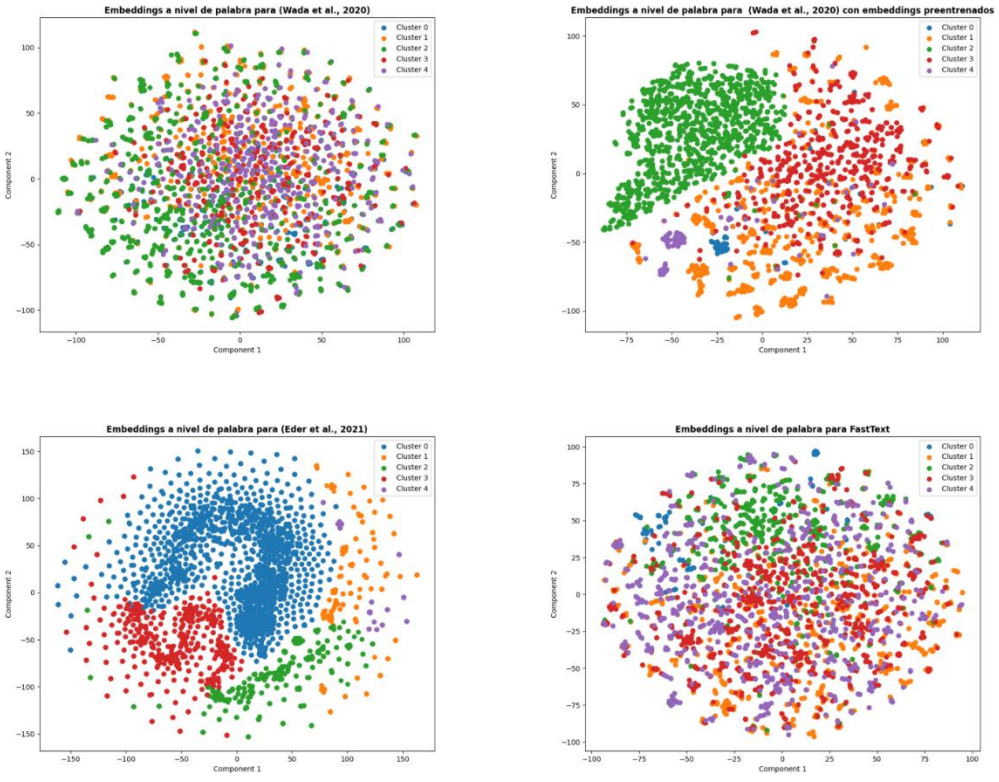



*Figura 8.* Visualización en dos dimensiones de la clusterización a nivel de palabra

Además, se realizó el mismo procedimiento, pero a nivel de oración. Una vez más, cada oración fue tokenizada y vectorizada usando los 4 grupos de embeddings. Los resultados pueden visualizarse en la Figura 9.

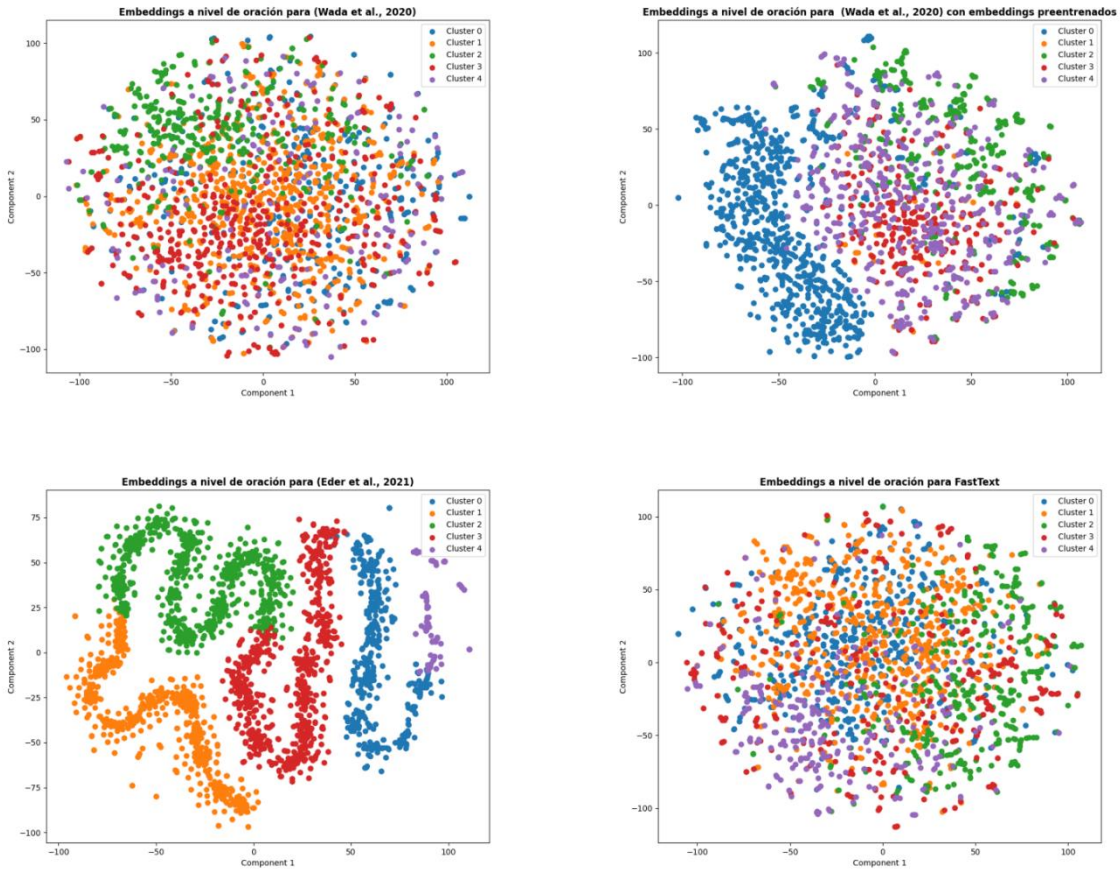



*Figura 9.* Visualización en dos dimensiones de la clusterización a nivel de oración

Además de analizar visualmente el desempeño de esta tarea, se calcularon una serie de métricas para comparar qué tan adecuados fueron los grupos formados durante el agrupamiento. Estas fueron elegidas considerando que no se contaba con una etiqueta de las oraciones o palabras para realizar una comparación con lo determinado por la clusterización. Las métricas calculadas fueron:

- **Silhoute Score:** Métrica que indica que tan cerca cada punto en un cluster está de clusters vecinos. Sus valores van entre -1 a 1. Mientras más cerca a 1 esté el valor, los clusters serán más densos y separados entre sí, lo que indica un buen agrupamiento (Rousseeuw, 1987).

- **Índice de Davies-Bouldin :** Métrica que calcula qué tan separados están los clusters. Es un número positivo y, mientras más cercano a 0 sea, demuestra que los grupos están mejor particionados (Davies y Bouldin, 1979).
- **Índice de Calinski-Harabasz:** Este índice calcula la razón entre la suma de dispersiones entre clusters y la suma de dispersiones "intra-cluster". Es un valor positivo sin un límite definido y, mientras más alto sea, indica un cluster denso y bien separado del resto (Caliński y JA, 1974).

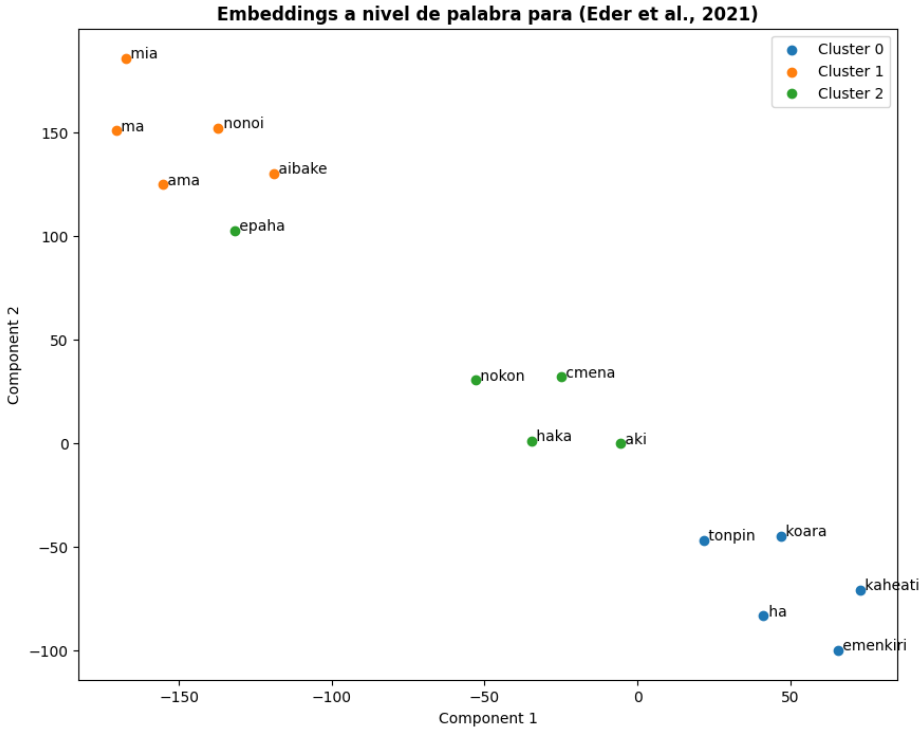
Los resultados de estas métricas para los dos escenarios de clusterización pueden consultarse en la Tabla 15.

Tabla 15

*Resultados de las métricas de clusterización*

| **Embeddings\Métrica** | Silhouette Score (Palabra) | Índice de Davies-Bouldin (Palabra) | Índice de Calinski-Harabasz (Palabra) | Silhouette Score (Oración) | Índice de Davies-Bouldin (Oración) | Índice de Calinski-Harabasz (Oración) |
|---|---|---|---|---|---|---|
| **(Wada et al., 2020)** | 0,24 | 1,85 | 315,09 | 0,0044 | 1,8748 | 110,64 |
| **(Wada et al., 2020) con embeddings preentrenados** | 0,33 | 0,50 | 29,88 | 0,035 | 1,4386 | 597,02 |
| **(Eder et al., 2021)** | **0,72** | **0,47** | **38757,31** | **0,6591** | **0,4839** | **148984,74** |
| **FastText** | -0,02 | 1,21 | 52,19 | -0,0005 | 5,9831 | 332,81 |

En las tres métricas elegidas, le técnica de embeddings bilingües anclados (Eder et al., 2021) tuvo el mejor performance tanto a nivel de palabra como a nivel de oración.




### 5.2. Discusión

En esta sección se presentó el flujo de transformaciones por el que el corpus de datos atravesará para ser alimentada al modelo. Se aplicaron una serie de operaciones de preprocesamiento para limpiar los datos de caracteres extraños y remover el uso de mayúsculas y espacios en blanco adicionales. Se aumentaron también los datos mediante alteración de partes de las palabras (al aprovechar la información morfológica de las anotaciones lingüísticas) y la inserción de ruido aleatorio. Asimismo, se separaron las palabras en subunidades mediante un algoritmo BPE para finalmente ser traducidas a cuatro espacios vectoriales.

Dichas operaciones iniciales de preprocesamiento fueron necesarias para evitar que los datos contengan elementos que puedan entorpecer el proyecto. Por ejemplo, si se hubiesen dejado a kapa y Kapa (ardilla) como dos elementos distintos, el tokenizador los habría codificado de forma distinta. Esto a su vez llevaría dos representaciones distintas en el espacio vectorial que podrían perjudicar el aprendizaje del modelo. Por otro lado, la elección de un tokenizador a nivel de subpalabra permitió trabajar de una mejor forma con la naturaleza aglutinante del Iskonawa. Se sugiere como trabajos futuros explorar el impacto de un tokenizador morfológico para la separación en subpalabras.

La aumentación a su vez impacta favorablemente en el futuro del desarrollo del proyecto, pues aproximadamente se duplicó la cantidad de registros con los que se contaba. Este incremento

fue realizado sin alterar excesivamente el significado y estructura de los registros como se describió en la sección. El uso de la gramática Iskonawa (Zariquiey, 2015) fue vital para decidir qué morfemas mutar y bajo qué condiciones hacerlo. En ese sentido, se recomiendo que futuras tareas de aumentación en otras lenguas de bajos recursos hagan uso de este conocimiento lingüístico; ya sea mediante el uso de una gramática en dicha lengua o el uso de anotaciones morfológicas. Adicionalmente, se evidenció que contar con anotaciones lingüísticas respecto a las dependencias en los registros hubiera permitido utilizar otras técnicas de aumentación, como las relacionadas a los árboles de dependencia.

Finalmente, la tarea de clusterización permitió tener una idea de qué métodos de vectorización fueron los más adecuados. Si bien las métricas indicaron que los embeddings conseguidos a partir del método de embeddings bilingües anclados fueron los que permitieron un mejor agrupamiento, la utilidad real de este espacio vectorial será verificada en las siguientes etapas del proyecto.