UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE POSTGRADO Y EDUCACIÓN CONTINUA

IMPLEMENTACIÓN DE MODELOS DE LENGUAJE PRE ENTRENADOS PARA LA MITIGACIÓN DEL SESGO DE GÉNERO EN COMUNICACIONES ESCRITAS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIA DE DATOS

GIANINA ALESSANDRA SALOMÓ LÓPEZ

PROFESOR GUÍA:
Felipe Tobar Henríquez

MIEMBROS DE LA COMISIÓN:
Darinka Radovic Sendra
Felipe Bravo Márquez

Este trabajo ha sido parcialmente financiado por:
Google y FONDECYT Regular 1210606

SANTIAGO DE CHILE
2024

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIA
DE DATOS
POR: GIANINA ALESSANDRA SALOMÓ LÓPEZ
FECHA: 2024
PROF. GUÍA: FELIPE TOBAR HENRÍQUEZ

IMPLEMENTACIÓN DE MODELOS DE LENGUAJE PRE ENTRENADOS PARA LA MITIGACIÓN DEL SESGO DE GÉNERO EN COMUNICACIONES ESCRITAS

Los modelos de aprendizaje automático para tareas de procesamiento del lenguaje natural (NLP) suelen entrenarse con texto que puede presentar sesgo en cuanto al género, produciendo que dichos modelos puedan generar también resultados considerando dicho sesgo. Para afrontar parte de este problema, se propone una metodología para mitigar el sesgo de género en textos en español de comunicaciones escritas dentro de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile (FCFM) mediante transferencia de estilo de texto (TST), utilizando datos paralelos y un modelo de lenguaje pre entrenado (PLM) ajustado sobre dichos datos. El modelo ajustado es capaz de generar salidas con mitigación de sesgo de género, a partir de entradas que pueden o no tener sesgo de género. Se explora el ajuste de dos PLMs, Modelo de Lenguaje Causal (CausalLM) y Modelo de Lenguaje de Secuencia a Secuencia (Seq2SeqLM), utilizando un ajuste fino eficiente en parámetros (PEFT) con Adaptación de Bajo Rango (LoRA) en forma cuantizada (QLoRA). Los resultados obtenidos son prometedores, logrando una mitigación del sesgo de género frente a diversas entradas, lo cual es evaluado comparando el puntaje BLEU obtenido entre entradas sesgadas y sus salidas esperadas insesgadas, y el puntaje BLEU obtenido entre las salidas esperadas y las generadas por el modelo ajustado. Además, se entrega como contribución una Interfaz de Programación de Aplicaciones (API) con una demostración que muestra sugerencias de mitigación de sesgo en tiempo real, ofreciendo una herramienta práctica para abordar el sesgo de género en diversos contextos.

Agradecimientos

En primer lugar quiero agradecer a Google y FONDECYT por facilitar financiamiento para el desarrollo de esta tesis. Quiero agradecer también a mi profesor guía, Felipe Tobar, quien ha sido un gran apoyo tanto en conocimiento como ideas a lo largo de este procesos. Finalmente, quiero agradecer también a mi pareja Victor Gaete por ser un apoyo emocional durante los dos años de estudio del magíster.

Tabla de Contenido

1. Introducción **1**

1.1. Motivación . . . 1

1.2. Antecedentes . . . 2

1.2.1. Transformers . . . 2

1.3. Modelos de lenguaje pre entrenados . . . 5

1.4. Objetivos . . . 6

1.4.1. Objetivo general . . . 6

1.4.2. Objetivos específicos . . . 6

1.5. Contribuciones . . . 7

2. Metodología **8**

2.1. Construcción del corpus . . . 10

2.1.1. Recolección de datos . . . 10

2.1.2. Construcción de datos paralelos . . . 10

2.2. Modelos utilizados . . . 11

2.2.1. Ajuste fino de parámetros en forma eficiente . . . 14

2.3. Preparación de datos . . . 15

2.3.1. Tokenización . . . 15

2.3.2. Añadir Prompt . . . 16

2.3.3. Largo máximo de secuencia . . . 16

2.4. Evaluación de modelos ajustados . . . 17

2.5. Aplicación Web . . . 20

3. Experimentos **21**

3.1. Congiruación general de hiperparámetros . . . 21

3.2. Seq2SeqLM . . . 22

3.3. CausalLM . . . 23

4. Resultados **25**

4.1. Exploración de clasificación de mensajes utilizando GPT 3.5 . . . 25

4.2. Inferencia de modelos ajustados sobre C_{test} . . . 26

4.2.1. Seq2SeqLM . . . 26

4.2.2. CausalLM . . . 27

4.2.3. Métricas de desempeño para modelos ajustados 28

5. Conclusiones **33**

Bibliografía **35**

Anexos **38**

A. Análisis exploratorio de Corpus construido 38

A.1. Frases posibles de ser pareadas 38

A.2. Cantidad de mensajes y frases por año 39

A.3. Presencia de sesgo de género 39

A.4. Palabras más frecuentes 41

Índice de Tablas

2.1. Estadística descriptiva de cantidad de Tokens por documento de entrenamiento 17
2.2. Situaciones de comparación BLEU 19
4.3. Métricas de desempeño en modelos ajustados 28
4.4. Resultados de comparación BLEU utilizando modelo Falcon ajustado 29
4.1. Ejemplos de clasificaciones de textos entre con y sin sesgo de género realizadas por GPT 3.5 turbo 31
4.2. Ejemplos de fragmentos de generaciones de modelos ajustados 32

Índice de Ilustraciones

1.1. Arquitectura transformer 4
1.2. Atención de producto punto escalado y Multi-Head Attention 4
1.3. Línea de tiempo de LLM 5
2.1. Elementos de la tarea a resolver 8
2.2. Elementos del entrenamiento 9
2.3. Arquitectura de PLMs escogidos (elaboración propia) 14
2.4. Uso del modelo ajustado 20
3.1. Pérdidas en C_{train} y C_{val} durante ajuste de modelos. 24
3.2. Decaimiento de tasa de aprendizaje durante ajuste de modelos. 24
4.1. Ejemplo de uso aplicación *demo* con modelo FLAN-T5 ajustado. 27
4.2. Ejemplo de uso aplicación *demo* con modelo Falcon ajustado. 28
4.3. Porcentaje de generaciones correctas e incorrectas en C_{test} 30
A.1. Exploración de frecuencias de frases posibles y no posibles de ser pareadas 38
A.2. Mensajes y frases obtenidas por año 39
A.3. Exploración de frecuencias de frases con y sin sesgo de género 40
A.4. Frecuencias de frases con sesgo de género por año 40
A.5. Palabras más frecuentes en frases posibles de parear utilizadas para construir C. . . . 41

Capítulo 1

Introducción

En los sistemas informáticos, el sesgo se refiere a la discriminación injusta y sistemática contra ciertos individuos o grupos, donde se niegan oportunidades o se asignan resultados indeseables basándose en motivos irrazonables o inapropiados [1]. En concreto, el sesgo de género se define como el trato sistemático y desigual basado en el propio género [2].

En la actualidad, existen muchos modelos que hacen uso de técnicas de procesamiento de lenguaje natural (NLP, por sus siglas en inglés) para resolver tareas específicas del lenguaje, tales como traducción automática, análisis de sentimientos, entre otros. Dichos modelos, que varían desde modelos de lenguaje Markovianos hasta redes neuronales profundas, son entrenados generalmente a partir de textos como noticias o literatura, que presentan un sesgo de género en el lenguaje [2]. Debido a esto, el sesgo de género puede presentarse en múltiples elementos del NLP [3], tales como los datos de entrenamiento, *word-embeddings* [4] (representaciones vectorizadas de palabras), y modelos pre entrenados.

En consecuencia, los modelos pueden realizar predicciones sesgadas, lo que se define como una disparidad en el resultado cuando al modelo se le entrega texto haciendo alusión a diferentes géneros [5]. Los modelos entrenados pueden incluso amplificar los sesgos presentes en los datos de entrenamiento, lo que puede traer consecuencias dañinas [3]. Por ejemplo, en el caso de los *word-embeddings* y el idioma inglés, se puede dar situaciones como: $\vec{man} - \vec{woman} \approx \vec{computer\ programmer} - \vec{homemaker}$ [6]. Esto podría generar predicciones sesgadas en un modelo de recomendación de empleo, o en un motor de búsqueda [7].

1.1. Motivación

Hasta el momento, se ha estudiado el problema de detectar, medir y mitigar la presencia de sesgo de género en distintos elementos de las tareas de NLP. Por ejemplo, se ha estudiado cómo medir y mitgar el sesgo de género en *word-embeddings* [6, 7], cómo medirlo en modelos de lenguaje [8], y cómo detectarlo y medirlo en tareas de aplicaciones posteriores de dichos modelos, como *machine-*

translation [9], generación de texto [10], entre otras [2]. También se ha contribuido elaborando conjuntos de datos etiquetados para facilitar la detección de sesgo de género en textos en inglés [11], y para detectar y clasificar contenidos o comportamientos misóginos en textos inglés, español e italiano [12].

Pese a estos avances, aún se presentan limitaciones en otros que requieren de mayor investigación. Por ejemplo, solo se ha hecho foco en dos géneros (femenino y masculino), existiendo la necesidad de incorporar la neutralidad de género [2]. Por otro lado, la mayor parte de las citas previas consideran mayoritariamente el idioma inglés, y se centran en la detección y mitigación del sesgo de género en los conjuntos de datos y algoritmos o modelos en sí, más que en el uso de aplicaciones posteriores para mitigar el sesgo de género en el texto. Dado esto, surge la motivación de *lograr una mitigación del sesgo de género directamente sobre texto* (en lugar de sobre otros elementos del NLP), específicamente para *texto en español*, mediante un modelo de aprendizaje automático supervisado.

1.2. Antecedentes

Frente a la pregunta *¿"Es posible mitigar el sesgo de género presente en texto en español mediante un modelo de aprendizaje automático supervisado?"*, una línea de trabajo que se presenta como alternativa de resolución es el desarrollo de una tarea de Transferencia de Estilo de Texto [13] (TST, por sus siglas en inglés), que consiste en reescribir una oración en un nuevo estilo, mientras se preserva su contenido semántico. Para ello, se puede requerir o no del uso de datos paralelos (frases en el estilo original pareadas con sus frases correspondientes en el estilo objetivo), dependiendo del enfoque a utilizar. Distintos enfoques, con y sin datos paralelos, han logrado buenos resultados en tareas de transferencia de sentimiento [14]. También se ha utilizado un enfoque no supervisado (sin datos paralelos) para la mitigación del sesgo de género en inglés, con resultados promisorios [15].

Las tareas de TST, al igual que otras tareas de NLP como *machine translation*, corresponden a tareas de tipo *secuencia a secuencia*, o *sequence to sequence* (Seq2Seq), llamadas así ya que implican la obtención de una secuencia de salida condicionado a una secuencia de entrada entregada. Para resolver estas tareas, comunmente se utiliza arquitecturas de redes neuronales profundas de tipo *encoder-decoder* [16, 17], lo cual también se ha utilizado en específico para el caso de TST [14, 15, 18].

1.2.1. Transformers

La evolución de las arquitecturas de tipo *encoder-decoder* en tareas de secuencia a secuencia comenzó con las redes recurrentes de tipo *long short-term memory* (LSTM) [19] propuestas en

1997, las cuales tienen una *memoria* en forma de estados de celda que les permite mantener la información a lo largo de secuencias largas. Posteriormente, en 2014, se introdujo en estas redes el mecanismo de *atención* [20], donde en lugar de tratar todas las partes de la secuencia de entrada por igual, el modelo *atiende* las partes de la entrada que son más relevantes para cada paso de la secuencia de salida. Tomando la idea del mecanismo de atención, en 2017 se introduce la arquitectura *transformer* [21], la cual solamente se basa en atención y no utiliza una red recursiva, permitiendo así un entrenamiento más rápido del modelo, y a su vez logrando mejores resultados.

Una función de atención se describe como una asignación de una consulta y un conjunto de pares clave-valor a una salida, donde la consulta, las claves, los valores y la salida son todos vectores. La salida se calcula como una suma ponderada de los valores, donde el peso asignado a cada valor se calcula mediante una función de compatibilidad de la consulta con la clave correspondiente. En *transformers*, se utiliza *atención de producto punto escalado* (Figura 1.2), donde la entrada consiste en consultas y claves de dimensión d_k, y valores de dimensión d_v. La función de atención se calcula en un conjunto de consultas simultáneamente, agrupadas en una matriz Q. Las claves y los valores también se agrupan en las matrices K y V, con lo que se calcula la matriz de salidas como:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.1}$$

Considerando lo anterior, se realiza además una proyección lineal de consultas, claves y valores h (número de cabezas) veces, lo que se denomina *Multi-Head Attention* (Figura 1.2). La función de atención se realiza entonces en paralelo en las versiones proyectadas de consultas, claves y valores. Luego, se concatenan y proyectan las versiones de todas las cabezas (para consultas, claves y valores), obteniéndose de esta forma los valores finales.

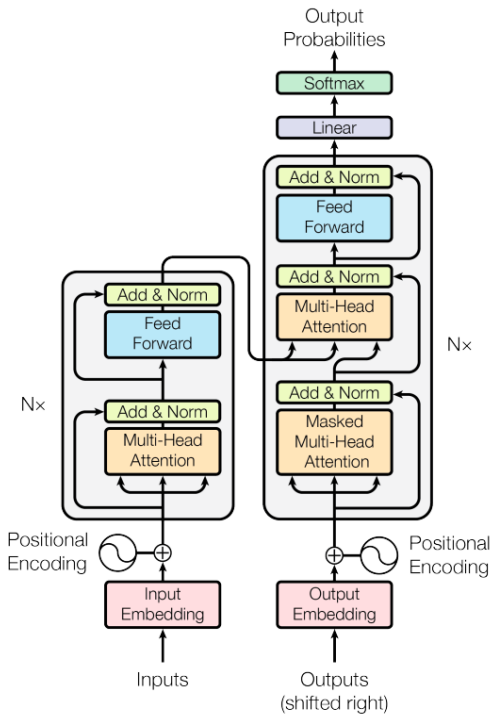


Figura 1.1: Arquitectura transformer, tomado de [21]. Utiliza capas de auto-atención apiladas y capas completamente conectadas punto por punto, tanto para el *encoder* (izquierda) como para el *decoder* (derecha).

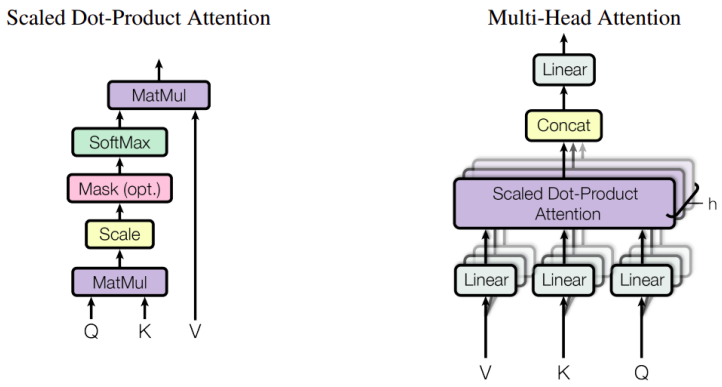


Figura 1.2: Atención de producto punto escalado (izquierda), y *Multi-Head Attention* (derecha), publicados en [21].

Los elementos a destacar de la arquitectura del *transformer* (Figura 1.1) son:

- *Input/Output embeddings*: Corresponde a la secuencia de entrada y salida respectivamente, las cuales han sido mapeadas a *tokens* [22], y posteriormente a *embeddings*. Los tokens corresponden a las unidades básicas dentro del texto, que no necesitan ser descompuestas en un procesamiento posterior, siendo una *palabra* el tipo de token más básico de token. Los *embeddings* corresponden a representaciones vectoriales de dichos tokens.

- *Positional encoding*: Son una forma de representar la posición de los tokens en una secuencia,

y son añadidos a los *embeddings*.

- *Encoder*: Recibe los *imput embeddings* junto con sus *positional encoding*. Consiste en 6 de bloques compuestos por una capa multi-cabeza de auto-atención, seguidas de una red *feed-forward* (FF). Se emplea además una conexión residual [23] alrededor de estas dos capas, seguido de una normalización [24].

- *Decoder*: Recibe los *output embeddings* junto con sus *positional encoding*. Tiene los mismos 6 bloques del *encoder*, con la diferencia de que se inserta una capa de *multi-head attention* que recibe la salida de los bloques *encoder*. Además, se modifica la capa de auto-atención de modo de evitar que las posiciones atiendan a posiciones posteriores, lo que se conoce como *masking*.

1.3. Modelos de lenguaje pre entrenados

La arquitectura *transformer* permitió la creación de una gran cantidad de modelos de lenguaje pre entrenados (PLM, por sus siglas en inglés), y en los últimos años (Figura 1.3), grandes modelos de lenguaje (LLM, por sus siglas en inglés), los cuales son utilizados ampliamente en la actualidad para la resolución de diferentes tareas de NLP [25], no solo tareas Seq2Seq.

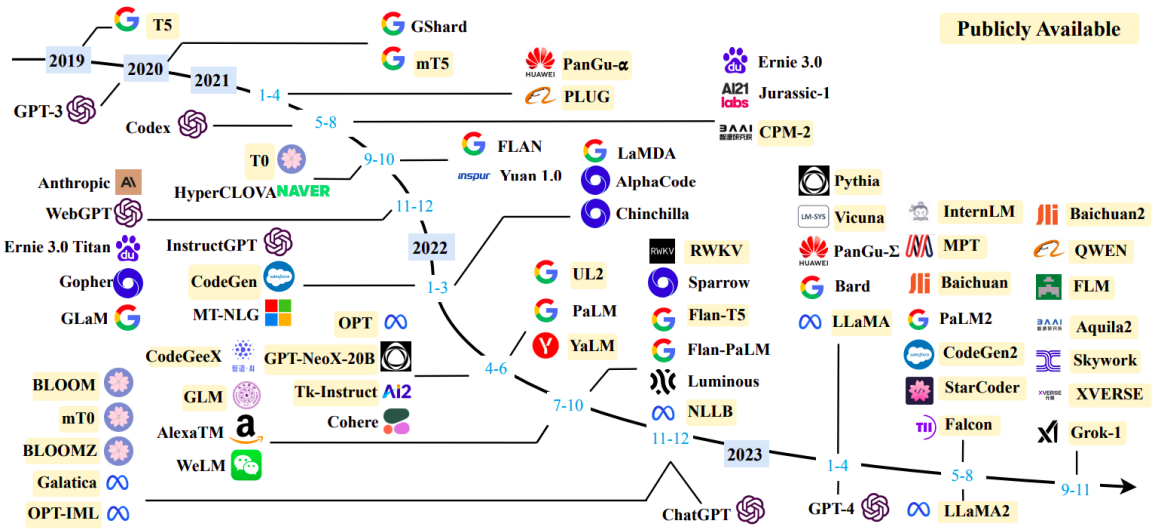


Figura 1.3: Línea de tiempo de grandes modelos de lenguaje (teniendo un tamaño superior a 10B) disponibles en los últimos años, tomado de [25].

El uso de PLM como base para tareas específicas de NLP ha demostrado ser ventajoso, ya que estos modelos, al ser entrenados en un gran volumen de datos, son capaces de generar representaciones vectorizadas de palabras (o *word-embeddings*) contextualizadas según el texto que les acompaña. Con esto se espera que estos modelos logren aprender tanto la sintaxis como gramática del lenguaje [26]. Si bien los PLM no son entrenados en una tarea específica, ofrecen una inicialización robusta, y es posible realizar *fine-tuning* sobre ellos, utilizando grandes conjuntos de datos con

ejemplos atingentes para ajustar el PLM, inicialmente agnóstico, hacia una tarea específica de NLP [27]. Así, para resolver una tarea específica, se puede hacer uso de los PLM ajustados, o incluso realizar un nuevo ajuste sobre ellos, utilizando un nuevo conjunto de datos [25].

Hoy en día, es frecuente el uso de LLM en combinación con técnicas como *few-shot* [28] (entregar como parte de la entrada ejemplos del resultado esperado), lo cual no actualiza gradientes ni realiza ajuste sobre los parámetros del modelo inicial, para su uso directo en tareas de NLP. Para el caso del uso de GPT-3, se ha demostrado alcanzar un gran rendimiento en tareas tales como traducción y respuesta a preguntas [27]. Esto supone una ventaja en términos computacionales y de datos, al no ser necesario un ajuste de parámetros ni contar con un gran volumen de datos. Sin embargo, en cuanto a rendimiento se ha encontrado una gran variabilidad de resultados para tareas *'Out of Domain'* (tareas no relacionadas con los datos de entrenamiento del modelo), dependiendo de la tarea específica y del tamaño del modelo utilizado, siendo en muchos casos mejor el rendimiento del ajuste de parámetros [29]. Otro aspecto a tener en cuenta del uso de LLM se da en cuanto a la disponibilidad. Por ejemplo, en el caso de modelos como GPT-3.5 o GPT-4, no es posible utilizarlos de forma local, lo que involucra tener que compartir información con un tercero.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un modelo de lenguaje que sea capaz de mitigar el sesgo de género presente en textos en español, específicamente en las comunicaciones escritas de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile (FCFM), mediante una tarea de TST con datos paralelos, llevada a cabo utilizando un PLM ajustado.

1.4.2. Objetivos específicos

- Definir tipos de sesgo de género acorde a las necesidades de la FCFM para sus comunicados escritos.
- Construir un conjunto de datos paralelos (con y sin sesgo de género) que contenga comunicados oficiales de la FCFM en la seccuón "Novedades"de U-Cursos.
- Ajustar un PLM, utilizando el conjunto de datos paralelos, para resolver la tarea de mitigación de sesgo de género en texto en español.
- Construir una aplicación con interfaz gráfica, que utilice el modelo ajustado, para la mitigación del sesgo de género en texto en español en tiempo real.
- Generar recomendaciones prácticas y sugerencias para futuras investigaciones relacionadas con la mitigación del sesgo de género en textos en español, particularmente en el ámbito académico.

1.5. Contribuciones

- Se entrega una metodología de trabajo útil para la construcción de datos paralelos para resolver una tarea de TST para la mitigación de sesgo de género.
- Se construye y entrega el un conjunto de datos paralelos, el cual puede ser utilizado para evaluar distintas metodologías para llevar a cabo la tarea de TST para la mitigación de sesgo de género en español.
- Se entrega una exploración del conjunto de datos construidos, en cuanto a la presencia de sesgo de género en los mensajes de "Novedades" de U-Cursos de la FCFM.
- Se entrega una metodología de ajuste de PLM en forma eficiente. Dicha metodología puede ser perfeccionada, y/o utilizada para mitigar el sesgo de género en datos de entrenamiento de modelos destinados a resolver otras tareas de NLP, de forma que dichos modelos no amplifiquen el sesgo de género que pueda estar presente inicialmente en sus datos de entrenamiento.
- Se construye y entrega un prototipo de herramienta de *software* que permite el uso del modelo ajustado para la mitigación del sesgo de género en texto en tiempo real.

Tanto el código fuente como el conjunto de datos se encuentra disponible en GitHub[1].

[1] https://github.com/GianniCatBug/spanish-gender-debias

Capítulo 2

Metodología

El problema consiste en una tarea de NLP de TST, concretamente de mitigación de sesgo de género. El objetivo es generar un documento de salida $\hat{d}_{\text{out}}$ donde se ha mitigado el sesgo de género y se ha preservado el contenido, a partir de un documento de entrada, el cual puede o no presentar sesgo de género. Los elementos de esta tarea, esquematizados en la Figura 2.1, son:

- *Entrada*: Documento de texto d_{in}, con o sin presencia de sesgo de género.
- *Salida esperada*: Documento de texto d_{out}, sin presencia de sesgo de género, que corresponde a la salida esperada del modelo.
- *Salida generada*: Documento de texto $\hat{d}_{\text{out}}$, con mitigación de sesgo de género, que corresponde a la salida del modelo ajustado.

El problema se resuelve entonces mediante el uso de aprendizaje supervisado, utilizando un corpus de datos paralelos. Por tanto, se tiene un conjunto de entrenamiento correspondiente a un corpus C_{train} que contiene m documentos d_{in} con o sin presencia de sesgo de género, donde para cada uno existe un documento d_{out} sin presencia de sesgo de género, que corresponde a su salida esperada. Durante el entrenamiento, tanto d_{in} como d_{out} se representan como concatenaciones de vectores o *word-embeddings* contextualizados, donde cada uno se asocia a un token del documento, el cual puede representar una palabra, sub palabra o caracter del mismo.

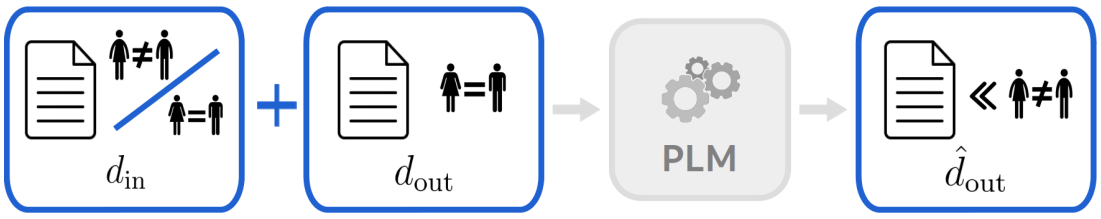


Figura 2.1: Elementos de la tarea a resolver

En cuanto a la arquitectura, a modo de comparativa se utiliza dos tipos de PLM:

- *Causal Language Model* (CausalLM): Corresponden a modelos tipo *decoder-only* (o *generativos*) de la arquitectura *transformer*. Se utilizan para la generación de texto. Si bien la tarea que se busca resolver se considera de tipo *sequence to sequence* (al tener un documento de salida esperado condicionado a un documento de entrada), el hecho de tener un modelo de tipo *decoder* entrenado en grandes volúmenes de datos permite que también pueda ser utilizado para este tipo de tareas, ya que cada token se genera en forma condicional a los tokens anteriores [25], los que en este caso corresponden al documento de entrada junto con la instrucción de eliminar el sesgo de género. Un ejemplo de este tipo son los modelos GPT [30].
- *Sequence to Sequence Language Model* (Seq2SeqLM): Corresponden a modelos de tipo *encoder-decoder* de la arquitectura *transformer*. Se utilizan para la generación de texto a partir de una entrada específica [25]. Un ejemplo de este tipo son los modelos T5 [31].

.

Los modelos escogidos, utilizando los datos paralelos de C_{train}, son posteriormente ajustados en forma eficiente en sus parámetros mediante *parameter efficient fine tuning*[2] (PEFT) con adaptadores cuantizados de bajo rango, y utilizando una única GPU. Con esto se define entonces los elementos del entrenamiento (Figura 2.2).

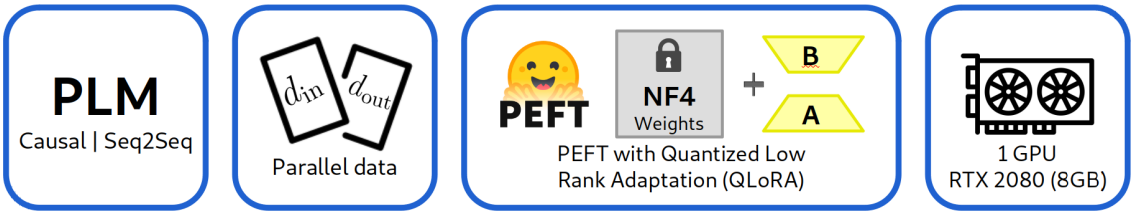


Figura 2.2: Elementos del entenamiento

El desempeño de los modelos ajustados se mide posteriormente mediante métricas de desempeño que permiten comparar las salidas generadas por el modelo $\hat{d}_{\text{out}}$ con sus correspondientes salidas esperadas d_{out}. Para ello se utiliza un corpus diferente al de entrenamiento, que corresponde a C_{test}.

En concreto, se desea aplicar esta tarea para la mitigación del sesgo de género en el contexto de comunicaciones escritas de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile (FCFM), tales como mensajes del portal U-Cursos[3], publicaciones en sitios web, entre otros. Esto se logra mediante el uso de una interfaz de programación de aplicaciones (API) que haga uso del modelo entrenado para entregar una versión mitigada en cuanto a género, a partir de un texto de entrada recibido mediante una interfaz de usuario (UI). Es importante recalcar que, en tales contextos, la corrección del sesgo de género es sugerida (en la UI) y no mandatoria.

[2] https://github.com/huggingface/peft
[3] https://www.u-cursos.cl/

2.1. Construcción del corpus

Se construyó un corpus C de 31,195 documentos paralelos $d_{\text{in}}, d_{\text{out}}$. A partir de C, se creó C_{train} con 21,836 documentos paralelos seleccionados aleatoriamente, para el *fine-tuning* de modelos, y C_{val} con los 9,359 documentos paralelos restantes, para validar el rendimiento del modelo durante su ajuste.

Se construyó además un corpus C_{test} con 853 documentos paralelos $d_{\text{in}}, \vec{d}_{\text{out}}$, donde en este caso cada instancia de $\vec{d}_{\text{out}}$ corresponde a un vector de posibles alternativas de versiones insesgadas para el correspondiente d_{in}.

2.1.1. Recolección de datos

El corpus C se construyó a partir de mensajes publicados en la sección "Novedades"del portal U-Cursos de la FCFM[4]. Se consideró los 8.868 mensajes publicados entre el 25 de octubre de 2007 y el 7 de agosto de 2023. Los mensajes se obtuvieron directamente desde el sitio web utilizando la técnica de *web scraping*, mediante Python 3.9[5] y los paquetes `requests`[6] y `BeautifulSoup`[7].

Posteriormente, cada mensaje se separó en frases mediante el método `sent_tokenize` del paquete `nltk`[8]. Con esto se obtuvo 45,792 frases, de las cuales se conserva solamente las que tuviesen al menos 12 caracteres y 2 palabras, y se elimina además frases duplicadas. De esta forma, se obtuvo un total de 31,195 frases. La separación de los mensajes en frases se hace con el propósito de facilitar el proceso de etiquetado manual, así como el ajuste de los modelos y las inferencias posteriores a partir de secuencias más pequeñas. Por lo tanto, el ajuste posterior de modelos se realiza en base a las frases dentro de los mensajes y no en base a los mensajes completos. Esto podría afectar el desempeño posterior del modelo frente a entradas de texto de gran longitud, pero se decide este enfoque debido a límite de recursos computacionales.

El mismo procedimiento anterior se aplicó a C_{test}, para el cual se utilizó 200 mensajes publicados en el mismo portal, entre el 23 de agosto de 2023 y el 6 de diciembre de 2023, pero preservando frases cortas. El proceso de llevar los mensajes a frases produce finalmente 853 frases.

2.1.2. Construcción de datos paralelos

Dado la tarea que se desea resolver, para cada documento d_{in} se debe asignar un documento paralelo d_{out}. Para ello, para cada frase de referencia se construyó manualmente, mediante personas instruidas en la construcción de textos sin sesgo de género, una versión con sesgo de género y una versión sin sesgo de género de la misma. Cabe destacar que muchas de las frases entregadas no

[4] https://www.u-cursos.cl/ingenieria/2/novedades_institucion
[5] https://docs.python.org/3.9/
[6] https://github.com/psf/requests
[7] https://www.crummy.com/software/BeautifulSoup/
[8] https://www.nltk.org/

corresponden a contenido del cual se pueda construir dichas versiones (por ejemplo, fechas, lugares, nombres de personas, entre otros), por lo que se instruyó a las personas etiquetadoras omitir estos casos.

A modo de estandarizar la forma de construcción de estas versiones, se creó un manual[9] para las personas etiquetadoras, donde se explica detalladamente distintas definiciones de sesgo de géneros (atingentes a los intereses de la FCFM), así como la forma de llenado de la planilla de datos. Además, a cada persona se le realizó una capacitación explicando el contenido del manual, y se les solicitó llenar una planilla de datos "piloto"de 20 frases. Una vez corroborada una correcta construcción de versiones pareadas para las frases del piloto, a cada persona se le asignó lotes de 1,000 frases para trabajar.

Finalmente, para construir C, para todas aquellas frases donde sí fue posible construir sus versiones con y sin sesgo de género, se escogió aleatoriamente entre su versión sesgada y su versión insesgada (con igual probabilidad) para construir los documentos d_{in}, y se utilizó siempre la versión insesgada para construir d_{out}. En los casos de frases donde no era posible construir versiones sesgadas e insesgadas, se utilizó la frase original tanto para d_{in} como d_{out}.

Para el caso de C_{test}, se utilizó siempre las frases originales como los documentos d_{in}. Para aquellos casos donde la frase de original presentase sesgo de género, se contruyó un correspondiente vector $\vec{d}_{\text{out}}$ de dimensión variable con distintas alternativas de versiones insesgadas. Para aquellos casos donde la frase original no presentase sesgo de género, o no fuese posible construir una versión sesgada o insesgada de ella, se construyó un correspondiente vector $\vec{d}_{\text{out}}$ de una dimensión cuyo único elemento corresponde a la frase de referencia.

En el *Anexo A* se encuentra disponible un breve análisis exploratorio de estos datos, del cual se destaca que en C el 30.8 % de los documentos corresponde a documentos donde sí es posible construir una versión sesgada y una insesgada a partir de la frase de referencia, mientras que en C_{test} esto corresponde al 39 %. De estos documentos, para el caso de C el 60.1 % corresponde a instancias donde la frase de referencia original sí presenta uno o más tipos de sesgo de género (siendo el mínimo al considerar por año 27.5 % para el año 2022, y el máximo 96.4 % para el año 2007). En cuanto a C_{test}, esta proporción corresponde solo al 12.3 %.

2.2. Modelos utilizados

En primer lugar se realizó una serie de solicitudes *"zero-shot"* utilizando el modelo GPT-3.5, solicitando mediante *prompt* solamente que se detectara la presencia o ausencia de sesgo de género en una muestra aleatoria de 1.816 mensajes de la sección "Novedades"del portal de U-Cursos de la FCFM, entre el 25 de octubre de 2007, y el 3 de abril de 2023. En este caso no se dio ejemplos,

[9] https://docs.google.com/document/d/1ZPFM8bUSvKjyA-QgZL8suog8sAzOCT9KG0qPaGx5aKs/edit?usp=sharing

debido a la gran cantidad de posibilidades donde se podría presentar sesgo de género [32]. Este ejercicio tiene como utilidad explorar en forma cualitativa qué es lo que el modelo considera "sesgo de género". Los hallazgos encontrados se presentan en la sección de *Resultados*.

Por otro lado, se utilizó dos tipos de PLM para realizar un ajuste sobre sus parámetros, utilizando el corpus paralelo construido, para la tarea de transferencia de estilo de texto de mitigación de sesgo de género:

- *CausalLM*: Se utiliza el modelo Falcon-7B-Instruct[10], el cual fue construido por el *Technology Innovation Institute* (TII)[11] y se basa en Falcon-7B, aplicándole *fine-tuning* utilizando datos de chat e instrucciones.

  La clave para la alta calidad de los modelos Falcon es su conjunto de entrenamiento, basado predominantemente (más del 80 %) en *RefinedWeb* [33], el cual es un conjunto de datos web masivos basados en CommonCrawl [12]. El modelo Falcon-7B se ha entrenado sobre 1 trillón de tokens.

  En cuanto a la arquitectura del modelo (Figura 2.1 (a)), según lo informado en el blog de HuggingFace[13], y al revisar el código fuente [14], se observa las siguientes diferencias respecto de una arquitectura clásica *decoder-only*:

  - *FlashAttention* [34]: Es un algoritmo que mejora la eficiencia de los modelos basados en *transformers*, permitiendo un entrenamiento e inferencia más rápidos. En lugar de cargar y escribir claves, consultas y valores desde la memoria de alto ancho de banda (HBM) a la SRAM en chip de la GPU en cada paso de atención, este algoritmo realiza esta operación una sola vez.
  - *Rotary Positional Embeddings (RoPE)* [35]: Los algoritmos *decoder-only* deben agregar la información de la posición de cada token dentro de la secuencia, lo cual se puede realizar con distintas técnicas de *position-embedding*. El caso específico de RoPE, se considera tanto la posición global de un token dentro de la secuencia (posición absoluta), como la posición basada en la distancia entre tokens (posición relativa).
  - *MultiQuery Attention* [36]: Este es un mecanismo de atención donde en lugar de realizar una proyección separada de claves y valores para cada cabeza, se comparte los *embeddings* de claves y valores a través de las cabezas de atención. Esto puede mejorar significativamente la velocidad de inferencia.
  - *Capas de atención y FF paralelas*: El proceso de atención se realiza en forma paralela a la transformación FF de los datos, a diferencia de otras arquitecturas donde FF suele ser un paso posterior a la atención. Esto también puede permitir una mejora en la velocidad de inferencia del modelo.

[10] https://huggingface.co/tiiuae/falcon-7b-instruct
[11] https://www.tii.ae/
[12] https://commoncrawl.org/
[13] https://huggingface.co/blog/falcon
[14] https://huggingface.co/tiiuae/falcon-7b-instruct/blob/main/modeling_falcon.py

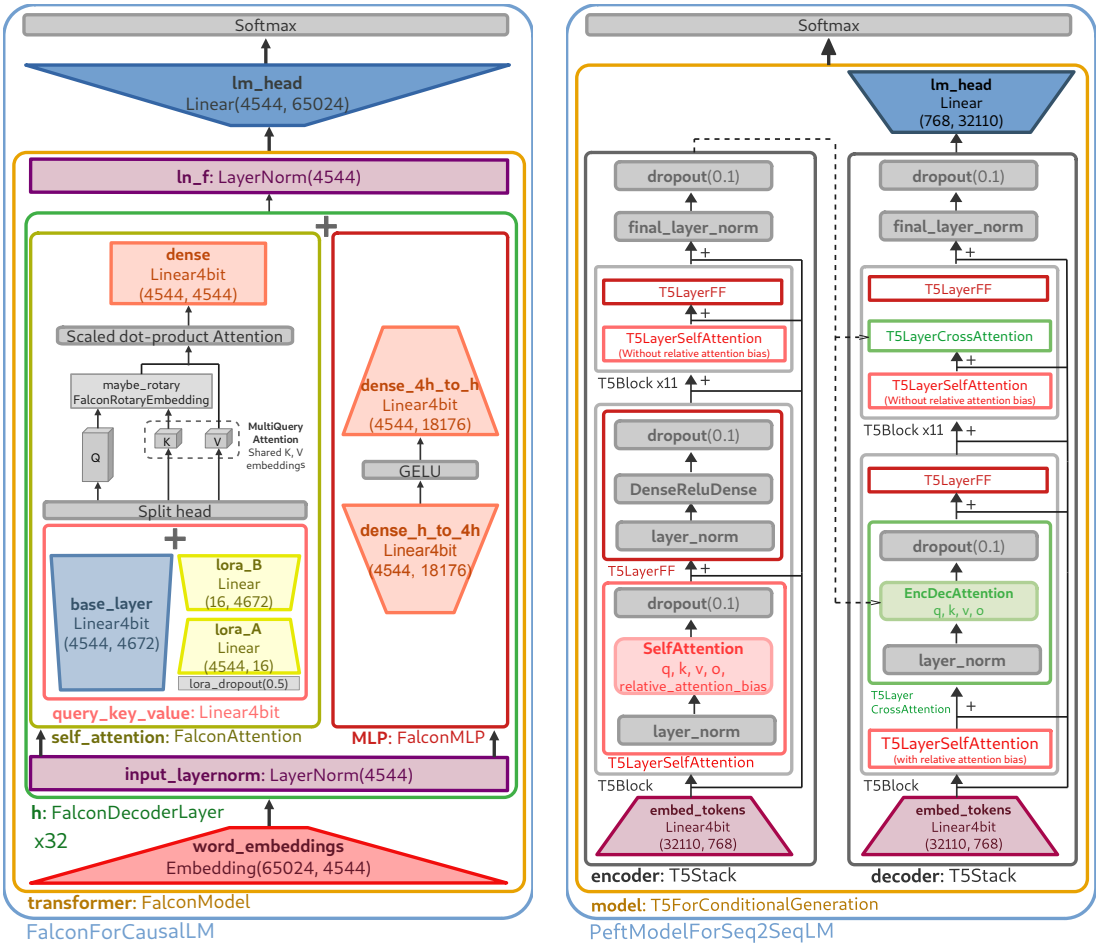
- *Seq2SeqLM*: Se utiliza el modelo FLAN-T5 [37] base[15]. Este modelo se basa en T5 (text-to-text-transfer-transformer) [31], pero se le ha aplicado *fine-tuning* para alrededor de 1.000 tareas específicas, y su versión *base* tiene 248M de parámetros.

  Algo a tener en cuenta, es que FLAN-T5 puede usarse potencialmente para la generación de lenguaje de forma dañina [38]. Por tal razón, FLAN-T5 no debe usarse directamente en ninguna aplicación, sin una evaluación previa de las preocupaciones de seguridad y equidad específicas de la aplicación.

  La arquitectura del modelo T5 corresponde a un *encoder-decoder* que sigue de cerca la implementación original del *transformer*. Se destaca como diferencias, respecto del *transformer* original, lo siguiente:

  - *Capa de normalización*: La capa de normalización se ubica fuera del camino residual. De esta forma, las activaciones solamente se re escalan, y no se aplica sesgo aditivo.
  - *Positional Embedding*: Se utiliza una forma simplificada, donde cada posición es simplemente un escalar que se agrega al *logit* correspondiente utilizado para calcular los pesos de la atención.

[15] https://huggingface.co/google/flan-t5-base



(a) Falcon-7b-instruct. Se incluye adaptador de bajo rango sobre la capa de claves, llaves y valores. Las capas lineales de 4bit cargan los pesos en 4bit, que para la inferencia son descuantizados a bfloat16.

(b) FLAN-T5. Se presenta la arquitectura general del modelo. Pese a no mostrarse en la figura, se aplica adaptadores de bajo rango sobre las capas de consultas y valores tanto para SelfAttention como EncDecAttention (con entrada y salida de 768, y dropout de 0.5). Además de en estas capas, se aplica además quantización de 4bit en las capas de claves y salida de la atención, y sobre las capas de pesos del bloque DenseReluDense

Figura 2.3: Arquitectura de PLMs escogidos (elaboración propia)

Los dos PLM escogidos se encuentran disponibles gratuitamente para descarga y ajuste de sus parámetros en forma local. Los LLM son aquellos que tienen al menos decenas de billones de parámetros [26], por lo que por sus tamaños los modelos escogidos no llegan a ser considerados LLM. De todas formas, tienen un tamaño y complejidad considerable, que se asume es suficiente para la tarea a resolver.

2.2.1. Ajuste fino de parámetros en forma eficiente

En cuanto a los PLM escogidos, pese a no ser de los modelos de mayor tamaño, de todas formas para su ajuste se requiere de una cantidad de recursos que pueden no estar disponibles en un entorno de bajo costo. Es por ello que para ambos modelos seleccionados se realiza un ajuste de parámetros

en forma eficiente (PEFT, por sus siglas en inglés), utilizando el paquete `peft`[16] de Python desarrollado por HuggingFace. Este paquete permite adapatar de manera eficiente los PLM a varias tareas, ajustando solo un pequeño número de parámetros adicionales mientras se congelan los parámetros del PLM, reduciendo así en gran medida los costos computacionales y de almacenamiento [17].

La metodología PEFT utilizada en ambos casos es mediante *Low Rank Adaptation* (LoRA) [39] en forma cuantizada (QLoRA) [40]. LoRA tiene como objetivo reducir el número de parámetros entrenables, aprendiendo exclusivamente pares de matrices de descomposición de rangos, mientras mantiene congelados los parámetros del modelo original. A su vez, QLoRA retropropaga los gradientes a través del PLM, el cual tiene sus parámetros congelados y cuantizados en 4-bit, hacia los adaptadores de bajo rango. De esta forma es posible aplicar ajuste fino sobre modelos de billones de parámetros en una única GPU de 48 GB[18], mientras se mantiene el rendimiento en comparación a un ajuste fino completo en 16-bit.

En la Figura 2.3 (a) se observa que la aplicación de LoRA se realiza en la capa *query_key_value* de los bloques de atención, mientras que la cuantización en 4 bits se realiza en los pesos originales de la capa lineal del bloque, además de otras capas lineales del *decoder*. Para el caso de T5, solo se aplica LoRA sobre las consultas y valores, pero la cuantización en 4 bit se realiza en todas las matrices de atención, además de sobre los pesos de las capas lineales del bloque FF.

2.3. Preparación de datos

En las sub secciones siguientes se detalla las distintas etapas realizadas en cuanto a la preparación de datos para el ajuste de modelos, considerando las cualidades de cada uno.

2.3.1. Tokenización

Para el uso de cada modelo, se debe aplicar su tokenizador específico sobre C_{train} y C_{val}, de forma de obtener un identificador numérico de cada token, el cual luego es utilizado como identificador para que así el modelo pueda obtener la representación vectorial (*word-embedding* contextualizado) de cada palabra o sub palabra. Dado que el contenido de C puede ser de un dominio diferente al utilizado para cada modelo en la construcción de sus capas de *embedding*, se debe explorar si la tokenización del corpus genera o no gran cantidad de tokens <*unk*>, es decir, palabras o sub palabras para las cuales el PLM no posee un *word-embedding*. Si bien en esta tesis no se tiene como objetivo realizar ajuste sobre la capa de *word-embeddings*, por lo que no se podría obtener una representación vectorial contextualizada de estas palabras y sub palabras, sí es importante añadir un índice para dichos tokens de forma que al momento de la decodificación (generación de nuevos textos en el posterior uso del modelo) estos sean llevados a la palabra o sub palabra deseada.

[16] https://github.com/huggingface/peft
[17] https://huggingface.co/docs/peft/index
[18] A modo de referencia, una instancia gratuita de Google Colab proporciona una GPU de 16 GB, y una instancia Pro proporciona una GPU de 40 GB

Para el caso del tokenizador de FLAN-T5, al ser aplicado sobre C, se obtuvo 28.022 ocurrencias de tokens <*unk*>, lo cual representa a tokens no conocidos por el tokenizador del modelo. Esto corresponde al 3.9 % del total de tokens de C, estando entre los tokens desconocidos más frecuentes los tokens {'í', 'ñ', 'ú', 'Á', '¡', '¿', 'Í', 'Ó', 'Ñ', 'Ú'}. Estos tokens fueron agregados al tokenizador, con el cual luego se ajusta el tamaño de *token embeddings* del modelo a ajustar. De esta forma, la capa de *embeddings* resulta con dimensiones $32,110 \times 768$, siendo la primera dimensión la cantidad de tokens del vocabulario, y la segunda la dimensión de los estados ocultos de la red. En cambio, en el caso de Falcon, al utilizar su tokenizador sobre C no se obtiene ocurrencias de tokens desconocidos. Por tanto, se mantiene la dimensión original de la capa de *embeddings* de $65,024 \times 4,544$.

2.3.2. Añadir Prompt

En el caso de Seq2SeqLM, para el ajuste del modelo, se recomienda añadir a cada documento de entrada una *instrucción* o *prefijo* que indique la tarea que se desea realizar para generar el documento de salida [31]. Por ello, para cada instancia original de d_{in}, se antepone la instrucción `Eliminar sesgo de género del siguiente texto:` .

En CausalLM a menudo se debe seguir un formato que incluye palabras reservadas para la *instrucción* o *pregunta* (tarea que se desea resolver), el *contexto* (corresponde a d_{in}) y la *respuesta* (corresponde a d_{out}). Para el caso específico de *Falcon instruct*, se recomienda incluir la instrucción de la tarea a realizar, junto con el texto del documento de entrada (contexto), a continuación del término <*human*>, y la respuesta esperada a continuación del término <*assistant*>. De esta forma, cada d_{in} de C_{train} se redefine como $d'_{\text{in}} =$ <*human*>*: "¿Puedes reescribir el siguiente texto sin sesgo de género?"* d_{in} <assistant>: d_{out}. Luego, tanto para C_{test} como para las inferencias del modelo ajustado, d_{in} se redefine como $d'_{in} =$<*human*>*: "¿Puedes reescribir el siguiente texto sin sesgo de género?"* d_{in} <*assistant*>*:*, con lo que $\hat{d}_{\text{out}}$ corresponde al texto a continuación de <*assistant*>*:* de la generación obtenida.

2.3.3. Largo máximo de secuencia

Para el caso de Seq2SeqLM, otro aspecto importante a considerar es el *máximo largo de secuencia* permitido, tanto para el *encoder* como el *decoder*. Para ello, se realiza una exploración de la cantidad de tokens (utilizando el tokenizador de FLAN-T5) tanto para cada d_{in} (con su prefijo) como cada d_{out} de C_{train}, con lo que se obtiene los resultados de la *Tabla 2.1*. En base a esta información, se decide utilizar como largo máximo de secuencia los valores de percentil 99 tanto para *encoder* como *decoder*.

Tabla 2.1: Estadística descriptiva de cantidad de Tokens por documento de entrenamiento

Estadístico	**FLAN-T5** d_{in}	**FLAN-T5** d_{out}	**Falcon** d'_{in}
count	21,836	21,836	21,836
mean	81.04	63.46	115.4
std	50.78	51.14	76.83
min	20	2	29
25 %	48	30	69
50 %	70	52	99
75 %	101	84	141
99 %	244	228	359
max	2,629	2,611	4,923

De esta formas, secuencias más largas de d_{in} son truncadas hasta 244 tokens, y secuencias más cortas son rellenadas con el token especial de *padding (<pad>)* hasta 244 tokens. Lo mismo ocurre con las secuencias de d_{out}, solo que hasta 228 tokens. Se debe considerar que al momento de hacer esta preparación de datos, a su vez se asigna el identificador -100 al token de *<pad>*, y posteriormente se indica en la instancia de la clase `DataCollatorForSeq2Seq` (que permite armar los lotes de entrenamiento) que este valor corresponde al token de *padding*, de manera que sea ignorado durante el cálculo de la pérdida.

Para el caso de CausalLM, no se utiliza el percentil 99 de los valores de largo de secuencia obtenidos para los d'_{in} de C_{train} (correspondiente a 369 tokens), debido a que no es posible aplicar el ajuste del modelo con los recursos computacionales disponibles y dicho largo máximo de secuencia. Por tal razón, se utiliza un largo máximo de secuencia de 250, lo que equivale aproximadamente al percentil 96.

2.4. Evaluación de modelos ajustados

Para evaluar el desempeño de los modelos ajustados se utiliza las métricas BLEU [41] y ROUGE [42]. Estas métricas son ampliamente utilizadas en la evaluación de modelos de generación de texto, especialmente en tareas de secuencia a secuencia como la traducción automática y el resumen de texto respectivamente. A grandes rasgos, miden la similitud entre las secuencias de entrada y de salida.

- *BLEU (Bilingual Evaluation Understudy)*: Es una métrica centrada en la precisión que calcula cuántos n-gramas (1 a 4) del texto generado por el modelo aparecen en el(los) texto(s) de referencia. Aunque originalmente fue diseñada para la evaluación de la traducción automática, su uso se ha extendido a otras tareas de generación de texto, como TST cuando se dispone de

datos paralelos [18]. Se calcula como:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.1}$$

Donde:

- BP: Corresponde a la penalización por brevedad (*Brevity Penalty*), que penaliza las generaciones más cortas que las referencias.
- N: Cantidad de n-gramas a considerar. En este caso, se utiliza el valor por defecto de 4.
- w_n: Corresponde a los pesos para cada orden de n-gramas. En este caso, se utiliza el valor por defecto de manera uniforme, es decir, $w_n = 0.25$.
- p_n: Precisión modificada de los n-gramas. Corresponde a la fracción de n-gramas en el texto generado que aparecen en el texto de referencia.

- *ROUGE (Recall-Oriented Understudy for Gisting Evaluation)*: Corresponden un conjunto de métricas que se utilizan para comparar la calidad de un texto generado por el modelo, midiendo cuántos n-gramas (1 a 2) del texto de referencia aparecen en el texto generado por el modelo. Incluye también la evaluación de concordancia de la subsecuencia común más larga de palabras entre los dos textos, donde las palabras no necesitan estar en el mismo orden. Las variantes utilizadas son:
  - ROUGE-N: Calcula la precisión, *recall* y F1-score de n-gramas entre el texto generado y un conjunto de referencias. Se utiliza ROUGE-1 y ROUGE-2. Se calcula como:

$$ROUGE - N = \frac{\sum_{S \in \text{Referencias}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \text{Referencias}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)} \tag{2.2}$$

    Donde n corresponde al largo del n-grama gram_n, y $\text{Count}_{\text{match}}(\text{gram}_n)$ es el número máximo de n-gramas que coexisten en un texto generado y un conjunto de textos de referencia.
  - ROUGE-L: Calcula la precisión, *recall* y F1-score de la misma forma que ROUGE-N, pero reemplazando la coincidencia de n-gramas con la coincidencia de las subsecuencias más largas (LCS, por sus siglas en inglés).

  Para cada texto generado, se considera el promedio del *F1-score* entre las 3 métricas.

En específico, se calcula el puntaje BLEU y ROUGE entre cada $\hat{d}_{\text{out}}$ generado a partir de su correspondiente d_{in}, y su(s) salida(s) esperada(s) correspondiente(s) d_{out}, utilizando C_{test}. De esta forma, se evalúa cada métrica en 4 escenarios:

- General: Consdiera los $\hat{d}_{\text{out}}$ para todos los d_{in}. Con sesgo: Considera solo los $\hat{d}_{\text{out}}$ asociados a d_{in} sesgables que sí poseen sesgo de género.

- Sin sesgo: Considera solo los $\hat{d}_{\text{out}}$ asociados a d_{in} sesgables que no poseen sesgo de género. Se espera que para estos casos se de que $d_{\text{in}} = \vec{d}_{\text{out}_1} = \hat{d}_{\text{out}}$.

- No sesgable: Considera solo los $\hat{d}_{\text{out}}$ asociados a d_{in} que no son género sesgables. Se espera que para estos casos se de que $d_{\text{in}} = \vec{d}_{\text{out}_1} = \hat{d}_{\text{out}}$.

Por la definición de las métricas a utilizar, se puede obtener una sobre estimación del desempeño del modelo, ya que para el caso de la mitigación del sesgo de género, la superposición de *n-gramas*, incluso entre las versiones sesgadas e insesgadas, es alta. Es por ello, que se considera además el BLEU obtenido entre cada d_{in} y sus d_{out} correspondientes, y esto se compara con el BLEU obtenido entre $\hat{d}_{\text{out}}$ y d_{out}. Las distintas situaciones posibles se presentan en la *Tabla 2.2*. Se considera solamente BLEU ya que la tarea a evaluar se acerca más a una tarea de traducción que de resumen.

Tabla 2.2: Situaciones de comparación BLEU

Tipo d_{in}	$\text{BLEU}(\hat{d}_{\text{out}}, d_{\text{in}}) - \text{BLEU}(d_{\text{in}}, d_{\text{out}})$		
	0	< 0	> 0
Con sesgo	Incorrecto	Revisión manual	Correcto
Sin sesgo	Correcto	Revisión manual	-
No sesgable	Correcto	Revisión manual	-

Los resultados deseados corresponden a obtener:

- Una diferencia mayor a cero cuando:
  - d_{in} tiene sesgo de género, ya que esto indicaría una mitigación del sesgo de género. En esta situación, la salida generada por el modelo es más similar a la salida esperada que la entrada original.

- Una diferencia igual a cero cuando:
  - d_{in} no tiene sesgo de género, o no es posible construir una versión sesgada de éste. Es decir, en esta situación se espera que la salida generada por el modelo sea igual a la entrada original.

Por otro lado, los resultados no deseados (o a revisar) corresponden a:

- No obtener diferencia cuando:
  - d_{in} tiene sesgo de género, ya que esto indicaría que no se ha realizado una mitigación del sesgo de género. En esta situación, la salida generada por el modelo es igual a la entrada original.

- Obtener diferencia menor a 0 cuando:
  - d_{in} tiene sesgo de género, ya que esto indicaría que se ha generado una salida diferente a la entrada, pero es menos similar a la salida esperada que la entrada. Requiere una revisión manual caso a caso.

– d_{in} no tiene sesgo de género, o no es posible construir una versión sesgada de éste. Requiere una revisión manual, ya que en esta situación el modelo ha generado una salida diferente a la entrada, cuando se espera que sean iguales.

Para las generaciones que requieren una revisión manual, se debe revisar cada salida generada, y determinar si la generación realizada por el modelo es correcta o incorrecta.

2.5. Aplicación Web

A modo de explorar el uso de los modelos ajustados en un entorno de aplicación real del objetivo que se busca alcanzar, es decir, mitigar el sesgo de género en comunicaciones escritas de la FCFM, se crea una pequeña aplicación web *demo* consistente en una API y una interfaz web de usuario. La API se construye utilizando el *framework* FastAPI[19] de Python, y la interfaz web se construye utilizando HTML y javascript nativo. La aplicación se ejecuta, para efectos de demostración, en un entorno local.

En el esquema de la Figura 2.4, se observa que la interfaz de usuario consiste en un cuadro de texto, donde a medida que se va escribiendo en éste, se envía solicitudes con el texto introducido a la API. La API utiliza este texto como entrada para el modelo ajustado escogido (añadiendo sufijo y/o prefijo según el modelo escogido), y retorna tanto el texto original de entrada como el texto generado por el modelo [20]. Además de lo anterior, se retorna también un diccionario indicando las posiciones del texto original que deben ser sustituidas, y los textos correspondientes a reemplazar en cada posición, con tal de mitigar el sesgo de género en el texto original. Con esta información, la interfaz agrega botones con una sugerencia de cambio para cada caso. Finalmente, en caso de que el usuario acepte la sugerencia, al hacer click en el botón referente a cada porción de texto, ésta se modifica en el texto original por la sustitución recomendada.

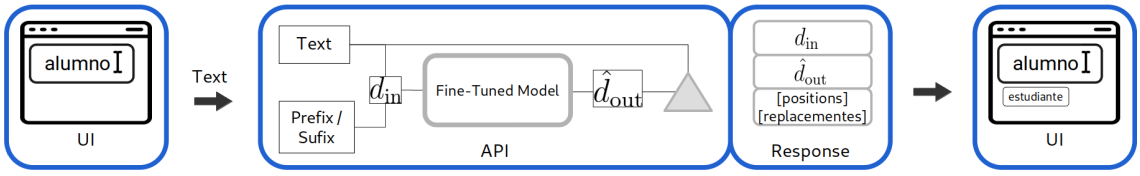


Figura 2.4: Uso del modelo ajustado

Las posiciones a substituir del texto original, así como las palabras de reemplazo a utilizar, se obtienen mediante el método `get_opcodes` de una instancia de la clase `SequenceMatcher` del paquete `difflib`[21] de Python.

[19] https://fastapi.tiangolo.com/

[20] Para el caso de CausalLM, se retorna solamente la generación posterior al token especial <*assistant*>, eliminando ocurrencias duplicadas e incompletas

[21] https://docs.python.org/3/library/difflib.html

Capítulo 3

Experimentos

3.1. Congiruación general de hiperparámetros

Ambos modelos se entrenaron siguiendo tutoriales y ejemplos entregados por el equipo de HuggingFace [22], y de QLoRA [23], por lo que la mayoría de los hiperparámetros utilizados corresponden a los utilizados en dichos tutoriales. Ambos modelos base (`google/flan-t5-base` para Seq2SeqLM y `vilsonrodrigues/falcon-7b-instruct-sharded` para CausalLM) se cargaron en forma cuantizada utilizando el paquete `bitsandbytes`[24], con los siguientes valores (en base a las recomendaciones de HuggingFace)[25],[26]:

- `load_in_4bit=True`: Indica que los pesos del modelo base se cargan y almacenan en una precisión de 4 bits.
- `bnb_4bit_use_double_quant=True`: Activa la doble cuantización, de forma que se utiliza una segunda cuantización después de la primera para ahorrar un adicional de 0.4 bits por parámetro.
- `bnb_4bit_quant_type="nf4"`: Indica el tipo de datos utilizado para la cuantización, en este caso flotante normal de 4 bits, el cual es teóricamente óptimo para pesos distribuidos normalmente.
- `bnb_4bit_compute_dtype=torch.bfloat16`: Asigna el tipo de dato de las entradas a `bfloat16`, solamente para el momento de cómputo.

En cuanto a la configuración PEFT, se utilizó una configuración de LoRA para una tarea de tipo `SEQ_2_SEQ_LM` para FLAN-T5, y de tipo `CAUSAL_LM` para Falcon. El resto de los valores de configuración utilizados son los mismos para ambos modelos, y son los siguientes:

[22] https://github.com/philschmid/deep-learning-pytorch-huggingface
[23] https://github.com/artidoro/qlora
[24] https://github.com/TimDettmers/bitsandbytes
[25] https://huggingface.co/docs/transformers/main/main_classes/quantizacion
[26] https://huggingface.co/blog/4bit-transformers-bitsandbytes

- `r=16`: Se utiliza rango o *rank* 16 para las matrices de actualización en LoRA. Un rango más bajo resulta en matrices de actualización más pequeñas con menos parámetros entrenables. Las recomendaciones varían entre valores de 1 y 32, pero para este caso se decide trabajar con 16.

- `lora_alpha=32`: Corresponde al factor de escala de LoRA. Determina cuánta importancia se le da a las nuevas actualizaciones de pesos de las matrices del adaptador LoRA cuando se añaden a los pesos pre entrenados originales. En general se utiliza el valor de 32.

- `bias="none"`: Indica que los parámetros de sesgo no son entrenados.

- `target_modules`: Corresponde a una lista de los nombres de los módulos a los que se aplican las matrices de actualización de LoRA. Siguiendo las recomendaciones mencionadas previamente, en ambos casos se aplica LoRA a las capas de atención, con la diferencia de que para el caso de FLAN-T5 se aplica solamente a consultas `q` y valores `v` (lo cual es posible ya que se tiene capas separadas para consultas, claves, valores y salidas), y para Falcon se aplica sobre la capa conjunta de consultas, claves y valores `query_key_value`.

- `lora_dropout=0.05`: Corresponde a la probabilidad de *dropout* para las capas de LoRA, con el fin de prevenir el sobreajuste durante el entrenamiento del modelo.

En cuanto a recursos, se utilizó una única tarjeta GPU NVIDIA GeForce RTX 2080 con 8 GB de memoria de video, un procesador Intel Core i7-10700KF de 3.8 GHz, y 16 GB de memoria RAM.

Los pesos adicionales (o adaptador), que corresponden al ajuste del modelo del *checkpoint* con menor pérdida en validación, fueron subidos a HuggingFace, quedando disponible en forma pública, junto con su configuración. El adaptador tiene un peso de 7.13 MB para el caso de FLAN-T5[27], y 18.9 MB para el caso de Falcon[28].

3.2. Seq2SeqLM

El modelo base entrenado cuenta con 249,319,680 parámetros, de los cuales 1,769,472 son entrenables (0.71 %). Para los hiperparámetros y configuración de entrenamiento, se utilizó una instancia de `Seq2SeqTrainingArguments`[29], utilizando la mayoría de los valores por defecto, y modificando algunos según las recomendaciones previamente citadas. Dentro de los valores a destacar, se utiliza:

- `per_device_train_batch_size=4`: Define el tamaño del lote de entrenamiento por dispositivo. En este caso, se procesarán 4 ejemplos a la vez en cada dispositivo durante el entrenamiento.

[27] https://huggingface.co/GianniCatBug/flan-base-4bit-005-gender-debias-spanish
[28] https://huggingface.co/GianniCatBug/falcon-7b-4bit-005-gender-debias-spanish. Los mejores resultados se obtuvieron con la revisión 87ae173, que es la que se utilizó para los resultados mostrados en esta tesis
[29] https://huggingface.co/docs/transformers/v4.35.0/en/main_classes/trainer#transformers.TrainingArguments

- `gradient_accumulation_steps=4`: Especifica el número de pasos de actualización para acumular antes de realizar una actualización de los parámetros. En este caso, los gradientes se acumularán durante 4 pasos antes de realizar una actualización.
- `optim=paged_adamw_8bit`: Corresponde a una variante del optimizador Adam [43]. En lugar de agregar estados del optimizador como Adafactor, mantiene el estado completo y lo cuantiza, lo que ahorra memoria y acelera el entrenamiento.
- `learning_rate=1e-3`: Valor inicial de tasa de aprendizaje. Se utiliza el mismo valor de las recomendaciones citadas.
- `num_train_epochs=5`: Número total de épocas de entrenamiento.

Se utilizó la función de pérdida del modelo base, que corresponde a *Cross Entropy Loss*. Para 5 épocas y un tamaño de lote efectivo de 16 (los gradientes se actualizan después de cada 16 ejemplos), el tiempo de entrenamiento fue de 3 horas, en 6,820 pasos de actualización del gradiente. Utilizando una calculadora del impacto del aprendizaje de máquinas [30], se estima que esto corresponde a la emisión de 0.28 Kg de CO_2, lo que equivale a 1.13 Km de conducción de un auto promedio o 0.14 Kg de carbón quemados.

Cada 682 pasos, se efectuó un *checkpoint* del modelo. En la Figura 3.1 (a), se muestra los valores de pérdida obtenidos tanto en entrenamiento como validación. El mejor modelo se obtuvo entonces en el último *checkpoint* (6,820), lo que corresponde a la época 5, con una pérdida de 0.024 en el conjunto de validación. En la Figura 3.2 (a) se muestra el decaimiento de la tasa de aprendizaje durante el ajuste del modelo, llegando a 0.0.

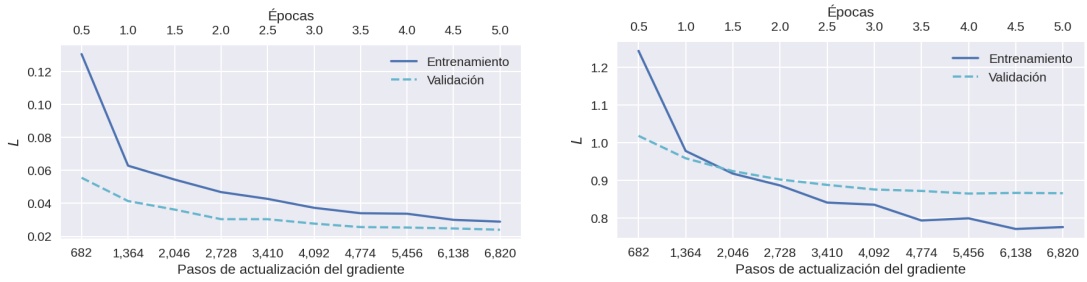
3.3. CausalLM

El modelo base entrenado cuenta con 6,926,439,296 parámetros, de los cuales 4,718,592 son entrenables (0.07 %). Para los hiperparámetros y configuración de entrenamiento, se utilizó una instancia de `TrainingArguments`, utilizando la mayoría de los valores por defecto, y modificando algunos según las recomendaciones previamente citadas. A diferencia del experimento para Seq2Seq, se utiliza una tasa de aprendizaje de 2e-4, y un planificador de la tasa de aprendizaje de tipo coseno. Se utiliza la función de pérdida del modelo base, que corresponde a *Cross Entropy Loss*.

Para 5 épocas y un tamaño efectivo de lote de 16, el tiempo de entrenamiento fue de 20.2 horas, en 6,820 pasos de actualización del gradiente. Se estima que esto corresponde a la emisión de 1.87 Kg de CO_2, lo que equivale a 7.56 Km de conducción de un auto promedio o 0.94 Kg de carbón quemados.

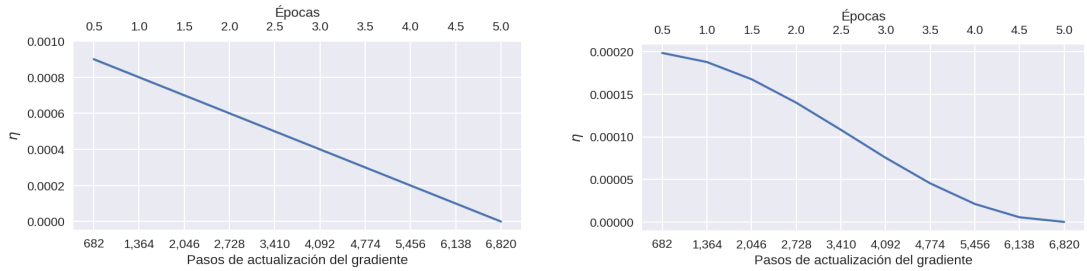
[30] https://mlco2.github.io/impact#compute

Se efectuó un *checkpoint* del modelo cada 682 pasos. En la Figura 3.1 (b) se observa que a partir de 2,046 pasos (época 1.5), el modelo comienza a presentar una pérdida mayor en validación que en entrenamiento, indicando un posible sobreajuste. Sin embargo, se conserva el modelo con menor pérdida en el conjunto de validación, la cual corresponde a 0.8647, obtenida en la época 4 (5,456 pasos). El decaimiendo de la tasa de aprendizaje, Figura 3.2 (b), llega a 0.0 en la quinta época, pero con un valor de 2.1e-05 al momento de alcanzar la menor pérdida en validación.



(a) FLAN-T5 (b) Falcon

Figura 3.1: Pérdidas en C_{train} y C_{val} durante ajuste de modelos.



(a) FLAN-T5 (b) Falcon

Figura 3.2: Decaimiento de tasa de aprendizaje durante ajuste de modelos.

Capítulo 4

Resultados

4.1. Exploración de clasificación de mensajes utilizando GPT 3.5

Debido a limitaciones de tiempo y del uso de la API de OpenAI, para esta revisión se seleccionó una muestra aleatoria de 2,008 mensajes. Para cada mensaje procesado de la muestra, se utilizó una llamada al método `create` de la clase `ChatCompletion` del paquete `openai` en su versión 0.27.4 [31]. Dentro de los parámetros del método, se especificó el uso del modelo `gpt-3.5-turbo`, el cual es el modelo utilizado por ChatGPT, y está optimizado para conversaciones. Para consultar la presencia de sesgo, se envió un mensaje con rol de usuario, y teniendo como contenido la interpolación de texto `f'{QUESTION}: "{m}"'`, donde "QUESTION" corresponde al texto "¿Puedes detectar sesgo de género en el siguiente texto?", y "m" corresponde a un mensaje procesado. Posteriormente, para cada respuesta obtenida desde la API, se realizó una clasificación semi automatizada con el fin de detectar si la respuesta entregada fue capaz de detectar o no sesgo de género. Para ello, primero se utilizó expresiones regulares para identificar las respuestas donde se indicó que no se presentó sesgo de género, y donde no fue posible de identificar presencia de sesgo de género. Las respuestas del primer caso fueron clasificadas como "sin sesgo de género", las segundas no se clasificaron, y el resto se clasificó como que "sí presenta sesgo de género".

De los 2,008 mensajes originales, se pudo detectar la presencia o ausencia de sesgo de género en 1,847 de ellos, entre los cuales había algunos mensajes duplicados, por lo que finalmente se pudo clasificar 1,816 mensajes únicos. Los mensajes donde no fue posible detectar presencia de sesgo de género constituyen en su mayoría textos vacíos (133 mensajes), nombres de archivos y direcciones web. En el caso donde sí se detecta sesgo de género, las respuestas del modelo suelen ser largas y variadas, explicando detalladamente la parte del texto entregado que es causante de la presencia de sesgo de género. Sin embargo, las respuestas no son consistentes. Por ejemplo, en algunos casos, se indica que el uso de "Estimados/as" es indicativo de que el texto no preenta sesgo de género, y en otros casos sí, argumentando muchas veces en este último caso que el sesgo se presenta porque

[31] https://pypi.org/project/openai/0.27.4/

solo se hace inclusión de los géneros masculino y femenino. Algo similar ocurre con el uso del símbolo "@". Ejemplos de esto se aprecia en la Tabla 4.1.

4.2. Inferencia de modelos ajustados sobre C_{test}

Para la inferencia, ambos modelos base fueron cargados en forma cuantizada, utilizando la misma configuración de `BitsAndBytes` utilizada durante el entrenamiento. Luego se cargó los respectivos tokenizadores, agregando los nuevos tokens correspondientes y ajustando el tamaño de la capa embedding del modelo base (solo para Seq2Seq). Para agregar el adaptador con los pesos ajustados previamente almacenados en HuggingFace, se utilizó el método `from_pretrained` de la clase `PeftModel`. Finalmente, se asignó el modelo a modo de evaluación con el método `eval`.

Para el caso de FLAN-T5, a cada d_{in} de C_{test} se le agregó el prefijo *"Eliminar sesgo de género del siguiente texto:"*, y en el caso de Falcon se agregó el prefijo *"<human>: ¿Puedes reescribir el siguiente texto sin sesgo de género?"* y el sufijo *"<assistant>"*. De esta forma, los documentos fueron tokenizados con el tokenizador correspondiente a cada modelo, y luego los tokens generados, junto con sus máscaras de atención, fueron entregados al método `generate` del modelo ajustado correspondiente, utilizando GPU y la siguiente configuración de generación:

- `num_return_sequences=1`
- `do_sample=False`: Indica que se debe utilizar argmax para seleccionar el token con la mayor probabilidad en cada paso.

Para el valor de `max_new_tokens`, se utilizó un factor de escalamiento sobre la cantidad de tokens de cada d_{in} de C_{test}, por lo que el valor de este parámetro varió para cada generación. Se utilizó un valor de 1.2 de este factor para ambos modelos. El proceso toma 26 minutos para el caso de FLAN-T5[32], y 42 minutos para el caso de Falcon.

4.2.1. Seq2SeqLM

Durante la exploración de los documentos generados, se observó que en ocasiones los nuevos tokens incorporados son asignados erróneamente. Adicionalmente, los espacios en blanco adicionales del texto original son removidos, lo que se aprecia en los ejemplos de la *Tabla 4.1*.

En la Figura 4.1, se muestra una captura de un ejemplo de uso de esta aplicación, junto con el texto generado por el modelo ajustado (respuesta de la API en la consola del navegador). Con cada palabra agregada al cuadro de texto, se envía una solicitud a la API. En la parte inferior de las imágenes, se muestra el resultado del último llamado. En (a), bajo el cuadro de texto, aparecen botones con las secuencias que se sugiere sean cambiadas, donde al pasar el *mouse* sobre el botón,

[32] En el caso de FLAN-T5, no se utiliza el tokenizador rápido, debido a que éste agrega espacios posteriores a los nuevos tokens agregados: https://github.com/huggingface/transformers/issues/19293

se muestra en un *tooltip* la sugerencia de cambio. Al hacer click en el botón “los alumnos nuevos” (b), se genera el cambio en el cuadro de texto. Al hacer esto, permanece solamente el botón para el texto “Estimados”.

(a) Sugerencia de cambio.

(b) Cambio aplicado.

Figura 4.1: Ejemplo de uso aplicación *demo* con modelo FLAN-T5 ajustado.

4.2.2. CausalLM

Al explorar las secuencias generadas, se observó que la generación de texto se repite hasta el largo de secuencia especificado, repitiendo solamente la fracción que incluye *“<assistant>”* y el texto generado posteriormente. Esto es un *issue* que ha sido reportado [33], del cual se ha propuesto algunas alternativas de solución. Se probó estas alternativas, pero no se logró solucionar el problema, por lo que se decidió utilizar la versión del modelo ajustado según el tutorial y recomendaciones originales, aplicando un post procesamiento para obtener el texto generado por el modelo una única vez.

A diferencia del caso de Seq2Seq, en este caso la generación sí conserva espacios en blanco adicionales. En la Tabla 4.1 se muestra algunos ejemplos de las salidas generadas para el modelo Falcon ajustado. En la Figura 4.2, se muestra el mismo caso explorado en la Figura 4.1, pero utilizando el modelo ajustado sobre Falcon. El texto generado por el modelo ajustado es similar al del caso de Seq2Seq.

[33] https://github.com/huggingface/transformers/issues/22794

(a) Sugerencia de cambio.

(b) Cambio aplicado.

Figura 4.2: Ejemplo de uso aplicación *demo* con modelo Falcon ajustado.

4.2.3. Métricas de desempeño para modelos ajustados

C_{test} cuenta originalmente con 853 documentos $\hat{d}_{\text{out}}$, pero para el cálculo de métricas se omite aquellos que tengan menos de 5 tokens, con el fin evitar que no se produzca ocurrencias de superposición de 4-gramas al calcular BLEU. De esta forma, finalmente se cuenta con 782 documentos. Los tokens se construyen, tanto para los documentos de salida del modelo como para los documentos de salida esperados, utilizando el método `word_tokenize` del paquete `nltk.tokenize`. En la *Tabla 4.2* se muestran las métricas obtenidas utilizando ambos modelos sobre todos los documentos de C_{test} (General), y también considerando los subconjuntos de documentos:

- *Con sesgo*: Considera solamente los 58 documentos que sí poseen sesgo de género.
- *Sin sesgo*: Considera solamente los 271 documentos que no poseen sesgo de género.
- *No sesgable*: Considera solamente los 453 documentos de cuyo contenido no es posible construir versiones sesgadas o insesgadas respecto del género.

Tabla 4.3: Métricas de desempeño en modelos ajustados

Modelo	Escenario	BLEU	ROUGE
FLAN-T5	General	0.96223	0.9938
FLAN-T5	Con sesgo	0.90088	0.96363
FLAN-T5	Sin sesgo	0.95554	0.99571
FLAN-T5	No sesgable	0.94511	0.99652
Falcon	General	0.98914	0.99516
Falcon	Con sesgo	0.94809	0.9701
Falcon	Sin sesgo	0.99355	0.99734
Falcon	No sesgable	0.99254	0.99706

Respecto a las diferencias BLEU($\hat{d}_{\text{out}}, d_{\text{in}}$ − BLEU($d_{\text{in}}, d_t extout$), solo se realizó la cuantificación de cada situación para las generaciones entregadas por el modelo ajustado de Falcon (*Tabla 4.3*). Esto debido a que las generaciones de FLAN-T5 poseen diferencias respecto de la entrada y la salida esperada en aquellos tokens ausentes del tokenizador original del modelo.

Tabla 4.4: Resultados de comparación BLEU utilizando modelo Falcon ajustado

Tipo d_{in}	BLEU($\hat{d}_{\text{out}}, d_{\text{in}}$) − BLEU($d_{\text{in}}, d_{\text{out}}$)		
	0	< 0	> 0
Con sesgo	27	9	22
Sin sesgo	253	18	0
No sesgable	440	13	0

El detalle de los casos que requerían de una revisión manual se encuentra disponible dentro de los archivos de datos del repositorio[34]. De ello, se destaca:

- 4 de los 9 casos donde la entrada presentaba sesgo de género y la generación del modelo fue menos similar a la salida esperada que la entrada original, corresponden a mitigaciones de sesgo de género correctas. En estos 4 casos, la similitud entre la generación y la salida esperada fue menor debido a que la mitigación generada no se encontraba dentro de los casos de salida esperada utilizados. En los otros 5 casos no se realiza mitigación de sesgo de género, destacando que solo en 1 de esos casos se realizó una mitigación de sesgo en una situación errónea (referida a objeto en lugar de persona).

- De los 31 casos en donde la generación del modelo fue diferente a la entrada, y se esperaba que fuesen iguales, la mayoría de las diferencias se da por cambios de ortografía o gramática entre la entrada y la generación del modelo (9 casos). Se presentan además 3 casos de una mitigación de sesgo de género incorrecta, y 4 casos de mitigación de sesgo de género innecesaria (se realiza cambio de una forma insesgada a otra forma insesgada). Estos últimos 4 casos se consideran correctos.

En la Figura 4.3, se muestra los porcentajes de generaciones correctas e incorrectas del modelo Falcon ajustado, sobre los tres tipos de d_{in} presentes en C_{test}. El caso de interés, que es el de generaciones correctas para los d_{in} con sesgo de género, no logra una mayoría de generaciones correctas. Considerando que las generaciones incorrectas corresponden en su mayoría a situaciones donde se preservó el mismo contenido del documento de entrada para la generación del documento de salida, se puede concluir que el modelo ajustado logró mitigar el sesgo de género en el 45 % de los documentos con sesgo de género.

[34] https://github.com/GianniCatBug/spanish-gender-debias/blob/main/data/processed/20231220_metrics_CAUSAL.xlsx

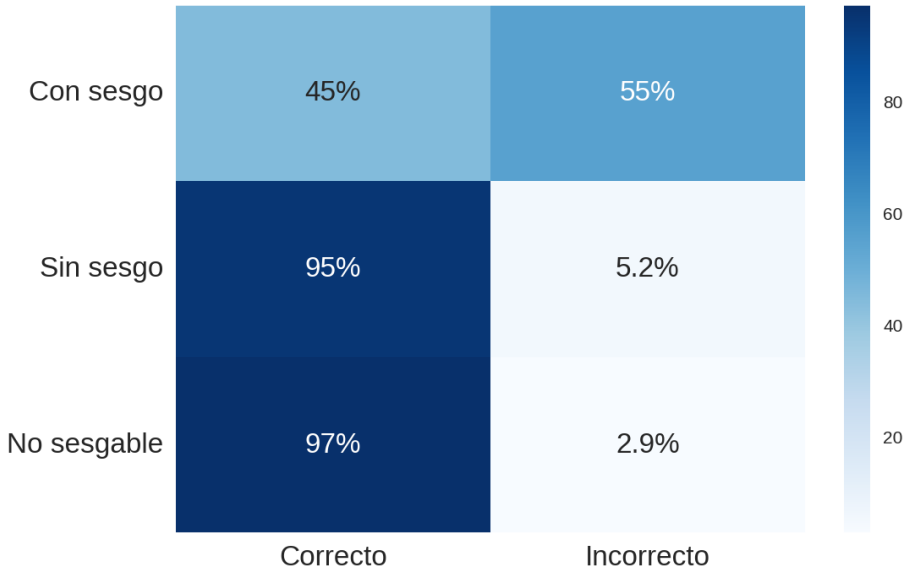


Figura 4.3: Porcentaje de generaciones correctas e incorrectas, en los distintos tipos de documentos disponibles en C_{test}

Tabla 4.1: Ejemplos de clasificaciones de textos entre con y sin sesgo de género realizadas por GPT 3.5 turbo

Mensaje	**Respuesta GPT**	**Clase**
"Estimadas/os estudiantes, Las/os invitamos a participar de la charla virtual sobre la beca Botín..."	No hay sesgo de género en el texto, ya que se utiliza lenguaje inclusivo al dirigirse a los estudiantes con la fórmula "Estimadas/os estudiantes" y al referirse a las/os participantes de la charla. Además, se utiliza el símbolo de la barra para incluir tanto el género masculino como femenino en algunas palabras.	No presenta sesgo de género
"Estimadas/os, El Programa de Pueblos Indígenas FCFM junto con la DDG, tenemos el agrado de invitarles al Taller de Aymara (Nivel 1)..."	Sí, se puede detectar un sesgo de género en el texto. La frase "Estimadas/os" utiliza la forma masculina y femenina para dirigirse al público, lo que indica un intento por ser inclusivo/a de ambos géneros. Sin embargo, en el párrafo no hay ninguna mención explícita de la inclusión de personas de otros géneros, lo que puede interpretarse como una exclusión implícita (...)	Sí presenta sesgo de género
"Estimad@s, Ya está abierta la segunda convocatoria del PME Beauchef para realizar un intercambio de un semestre en el extranjero. Invitamos a tod@s l@s interesad@s a una charla informativa..."	No se percibe sesgo de género en el texto, ya que se utiliza el signo "@" para incluir a ambos géneros en las palabras y frases escritas.	No presenta sesgo de género
"Estimad@s Estudiantes: (...). Si requieres enviarnos información o tienes conocimiento de algún compañer@ que requiera apoyo..."	El texto parece incluir un lenguaje inclusivo al utilizar el signo @ para referirse tanto a estudiantes masculinos como femeninos. Sin embargo, hay un posible sesgo de género al utilizar el término "compañer@" en lugar de un término neutro como "compañero/a" o "compañerx". Además, no se hace mención de posibles estudiantes no binarios o de género no conforme.	Sí presenta sesgo de género

Tabla 4.2: Ejemplos de fragmentos de generaciones de modelos ajustados

Entrada	"...¡No te lo pierdas!..."
FLAN-T5	"...¿No te lo pierdas!..."
Falcon	"...¡No te lo pierdas!..."
Entrada	"...para mañana jueves 12..."
FLAN-T5	"...para mañana jueves 12..."
Falcon	"...para mañana jueves 12..."
Entrada	"...En Tienda Beauchef estamos buscando monitores..."
FLAN-T5	"...En Tienda Beauchef estamos buscando monitores..."
Falcon	"...En Tienda Beauchef estamos buscando monitores/as..."
Entrada	"...oportunidad académica para nuestros estudiantes..."
FLAN-T5	"...oportunidad académica para nuestros/as estudiantes..."
Falcon	"...oportunidad académica para nuestros estudiantes..."
Entrada	"...investigadores franceses y chilenos..."
FLAN-T5	"...investigadores franceses y chilenos..."
Falcon	"...investigadores franceses y chilenos..."

Capítulo 5

Conclusiones

Los resultados obtenidos permiten concluir que el uso de PLMs ajustados es útil para tareas como la mitigación del sesgo de género en textos en español, utilizando un conjunto de datos paralelo de documentos con y sin sesgo de género para el ajuste. Si bien los resultados obtenidos no logran mitigar el sesgo de género en la mayoría de los datos evaluados, los resultados obtenidos son prometedores, e indican una dirección correcta de trabajo.

Se concluye además que sí es posible el uso de un PLM para mitigar el sesgo de género en texto en tiempo real, considerando que a mayor complejidad del modelo a utilizar, el tiempo de generación será mayor. Si se utiliza un modelo de mayor complejidad, una forma de poder reducir este tiempo es utilizando mayor cantidad de capacidad de GPU, lo que permitiría la carga del modelo en una cuantización menor (como 8 bits), permitiendo así inferencias más rápidas. Otra alternativa es el uso de un modelo menos complejo, del cual se podría lograr un buen rendimiento siguiendo las oportunidades de mejora previamente mencionadas.

En cuanto a la capacidad del modelo para mitigar el sesgo de género, existen oportunidades de mejora que podrían permitir mejores resultados, tales como:

- Uso de PLMs de mayor tamaño, o de LLM. Para ello se debe considerar que sería requerida mayor cantidad de recursos computacionales (mayor capacidad de GPU).
- Ajuste de hiperparámetros, tales como `dropout`, `r` o `lora_alpha`. Esto requeriría realizar múltiples ajustes registrando los valores de pérdida en el conjunto de validación al utilizar distintas combinaciones de estos hiperparámetros. La combinación con una menor pérdida en el conjunto de validación, sin sobrepasar el valor de pérdida en el conjunto de entrenamiento, sugeriría ser una combinación de hiperparámetros capaz de generalizar mejor la mitigación del sesgo de género frente a nuevos datos. La exploración de distintas combinaciones de estos hiperparámetros requeriría un mayor tiempo de entrenamiento.
- Mejorar la variabilidad de los datos para el ajuste del modelo, y/o ajustar el modelo utilizando secuencias más largas. Esto podría ayudar a generar mitigaciones de sesgo de género en situaciones en que el modelo actual no es capaz de hacerlo. Se debe considerar que el utilizar

secuencias más largas podría requerir de mayor cantidad de requerimientos computacionales (memoria GPU), y el aumentar la cantidad de datos para dar una mayor variabilidad requeriría además de un mayor tiempo de entrenamiento.

- Uso de *reinforcement learning* mediante retroalimentaciones de personas que indiquen generaciones correctas e incorrectas del modelo. Esto requeriría la implementación de una interfaz y uso de una base de datos, por lo que aumentaría la complejidad de la implementación.

- Ajuste de capa de *word-embeddings*, incorporando tokens de palabras que no se encuentren en el vocabulario del tokenizador original del modelo. Esto especialmente para modelos con un vocabulario menos extenso, como era el caso de T5. Esto requeriría de un mayor tiempo de ajuste del modelo.

En la exploración de datos realizada, se observó que existe tendencia a la disminución de la presencia de sesgo de género en las comunicaciones escritas publicadas en la sección "Novedades" del portal de U-Cursos de la FCFM. Esto es ciertamente positivo, y si bien la prioridad en pos de lograr una mejor inclusión y equidad de género debe ser el de educar a las personas en estas materias, el uso de modelos de aprendizaje automatizado (como los modelos previamente propuestos) deben buscar ser una herramienta de apoyo a las personas en lograr dicha inclusión y equidad.

Bibliografía

[1] Friedman, B. y Nissenbaum, H., "Bias in computer systems," ACM Transactions on information systems (TOIS), vol. 14, no. 3, pp. 330–347, 1996.

[2] Stanczak, K. y Augenstein, I., "A survey on gender bias in natural language processing," arXiv preprint arXiv:2112.14168, 2021.

[3] Sun, T., Gaut, A., Tang, S., Huang, Y., ElSherief, M., Zhao, J., Mirza, D., Belding, E., Chang, K.-W., y Wang, W. Y., "Mitigating gender bias in natural language processing: Literature review," arXiv preprint arXiv:1906.08976, 2019.

[4] Mars, M., "From word embeddings to pre-trained language models: A state-of-the-art walkthrough," Applied Sciences, vol. 12, no. 17, p. 8805, 2022.

[5] Qian, Y., Muaz, U., Zhang, B., y Hyun, J. W., "Reducing gender bias in word-level language models with a gender-equalizing loss function," arXiv preprint arXiv:1905.12801, 2019.

[6] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., y Kalai, A. T., "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," Advances in neural information processing systems, vol. 29, 2016.

[7] Zhao, J., Zhou, Y., Li, Z., Wang, W., y Chang, K.-W., "Learning gender-neutral word embeddings," arXiv preprint arXiv:1809.01496, 2018.

[8] Nadeem, M., Bethke, A., y Reddy, S., "Stereoset: Measuring stereotypical bias in pretrained language models," arXiv preprint arXiv:2004.09456, 2020.

[9] Stanovsky, G., Smith, N. A., y Zettlemoyer, L., "Evaluating gender bias in machine translation," arXiv preprint arXiv:1906.00591, 2019.

[10] Borchers, C., Gala, D. S., Gilburt, B., Oravkin, E., Bounsi, W., Asano, Y. M., y Kirk, H. R., "Looking for a handsome carpenter! debiasing gpt-3 job advertisements," arXiv preprint arXiv:2205.11374, 2022.

[11] Doughman, J. y Khreich, W., "Gender bias in text: Labeled datasets and lexicons," 2022.

[12] Fersini, E., Rosso, P., Anzovino, M., *et al.*, "Overview of the task on automatic misogyny identification at ibereval 2018.," Ibereval@ sepln, vol. 2150, pp. 214–228, 2018.

[13] Toshevska, M. y Gievska, S., "A review of text style transfer using deep learning," IEEE Transactions on Artificial Intelligence, 2021.

[14] Jin, D., Jin, Z., Hu, Z., Vechtomova, O., y Mihalcea, R., "Deep learning for text style transfer:

A survey," Computational Linguistics, vol. 48, no. 1, pp. 155–205, 2022.

[15] Ma, X., Sap, M., Rashkin, H., y Choi, Y., "Powertransformer: Unsupervised controllable revision for biased language correction," en EMNLP, 2020.

[16] Sutskever, I., Vinyals, O., y Le, Q. V., "Sequence to sequence learning with neural networks," Advances in neural information processing systems, vol. 27, 2014.

[17] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., y Bengio, Y., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[18] Hu, Z., Lee, R. K.-W., Aggarwal, C. C., y Zhang, A., "Text style transfer: A review and experimental evaluation," ACM SIGKDD Explorations Newsletter, vol. 24, no. 1, pp. 14–45, 2022.

[19] Hochreiter, S. y Schmidhuber, J., "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] Bahdanau, D., Cho, K., y Bengio, Y., "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, $\mathscr{L}$., y Polosukhin, I., "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[22] Webster, J. J. y Kit, C., "Tokenization as the initial phase in nlp," en COLING 1992 volume 4: The 14th international conference on computational linguistics, 1992.

[23] He, K., Zhang, X., Ren, S., y Sun, J., "Deep residual learning for image recognition," en Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[24] Ba, J. L., Kiros, J. R., y Hinton, G. E., "Layer normalization," arXiv preprint arXiv:1607.06450, 2016.

[25] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., *et al.*, "A survey of large language models," arXiv preprint arXiv:2303.18223, 2023.

[26] Zaib, M., Sheng, Q. Z., y Emma Zhang, W., "A short survey of pre-trained language models for conversational ai-a new age in nlp," en Proceedings of the Australasian computer science week multiconference, pp. 1–4, 2020.

[27] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.*, "Language models are few-shot learners," Advances in neural information processing systems, vol. 33, pp. 1877–1901, 2020.

[28] Parnami, A. y Lee, M., "Learning from few examples: A summary of approaches to few-shot learning," arXiv preprint arXiv:2203.04291, 2022.

[29] Mosbach, M., Pimentel, T., Ravfogel, S., Klakow, D., y Elazar, Y., "Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation," arXiv preprint arXiv:2305.16938,

2023.

[30] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., *et al.*, "Improving language understanding by generative pre-training," 2018.

[31] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., y Liu, P. J., "Exploring the limits of transfer learning with a unified text-to-text transformer," The Journal of Machine Learning Research, vol. 21, no. 1, pp. 5485–5551, 2020.

[32] Doughman, J., Khreich, W., El Gharib, M., Wiss, M., y Berjawi, Z., "Gender bias in text: Origin, taxonomy, and implications," en Proceedings of the 3rd Workshop on Gender Bias in Natural Language Processing, pp. 34–44, 2021.

[33] Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., y Launay, J., "The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only," arXiv preprint arXiv:2306.01116, 2023.

[34] Dao, T., Fu, D., Ermon, S., Rudra, A., y Ré, C., "Flashattention: Fast and memory-efficient exact attention with io-awareness," Advances in Neural Information Processing Systems, vol. 35, pp. 16344–16359, 2022.

[35] Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., y Liu, Y., "Roformer: Enhanced transformer with rotary position embedding," Neurocomputing, p. 127063, 2023.

[36] Shazeer, N., "Fast transformer decoding: One write-head is all you need," arXiv preprint arXiv:1911.02150, 2019.

[37] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., *et al.*, "Scaling instruction-finetuned language models," arXiv preprint arXiv:2210.11416, 2022.

[38] Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," arXiv preprint arXiv:2112.11446, 2021.

[39] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., y Chen, W., "Lora: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, 2021.

[40] Dettmers, T., Pagnoni, A., Holtzman, A., y Zettlemoyer, L., "Qlora: Efficient finetuning of quantized llms," arXiv preprint arXiv:2305.14314, 2023.

[41] Papineni, K., Roukos, S., Ward, T., y Zhu, W.-J., "Bleu: a method for automatic evaluation of machine translation," en Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pp. 311–318, 2002.

[42] Lin, C.-Y., "Rouge: A package for automatic evaluation of summaries," en Text summarization branches out, pp. 74–81, 2004.

[43] Kingma, D. P. y Ba, J., "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

Anexos

Anexo A. Análisis exploratorio de Corpus construido

A.1. Frases posibles de ser pareadas

Considerando las frases originales, a partir de las cuales se construye d_{in} y d_{out} para cada conjunto de datos, alrededor de un tercio son posibles de ser pareadas, es decir, que es posible construir a partir de ellas una versión con sesgo de género y otra sin sesgo de género. Esto se observa en la Figura A.1.

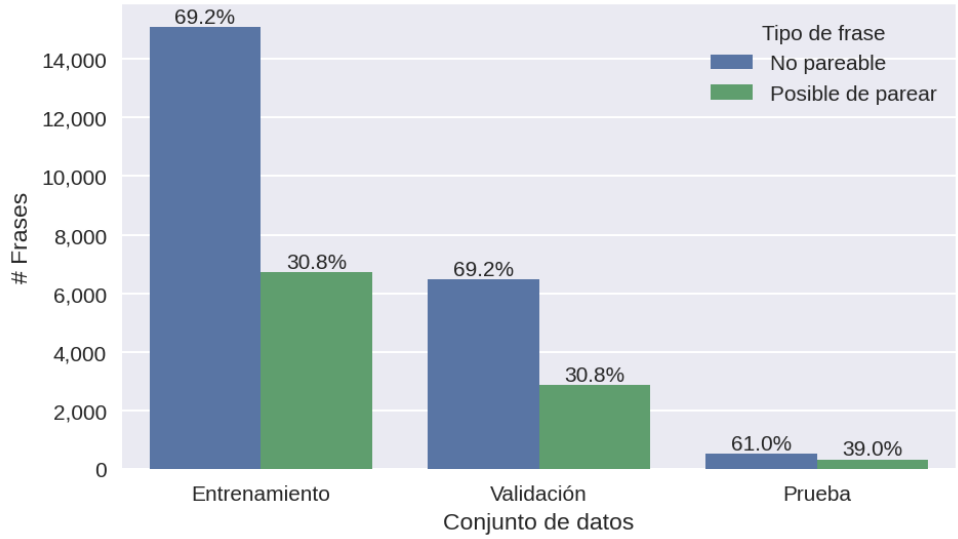


Figura A.1: Exploración de frecuencias de frases posibles y no posibles de ser pareadas utilizadas para construir los conjuntos de datos de entrenamiento, validación y prueba.

Las frases que no son posibles de ser pareadas son aquellas cuyo contenido no hace referencia al género, o que no alude a personas. También se excluyen frases en idiomas distintos al español. Ejemplos de estos casos son los siguientes:

- *"We are experiencing a polar wave in the capital"*
- *"Subdirección de Asuntos Estudiantiles"*
- *"Postulaciones abiertas entre el 5 y el 26 de junio"*

A.2. Cantidad de mensajes y frases por año

Considerando solamente las frases utilizadas para construir el conjunto de datos de entrenamiento y validación, C, se explora la cantidad de mensajes y frases por año. Considerando que cada mensaje corresponde a una publicación, en la Figura A.2 se observa que a partir del año 2008 hasta el 2010 existe un aumento en la frecuencia de publicaciones, manteniéndose luego en alrededor de 500 mensajes por año, superando luego nuevamente esta cantidad desde el año 2018. Se debe tener en cuenta que las cantidades de los años 2007 y 2023 no pueden ser consideradas dentro de la tendencia, ya que corresponden a años incompletos donde se tiene datos para solamente los últimos y primeros meses respectivamente. En cuanto a la cantidad de frases, destaca que para los años 2018 a 2022 se cuenta con más de 2,000 frases.

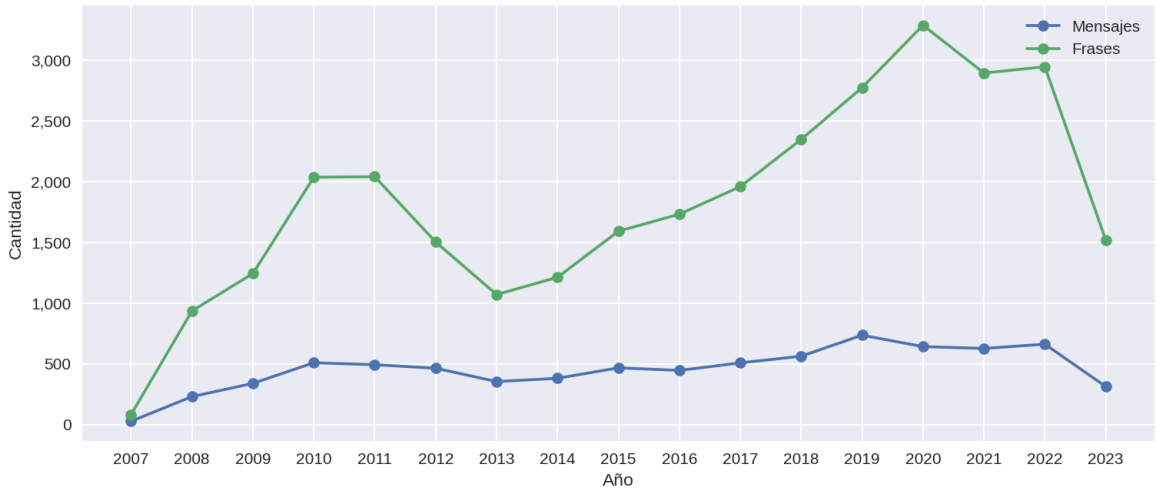


Figura A.2: Cantidad de mensajes publicados, y frases obtenidas a partir de ellos, en cada año, a partir de los datos utilizados para construir los conjuntos tanto para entrenamiento como validación.

A.3. Presencia de sesgo de género

Considerando solamente las frases que sí son posibles de ser pareadas, en la Figura A.3 se observa la frecuencia de frases con y sin sesgo de género utilizadas para construir los tres conjuntos de datos. Se observa que al considerar todos los datos utilizados para C, la distribución de los datos para construir el conjunto de prueba, C_{test}, es muy diferente, donde la gran mayoría de las frases posibles de ser pareadas no poseen sesgo de género. En cambio, para los datos utilizados para construir C se alcanza alrededor del 40 % para este caso. Sin embargo, se debe considerar que esta distribución tampoco es constante a lo largo de los años.

A modo de exploración en la escritura, en cuanto al sesgo de género, se explora la presencia o ausencia de sesgo de género en las frases posibles de ser pareadas de las publicaciones realizadas entre el 25 de octubre de 2007 y el 7 de agosto de 2023 (es decir, datos para construir C). En la Figura A.4, se observa que hasta el año 2015 más del 80 % de las frases, respecto de las frases

posibles de ser pareadas en cada año, presenta sesgo de género. Luego esta frecuencia mantiene en general una tendencia a seguir disminuyendo, pero de todas formas se presentan frecuencias mayores que para el caso de las frases utilizadas para construir C_{test}, cuyos datos van del 22 de agosto al 19 de octubre de 2023.

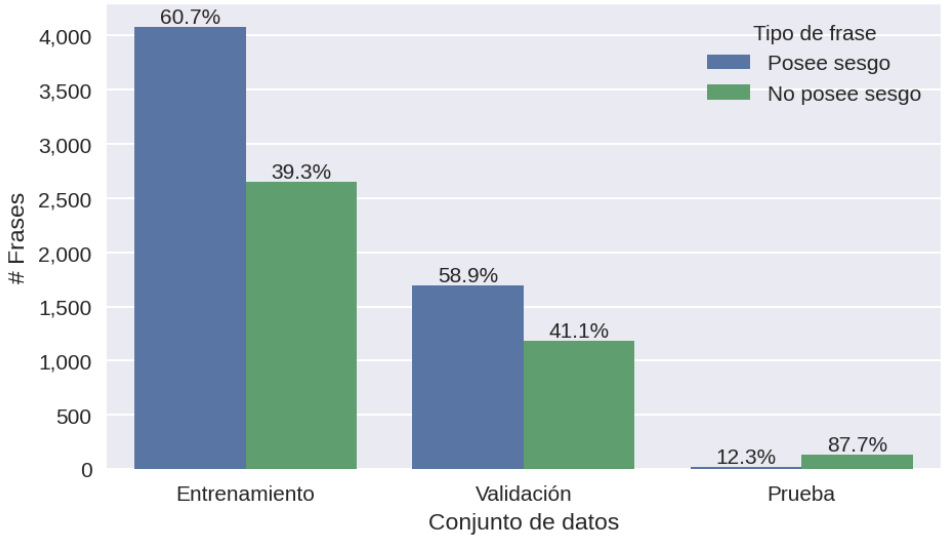


Figura A.3: Exploración de frecuencias de frases con y sin sesgo de género, dentro de las frases posibles de ser pareadas utilizadas para construir los conjuntos de entrenamiento, validación y prueba

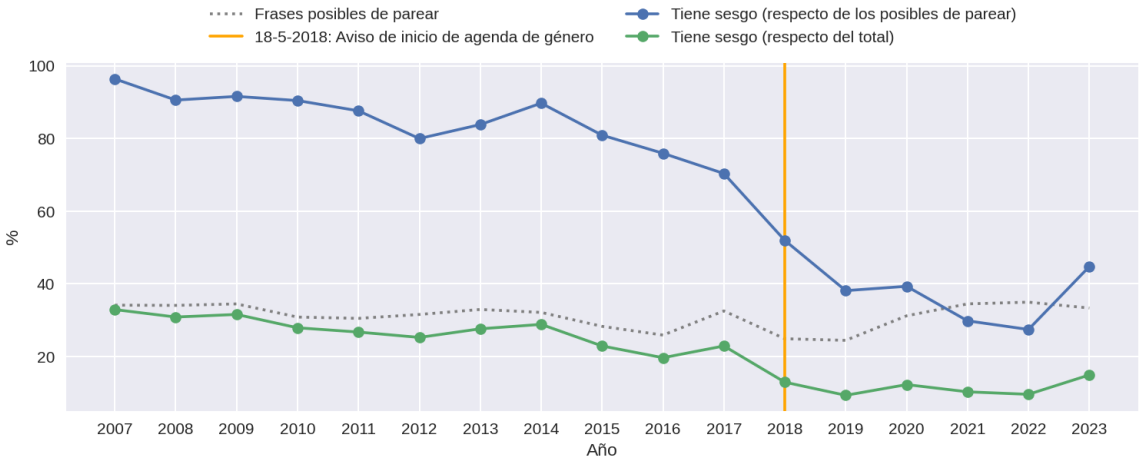


Figura A.4: Frecuencia de frases con sesgo de género en los datos de entrenamiento y validación, respecto del total de frases y de las frases posibles de ser pareadas. Se indica además, a modo de referencia, la frecuencia de frases posibles de parear respecto del total de frases, y el año donde se da inicio a la agenda de género en la FCFM.

A.4. Palabras más frecuentes

Considerando solamente las frases posibles de ser pareadas utilizadas para construir C, se explora las 40 palabras más frecuentes tanto para frases con sesgo de género (Figura A.5 (a)) como frases sin sesgo de género Figura A.5 (b). Para la construcción de las nubes de palabras se omite una lista de palabras comunes o poco informativas.

De las nubes de palabras en la Figura A.5, destaca para el caso de las frases sin sesgo de género el uso de recursos tipográficos tales como '\' y '@', así como una mayor frecuencia de la palabra 'comunidad' y el artículo 'les', con respecto a las frases con sesgo de género. En las frases con sesgo de género, destaca la gran presencia de la palabra 'alumnos', que no se encuentra entre las más frecuentes para las frases sin sesgo de género.

(a) Frases con sesgo de género

(b) Frases sin sesgo de género

Figura A.5: Palabras más frecuentes en frases posibles de parear utilizadas para construir C.