Universidad del Desarrollo
Facultad de Ingeniería

ESTE POLÍTICO NO EXISTE

Usando GPT-2 para la generación de tweets políticos

POR: ENRIQUE ALBERTO RODRÍGUEZ GARRIDO

Proyecto de grado presentado a la Facultad de Ingeniería de la Universidad del Desarrollo para optar al grado académico de Magíster en Data Science

PROFESOR GUÍA:

Dr. Leo Ferres

Diciembre 2020

SANTIAGO

Dedicado a las Elizabeths

TABLA DE CONTENIDO

Resumen

Los modelos de lenguaje natural han adquirido una importante relevancia desde 2018 cuando se incorpora de la arquitectura Transformer. Aunque existe una gran cantidad y muy diversos trabajos al respecto, muy poco podemos encontrar en español.

Este trabajo aborda una aplicación del modelo GPT-2, una implementación de arquitectura transformer en un modelo de lenguaje, para ejecutar tareas de generación de texto casual y texto controlado. En ambos casos hemos realizado el entrenamiento con las publicaciones más recientes de las autoridades políticas chilenas en la red social twitter.

Los resultados muestran que incluso un modelo GPT-2 base puede ser ajustado para cumplir una tarea específica como generar nuevos Tweets políticos reutilizando el aprendizaje del lenguage pre entrenado. Además entrenamos el modelo para que aprenda de los contextos de cada uno de los tweets y utilizar estos mismos contextos para solicitar la generación de acuerdo a parámetros de control.

Las aplicaciones de estos nuevos modelos de lenguaje con atención están en la vida personal como en el mundo empresarial. Usando estas innovaciones es posible agregar más valor a servicios actuales o crear nuevos servicios, habilitados con esta tecnología.

1. Introducción

1.1. Los modelos de lenguaje natural y su relevancia actual

Los modelos de generación de texto han tenido un importante avance en los últimos 3 años. Al punto de que actualmente han roto barreras que hace no mucho tiempo se pensaba que estaban en la ciencia ficción [1]. La teoría que está detrás de todo esto se remonta a varias décadas, pero las recientes avances en tecnología como también en la investigación han acelerado las innovaciones en el área.

Las nuevas arquitecturas de redes neuronales para modelamiento de secuencias incorporan innovaciones que permiten un entendimiento y procesamiento más sofisticado del lenguaje. Hace no más de 3 años, se publicó una novedosa arquitectura y metodología que permite entender las relaciones subyacentes de los textos de largas secuencias: El transformer [2]. Esta innovación propicia un salto en el estado del arte en esta materia, generando una carrera por la implementación de modelos cada vez más avanzados utilizando con esta arquitectura. Esta competencia está lejos de detenerse, sin conocer aún los límites a los que se podría alcanzar [3].

La relevancia en el estudio también se despdrende de la cantidad de artículos publicados en los últimos años que contienen “NLP” (Natural Language Processing), “NLU” (Natural Language Understanding), “NLG” (Natural Language Generation). Como vemos en la Figura 1, la cantidad de publicaciones de ha duplicado año contra año desde 2017.

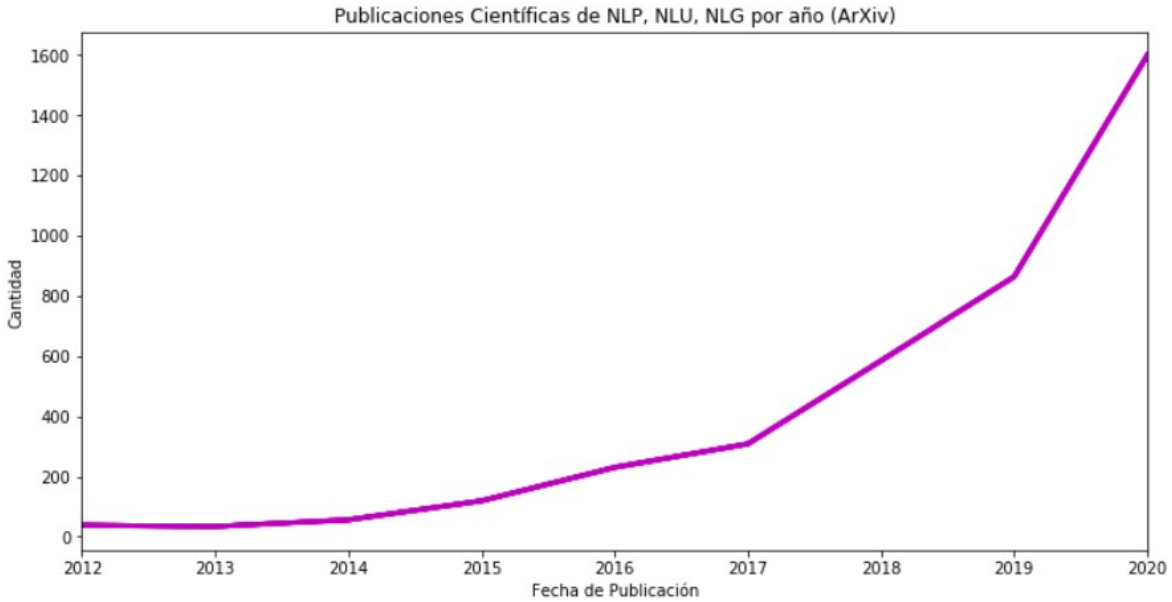


Figura 1 Cantidad de publicaciones NLP, NLG, NLU en ArXiv

El objetivo de un modelo de generación de texto es predecir con alta probabilidad un nuevo elemento en la secuencia del texto, ya sea el siguiente elemento o uno faltante en la secuencia. Al incorporar la arquitectura transformers se habilita el mecanismo de atención al resto de la secuencia; es decir, la predicción del nuevo elemento depende ahora del contexto y no sólo es una representación estadística. Gracias a esto, el modelo es capaz de aprender cuándo retener informaciones de la secuencia tales como género, ubicación, nombres, sintaxis, ideas centrales, conjugaciones, entre otras. El resultado: el texto sintetizado se incorpora de mejor manera en el contexto, con sintaxis e ideas que lo recorren de principio a fin, entregando una naturalidad a la generación del texto.

El valor de modelos de lenguaje aumenta cuando se utiliza para tareas específicas, y es donde se aprecia el beneficio más concreto de estas implementaciones. Basta un ajuste fino sobre un modelo pre-entrenado para utilizar todo el aprendizaje del lenguaje base y entrenarlo los datos específicos asociados a la tarea.

Las aplicaciones de los modelos de lenguaje están hoy en nuestra vida personal y en la empresa. Los asistentes virtuales [5], buscadores [6], traductores de idiomas [7] [8] son ejemplos de ello por mencionar algunos donde ya han incorporado mecanismos de atención o arquitecturas transformers a sus modelos (o están en camino a ello). Incluso se ha aplicado esta innovación en modelos de composición musical [9].

1.2. Modelos de lenguaje en español y tareas específicas

A pesar de la gran popularidad y avance de estos modelos, a la fecha es difícil encontrar implementaciones aplicadas a lenguaje en español, a pesar de que es el lenguaje más hablado en el mundo después de Chino Mandarín [10]. Sólo encontramos un trabajo para español que realiza una implementación y evaluación de BERT [11], pero aún no hay aplicaciones para tareas específicas en español utilizando Transformers. Inclusive dentro de las comunidades abiertas de colaboración de modelos transformers más populares el español no aparece [12].

Esta ausencia genera una brecha en las aplicaciones en español normalmente hace que las innovaciones sean incorporadas en aplicaciones en inglés y tiempo después en español (también influyen otras variables como demanda, momento y mercado). A pesar de que hay diferencias en el lenguaje, las implementaciones no revistan una complejidad mayor de un idioma a otro.

En este trabajo implementamos un modelo de generación de texto casual GPT-2 en español y, como una tarea específica, le hemos enseñado a generar tweets como una autoridad

política chilena. En cuanto a la generación de texto, le instruimos de cuál tema hablar y con qué sentimiento, poniendo en práctica una técnica para controlar el contenido de esta síntesis.

2. Revisión Bibliográfica y Trabajo Relacionado

2.1. Breve historia de los Modelos de Lenguaje

Bag of words y n-grams

La bolsa de palabras (bag of words) es un modelo para representar un texto numéricamente, y esta forma almacenarlo como un vector para utilizarlo en tareas de aprendizaje automático. En pocas palabras, *bag of words* consiste en la creación de un vocabulario a partir de los textos, para luego utilizar este vocabulario en una transformación de los textos a vectores que representan el índice de cada una de las palabras en el vocabulario.

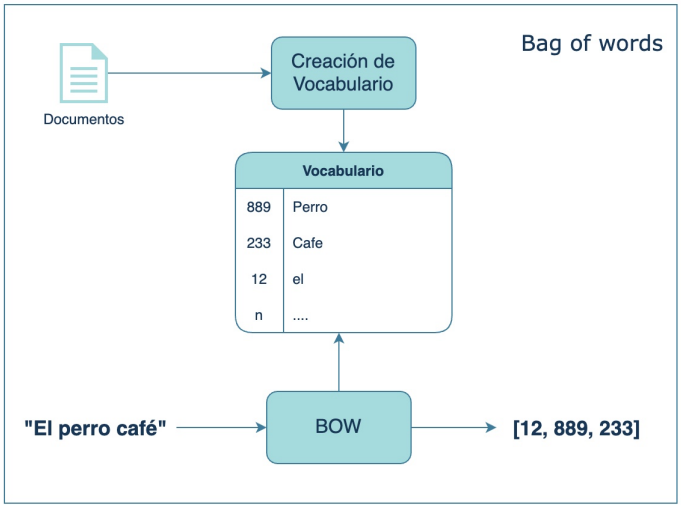


Figura 2 Modelo bag of words

Esta representación podría no ser suficiente para algunas tareas. Bajo esta modalidad frases como *"el perro café"* y *"el céfe perro*" tienen el mismo resultado como vector final (presencia o ausencia de la palabra en la frase). Para representar el texto de una forma más específica y detallada se utiliza el modelo *n-gram*, donde se consideran las combinaciones de n-palabras consecutivas que aparecen en el texto.

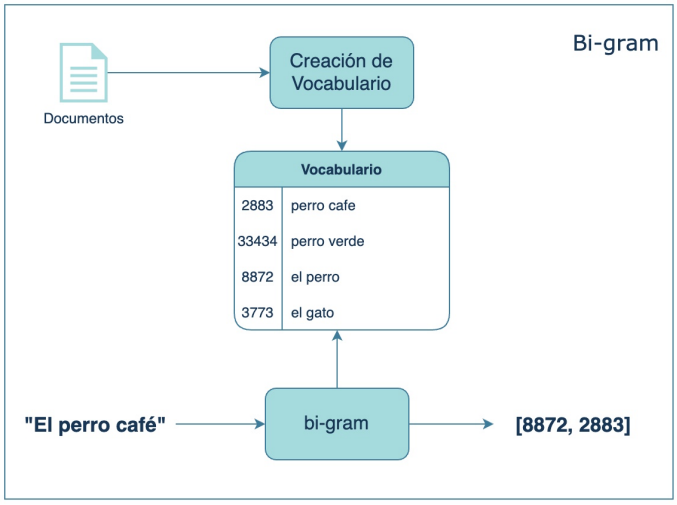


Figura 3 Modelo bi-gram

De esta forma "*el perro café"* y "e*l café perro*" no son representados de la misma forma en el vector resultante.

Una representación de pares de palabras se obtiene utilizando bi-gram, los tríos de palabras se representan en tri-grams. En general se habla de n-gram para abarcar los modelos de representación de texto que codifican ventanas de n elementos consecutivos. Donde bag of words no es más que el caso particular de n-gram con n=1.

La utilización de estos modelos tiene un uso importante en la clasificación de documentos (detección de spam, por ejemplo) y es la forma más básica de entendimiento del lenguaje.

Redes Neuronales Recurrentes

Hasta finales de la década de los 90, la aproximación para entender y realizar tareas de lenguaje era utilizar redes neuronales recurrentes (RNN). El funcionamiento, consiste en alimentar la red con el elemento actual de la secuencia y obtener como salida las probabilidades del siguiente elemento (véase **¡Error! No se encuentra el origen de la referencia.**). Adicionalmente a esto, se le informa a la red el estado oculto del elemento anterior. De esta forma, la RNN accede a un contexto previo reciente de la secuencia que utiliza para establecer una mejor predicción.

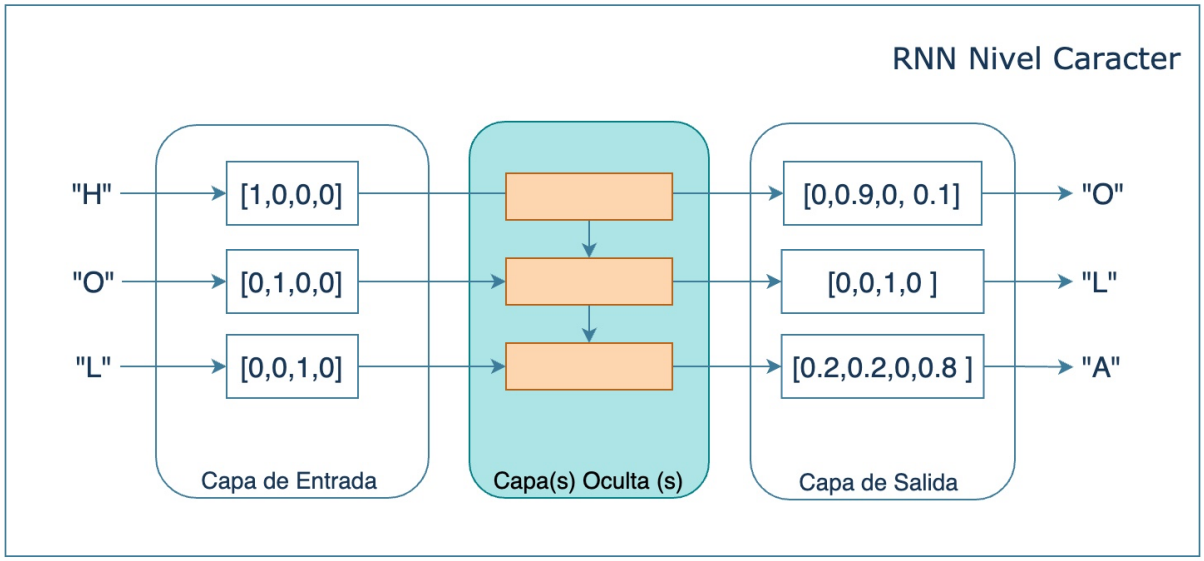


Figura 4 Ejemplo simplificado de una RNN a nivel de carácter.

En la Figura 4, en el primer paso a partir del vector [1,0,0,0] (Letra H) la red predice que la siguiente letra debe ser "O" que se utiliza como input en el siguiente paso, que también recibe actualización del estado

Dado que estas redes recurrentes utilizan estados anteriores como información, su entrenamiento se dificulta en con secuencias largas. El algoritmo utilizado en el

entrenamiento para llegar al óptimo (Back-Propagation Thought Time, BPTT) requiere que las actualizaciones de pesos se propaguen en todos los pasos hacia atrás. Siendo impracticable para secuencias largas como oraciones largas o frases [13]. Un ejemplo de esta aplicación sería el autocompletar palabra que nos sugiere el editor de texto o el smartphone a medida que vamos escribiendo.

Long Short Term Memory RNNs

En el año 1997 Hochreiter y Schmidhuber [13] plantean una arquitectura diferente para abordar tareas de NLP de secuencias más largas. Esta nueva RNN llamada LSTM (de *"Long Short Term Memory"*) cuenta con una arquitectura que permite mantener o descartar información de la secuencia. LSTM-RNN también trata a la secuencia de forma recurrente, pero ahora se incorpora el estado de la celda, que se transmite de un paso al siguiente. En este estado se agrega o remueve información antes de pasar a la siguiente iteración.

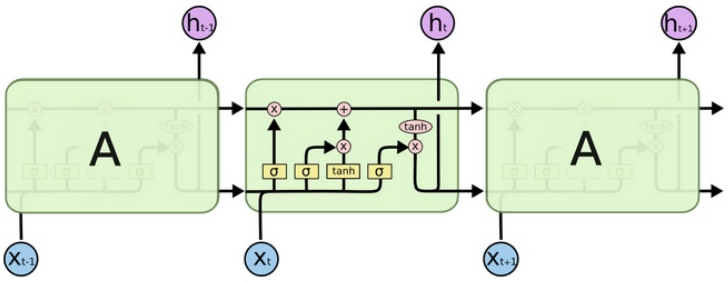


Figura 5 Funcionamiento LSTM RNN

De la Figura 5, en la línea inferior se recibe el input actual y el estado oculto anterior. Esta información se utiliza para actualizar el estado de la celda que va por la línea superior. Foto de[14]

Esta arquitectura permite que la información relevante se mantenga hacia los siguientes pasos y la información no relevante se descarte. Estos modelos pueden trabajar con secuencias más largas que las RNN.

Si bien es cierto la capacidad de las LSTM-RNN para recordar elementos más lejanos en la secuencia respecto a RNN (de ahí el *"Long" Short Term Memory*) es un importante progreso, no fue sino hasta finalizado el denominado Invierno de la IA [15] cuando se comenzó a aplicar a nuevas tareas de mayor complejidad utilizando (impulsado también por los avances en tecnologías de cómputo y storage).

Algunas aplicaciones utilizando esta arquitectura son Neural Machine Translation (NMT), Named Entity Recognition (NER), Asistentes Virtuales, Análisis de Sentimiento, subtitulado de imágenes y videos por mencionar algunas. No obstante sus virtudes, también puede sufrir el problema de las RNN, pero a secuencias más largas, es decir una inestabilidad (gradientes que explotan) o incapacidad para llegar al óptimo (gradientes a cero) en el proceso del entrenamiento.

Otro aspecto importante por mencionar es que, dado que LSTM olvidan información no relevante de la secuencia, en conjunto con la característica recurrente de la red, para cada tarea específica diferente se debe realizar un nuevo entrenamiento desde cero. En otras palabras, transfer-learning se hace impracticable (o sólo en ciertos casos específicos muy acotados [16])

Word2Vec y Embedding

Word2Vec es presentada el año 2013 por Mokolov et al [17] como una nueva técnica de procesamiento de lenguaje natural. El algoritmo permite representar cada palabra en un vector de *n* dimensiones, de manera tal que la relación de vectores refleja la relación de palabras. Veamos el siguiente ejemplo de la Figura 6, un modelo entrenado word2vec conoce las representaciones vectoriales de las palabras como resultado del entrenamiento de un gran corpus de texto (luego explicaremos cómo conoce esas representaciones). En este ejemplo el modelo funciona entregando esta representación en un espacio de 4 dimensiones numéricas. Esta representación de espacio de las palabras (tamaño del diccionario de las palabras) a vectores de numeros reales de menor tamaño es conocido como *embedding*.

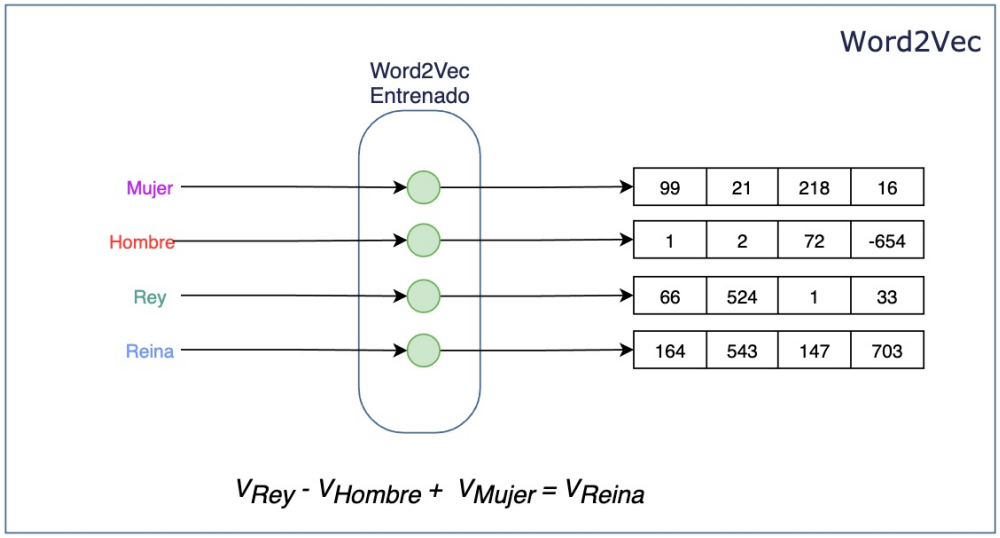


Figura 6 Funcionamiento de un modelo entrenado Word2Vec

En la Figura 6 hay 4 ejemplos de los vectores asociados a las palabras Mujer, Hombre, Rey y Reina. Debido que los resultantes son vectores que guardan la relación entre las palabras, es posible realizar operaciones matemáticas con ellos. Un ejemplo es restar al Vector de *Rey* el vector Hombre y sumando el vector *Mujer* se obtiene el vector muy similar al de *Reina*. También se pueden aplicar reglas de distancias para obtener palabras similares

o funciones trigonométricas para extraer distintas relaciones entre los vectores (ortogonalidad, paralelismo)

¿Como word2vec conoce estas representaciones vectoriales o embeddings? El principio lo vemos en la Figura 7, consiste en entrenar una red neuronal para una tarea específica, como lo es predecir una palabra en base a un input que está representando la palabra en un vector codificado one-hot. El tamaño del vector de entrada es el tamaño del vocabulario, donde existe un 1 en la posición de la palabra y el resto son 0s.

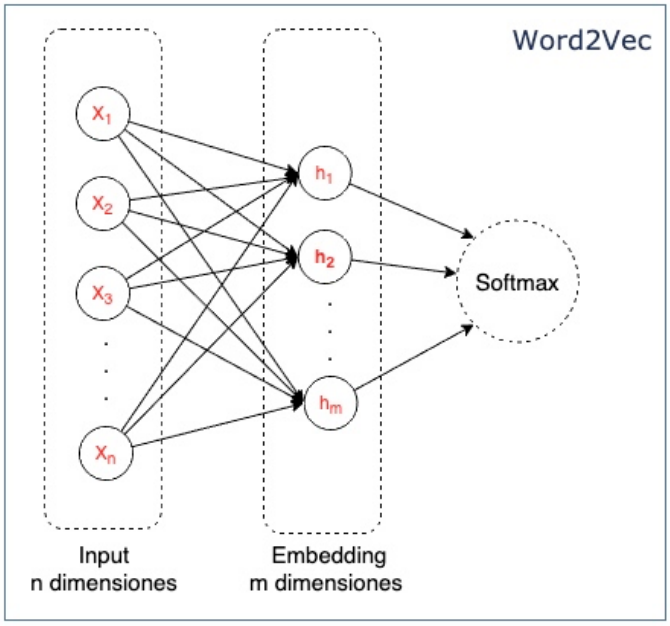


Figura 7 Obtención de las representación vectoriales en el Word2Vec.

Una vez entrenado el modelo, se utilizan los pesos de la capa oculta como representación vectorial de las palabras. La dimensión del embedding entonces es la cantidad de neuronas de esta capa, que es mucho menor que el vector de entrada.

Redes con Atención

Para explicar las redes con Atención primero veamos cómo funcionaba un modelo Neural Machine Translation (una aplicación de modelos secuencia a secuencia). NMT Es un modelo que consta de un Encoder y un Decoder, ambos componentes son una implementación de LSTM-RNN.

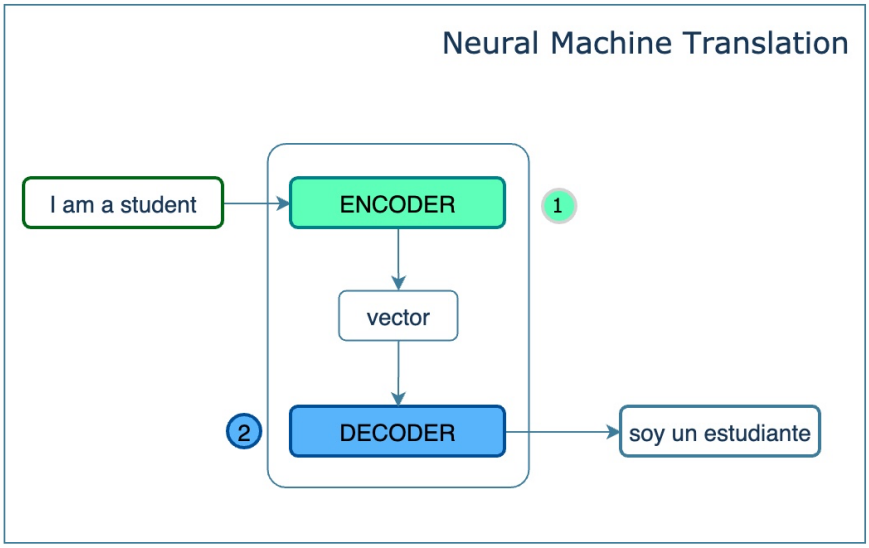


Figura 8 Funcionamiento de un modelo entrenado NMT.

En la Figura 8 se aprecia que el encoder procesa cada uno de los elementos de la secuencia de entrada y entrega una versión vectorizada del texto original al decoder. El decoder entiende estos vectores y su representación en el mundo del segundo lenguaje, logrando la traducción.

En modelos tradicionales NMT el input hacia el decoder no es más que la representación vectorial de la secuencia de entrada. Ambos decoder y encoder conocen la relación entre los propios lenguajes y este espacio común de vectores. Los vectores incorporan el contexto de la secuencia utilizando LSTM-RNN.

Ya en 2015 [18], cuando se incorpora el concepto de "*atención*", los modelos mejoran su desempeño considerablemente. La razón es que es posible pasar al decoder tanto el vector representativo de la secuencia, como también un vector de atención. Este vector no es mas que los estados ocultos de todos los pasos del encoder.

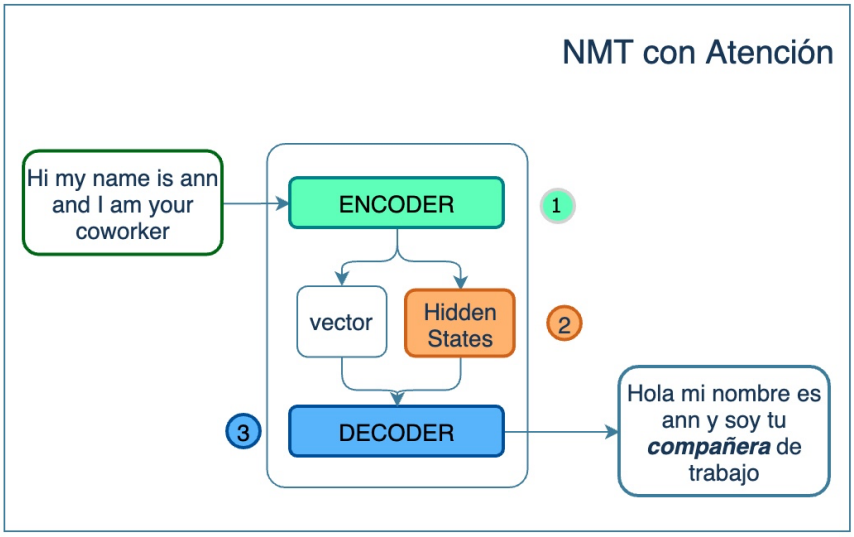


Figura 9 NMT con Atención

En el ejemplo de la Figura 9 se ve que cuando se agregan los estados ocultos del encoder es posible acceder al contexto para cada elemento de la secuencia. El decoder toma este contexto y calcula los vectores de atención que utiliza en conjunto con el vector de entrada para predecir el siguiente elemento.

Veamos un ejemplo concreto. En la Figura 10, para una traducción de "*are you still at home*?" al español, un modelo NMT codifica desde el inglés y descodifica al español. En la siguiente imagen la secuencia de entrada está el eje y. Para cada palabra traducida (eje x) se muestran los niveles de atención para los elementos de la secuencia de entrada, mientras más claro, más atención.

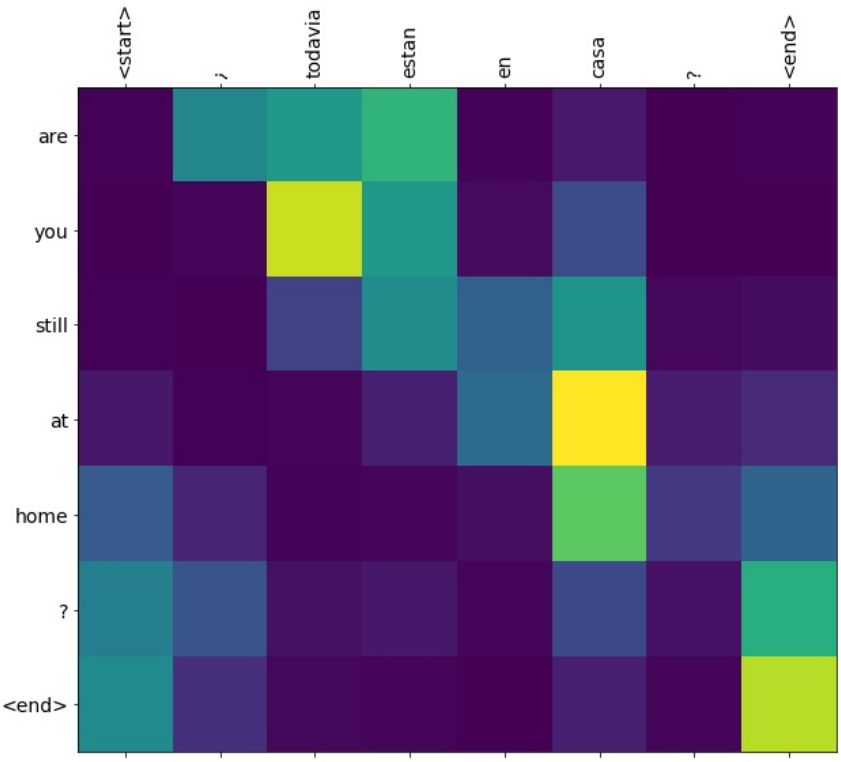


Figura 10 Vectores de atención en NMT (https://www.tensorflow.org/tutorials/text/nmt_with_attention)

De la Figura 10, para la traducción de la secuencia de entrada. Por ejemplo, la primera palabra se omite dado que parte con un verbo, la segunda se concentra en los tres primeros elementos para definir su significado

Transformers

En el año 2017 Vaswani et al proponen la arquitectura Transformer [2] para abordar de mejor forma las tareas de procesamiento de lenguaje. Se incorpora la atención ya utilizada en NMT pero ahora en la forma de mecanismo de atención múltiples (llamados cabezales de

atención múltiples), una importante innovación que permite poner atención en distintas formas dentro de la secuencia.

Atención Multi Cabezal

La atención multicabezal implica utilizar el mecanismo de atención varias veces dentro de la red. Esto se logra inicializando y entrenando varios vectores de atención para cada elemento de la secuencia. De esta forma cada atención puede aportar un contexto diferente para cada palabra.

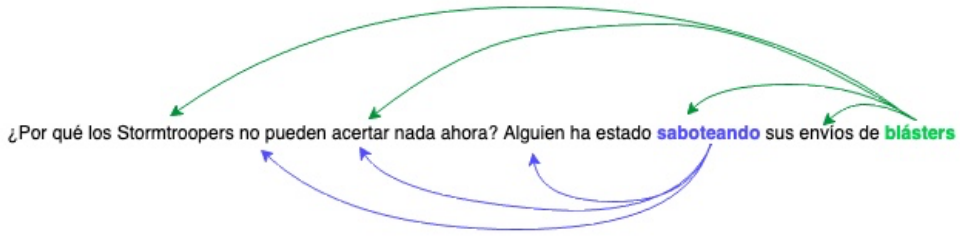


Figura 11 Atención múltiple como ejemplo.

En el ejemplo anterior, vemos que para la palabra "**saboteando**" se consideran también las palabras "**Stormtroopers**", "**acertar**" y "**alguien**" como atención . Algo similiar ocurre para la palabra "**blásters**" pero con otros elementos de la secuencia. Aunque en este ejemplo se visualizan dos palabras, la atención multicabezal se calcula para todos los elementos de la secuencia.

Cálculo de los vectores de Atención

En detalle, los vectores de atención se calculan primero obteniendo, para cada elemento de la secuencia, los siguientes vectores:

1. El vector Query (q) multiplicando el vector de entrada (token) por W^Q
2. El vector Key (k) multiplicando el vector de entrada por W^K
3. Finalmente, el vector Values (v) multiplicando el vector de entrada por W^V

Las matrices W^Q, W^K y W^V son aprendidas durante el entrenamiento del modelo.

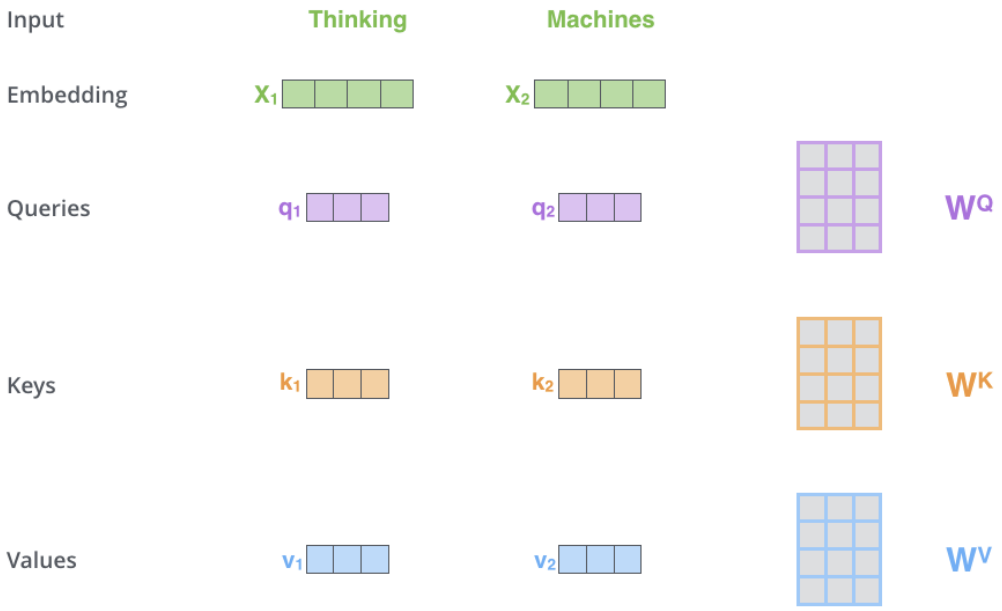


Figura 12 Paso 1 en el cálculo de los vectores de atención: Cálculo de vectores query, key y value. De [19]

A continuación, para cada elemento se itera en la secuencia calculando el score de cada token para el elemento (query vector). Este score, luego de ser normalizado y transformado con una función softmax se multiplica por el vector de value. Los vectores resultantes se suman para conformar el vector de auto-atención. Recordemos que estos pasos se realizan para cada posición en la secuencia. Entonces hay un vector para cada posición.

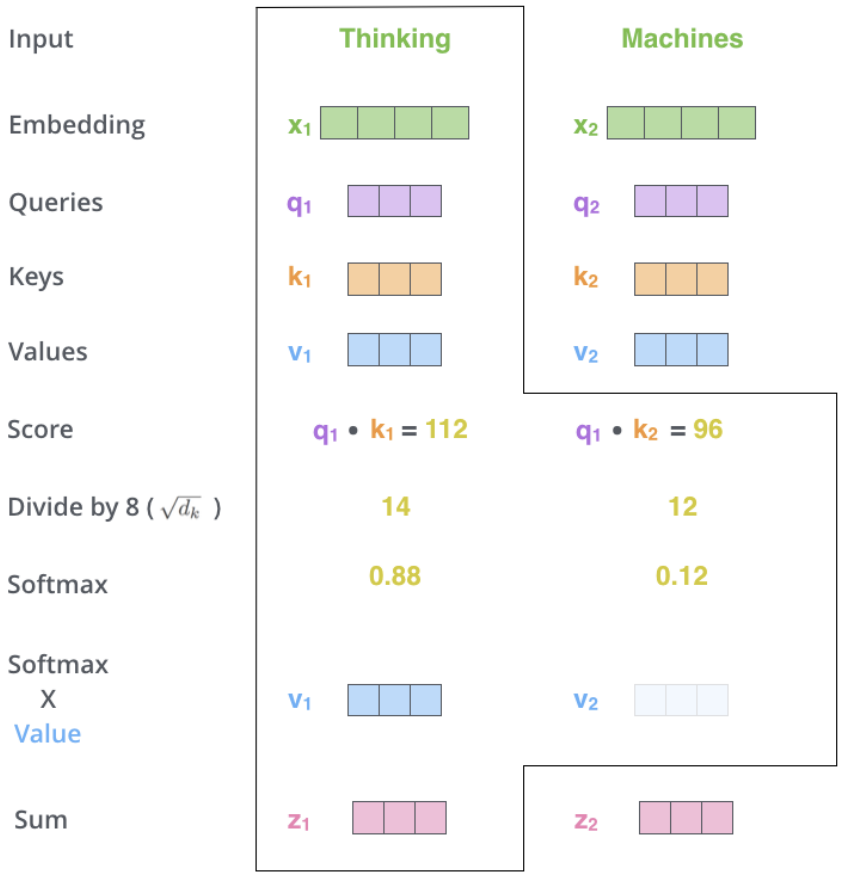


Figura 13 Paso 2 en el cálculo de los vectores de atención. De [19]

La atención multi-cabezal consiste en realizar este proceso varias veces, y en cada caso el set de matrices W^Q, W^K y W^V es inicializada de forma aleatoria y entrenada en conjunto con las demás. En la unidad Multi-Cabezal se combinan las atenciones.

Esta innovación permite tener distintas combinaciones de atención para cada posición. Una forma de visualizar esta idea es pensar en las formas en que se puede mirar una secuencia de texto: gramática, conjugaciones, sintaxis, etc.

Codificando la posición en la secuencia

Otro aspecto de innovación en esta arquitectura es cómo codifica la posición del token en la secuencia. Pensemos en el siguiente ejemplo:

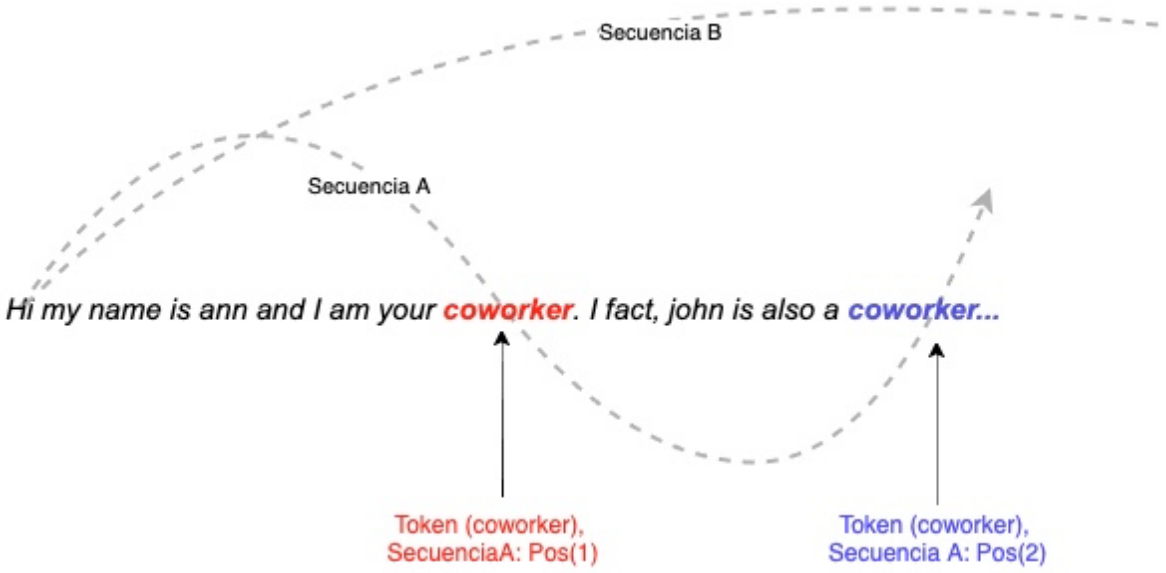


Figura 14 Introducción de información de las posiciones dentro la secuencia para los elementos.

La segunda vez que aparece **coworker** es bajo un contexto diferente a la primera vez. El mecanismo que utiliza transformer para entregar esta información (ubicación del token dentro de la secuencia) son funciones sinusoidales a distintas frecuencias. De esta manera el modelo puede incorporar el orden de la secuencia, utilizando diferentes ondas que pueden captar relaciones distantes de todo el documento.

Arquitectura Transformer

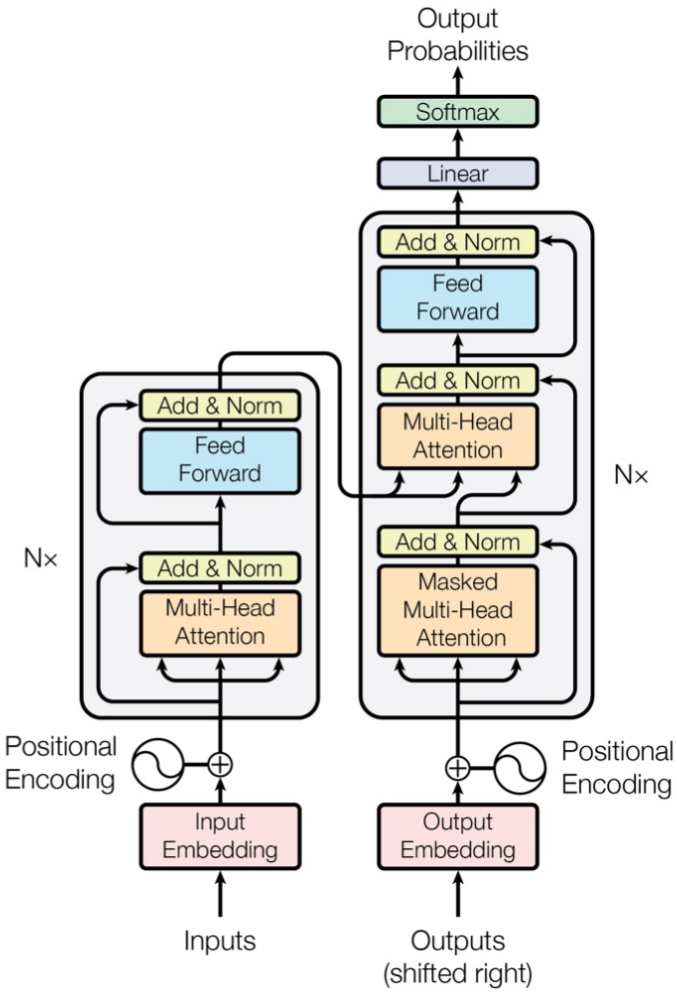


Figura 15 Arquitectura completa del Transformer[2]

Habiendo revisado los conceptos principales podemos abordar con mayor detalle la arquitectura en la Figura 15: consta de un encoder y un decoder. El encoder se alimenta de la vectorización de la secuencia de entrada (utilizando un tokenizador) y la posición codificada. Al interior se calculan los cabezales de atención con esta información se alimenta a una red neuronal totalmente conectada *feed fordward.* La salida de esta red se pasa como entrada a

la capa de atención del decoder. La estructura Atención Multi-Cabezal-FFNN se repite 6 veces.

En el caso del decoder la arquitectura es la misma que la del encoder, pero adicionalmente cuenta con una capa que realiza la atención multi cabezal sobre la salida del encoder.

El avance de transformers

Esta nueva arquitectura y sus principales innovaciones (la atención multi cabezal y la codificación de posiciones) permiten que los cálculos pueden realizarse en forma paralela y no en secuencia como se hace en LSTM-RNN. Esto cobra especial relevancia cuando es posible utilizar GPU para realizar cálculos matriciales complejos de forma muy eficiente.

Otro aspecto por destacar es la capacidad de reutilización de modelos transformers pre-entrenados. A diferencia de LSTM-RNN, el ajuste fino de modelos vía transfer learning es posible.

2.2. Estado del arte de Modelos Transformers

La publicación del artúculo que introduce Transformers en diciembre 2017 desbloquea de inmediato el entrenamiento de modelos con esta arquitectura para distintas tareas. De hecho, comienza una competencia por establecer los límites de estos modelos ampliando los tamaños de set de datos, parámentros y capacidades de los modelos año contra año.

Google BERT

El año 2019 Devlin et al [20] con gente de Google elaboran un modelo llamado *Bidirectional Encoder Representations from Transformers (BERT)* para modelos de lenguaje enmascarado (MLM)**.** Es decir, predecir la palabra faltante en una secuencia.

BERT aprende las representaciones bidireccionales dentro de un corpus de texto de 3.300 millones de palabras. Generando un modelo de 340 millones de parámetros que puede ser ajustado posteriormente con data específica para tareas especiales.

En el mismo 2019, un derivado de BERT llamado *RoBERTa* [21], *A Robust Optimized BERT Pretraining Approach*, se basa en BERT modificando ciertos hiperparámetros clave. Planteando que el modelo BERT original puede ser optimizado para mejores resultados.

Ya en 2020 *DestilBERT* [22] se define como una versión de BERT más pequeña, más rápida, más barata y liviana. Proponen un modelo de NLP de propósito general que luego puede ser ajustado para tareas más específicas que tengan limitaciones de cómputo y presupuesto. Este modelo tiene un tamaño 40% de su versión original y es 60% más rápido que BERT, pero mantiene un 97% de las capacidades de entendimiento de lenguaje.

Cabe mencionar *CamemBERT* como una implementación de BERT en francés utilizando para el entrenamiento un corpus francés. Es un ejemplo de implementaciones de este modelo en lenguaje distinto al inglés.

OpenAI: GPT

Un poco antes que Google, en junio de 2018, Radford et al [23] confeccionan y entrenan un modelo agnóstico a una tarea específica: GPT que son las siglas de *Generative Pre-Training.* Este modelo fue entrenado sobre corpus de más de 7000 libros de distintos

géneros; esto le ha permitido entender las dependencias dentro del lenguaje, para posteriormente realizar un ajuste fino y transferir ese aprendizaje para resolver una tarea específica.

GPT-2

Al año siguiente, el mismo autor publica un artículo con los resultados de GPT-2 [24] el cual es un es un escalamiento directo de 10 veces de GPT. Entrenado con más de 40 GB de datos y generando un modelo de más de 1500 millones de parámetros en su tamaño más grande. Por la diversidad de los datos el objetivo de GPT-2 se convierte en predecir la siguiente palabra dado una secuencia de palabras anteriores. Pero también puede asumir tareas diversas en otros dominios, por ejemplo pregunta-respuesta, resumen de texto y generación de texto casual.

GPT-3

OpenAI GPT-3 [3] el 2020 muestra nuevamente un salto de escala con respecto a GPT-2 tanto en tamaño de set de datos para entrenamiento de 500.000 millones de palabras, como en la cantidad de parámetros: 175.000 millones, en su modelo más grande.

Los resultados de GPT-3 indican que el modelo tiene un desempeño que supera el estado del arte en varias tareas [3]. Interesante de mencionar, en aritmética básica, llega a una exactitud de 100% para la suma de números de 2 dígitos y más de 80% para la suma de 3 dígitos, sólo mostrando al modelo algunos ejemplos de la tarea. El artículo original plantea la pregunta si este modelo es capaz de reconocer la tarea “suma” o la aprende con los ejemplos al momento de la prueba.

GPT-3 ha logrado comprensión del lenguaje tal que es capaz de elaborar artículos completos en base a una pequeña introducción. Estas creaciones por lo general son indistinguibles de los artículos escritos por humanos, pero la gran diferencia es que con un modelo así se puede crear cientos de artículos diferentes por segundo. Esto ha generado cuestionamiento acerca del mal uso que se le puede dar en la industria de las noticias falsas.

Este modelo aún no ha sido liberado para el uso público. Sólo está disponible vía API y su uso es restringido bajo solicitud directa a Open AI.

XML: Cross Lingual Language Model

XML fue publicado por Lample y Connau con Facebook en 2019 [25]. Es un transformer pre entrenado pensado en resolver tareas de lenguaje cruzado, que son:

- Modelamiento de lenguaje casual (CLM) como GPT
- Modelo de lenguaje enmascarado (MLM) como BERT
- Modelo de traducción de lenguaje (TLM) como un Traductor Neural.

Su foco es el modelamiento de lenguaje cruzado y aprovechar la nueva arquitectura de transformers en ello.

CTRL: Conditional Transformer Language Model for Controllable Generation

Keskar et al, en conjunto con el equipo de Salesforce Research, publican un modelo [26] que utiliza códigos para controlar el texto generado por el modelo. De esta forma,

utilizando una secuencia inicial el modelo genera texto acorde. Es un modelo de lenguaje casual (CLM) cuyo objetivo es generar el siguiente token en la secuencia utilizando prefijos de control. Fue brevemente el modelo más grande con 1630 millones de parámetros.

Estos son los principales modelos que han marcado una diferencia en los últimos años, pero la lista es mucho más larga (y sigue creciendo) [27]

Proyectos Similares

El trabajo más específico que encontramos corresponde a Ressmeyer et al [28] quienes realizan el entrenamiento de un modelo que genera publicaciones de Twitter similares a un Político Estadounidense. Utilizando publicaciones reales de dicho político entrenan un modelo LSTM-RNN y un GPT-2, siendo el segundo quien entrega los mejores resultados. Para darle un contexto a la generación de texto primero preparan el set de datos extrayendo las entidades y palabras clave de los tweets. Luego, en la generación, lo utilizan como contexto para manejar la generación de nuevos textos.

Para el entrenamiento de un modelo GPT-2 en un lenguaje diferente al inglés (original) nos basamos en [29]. Aunque realizaron el entrenamiento con idioma bengali a partir del modelo original GPT-2, la metodología utilizada es extensible a español.

Respecto al ajuste fino de tareas con GPT-2 observamos que en [30] desarrollan un modelo para solicitar patentes utilizando reclamaciones de patentes reales. Es posible con este modelo dar un tema y desarrolla la redacción del reclamo de patente.

En [31] Wang y Cho. concluyen que GPT-2 genera textos casuales de mejor calidad que BERT, además elaboran un mecanismo para evaluar con opinión de personas la fluidez de los textos generados.

El único trabajo que encontramos donde se entrena un modelo BERT a español fue en un trabajo de la Universidad de Chile [11]. A pesar de ser uno de los lenguajes con mayor cantidad de hablantes en el mundo, estos recursos son difíciles de encontrar.

3. Hipótesis y Objetivos

Nuestra hipótesis es que es posible sintetizar textos con una narrativa política dictada por parámetros de generación, utilizando un modelo GPT-2 entrenado con las publicaciones de los políticos chilenos.

El objetivo de este trabajo es recopilar el set de datos de las publicaciones recientes de los políticos en la red social de twitter y, con éstos, realizar un ajuste fino sobre un modelo GPT-2 para que aprenda de las autoridades políticas a cómo generar nuevos tweets en base a parámetros de control tales como tópico, sentimiento y partido.

Como parte de este objetivo entrenaremos un modelo GPT-2 en español que servirá de base para la generación de texto en español.

4. Datos y Metodología

4.1. Datos

El dataset base se construye a partir del listado de políticos en Chile [32] considerando Todas las Autoridades: Diputados, Senadores, Gobernadores, Presidente, Ministros, Seremis, Alcaldes. Sin excepción de Partido o Género. En total 698 Autoridades.

4.1.1. Extracción y enriquecimiento del Set de Datos

El procedimiento para obtener el set de datos base fue:

1. **Obtener cuentas twitter de autoridades políticas con actividad en Twitter**

   Para todas estas autoridades, buscamos sus cuentas de la red social Twitter utilizando las herramientas de búsqueda por API de twitter [33]. Luego de una revisión manual de los resultados (eliminando cuentas parodia, alcances de nombre, actividad y seguidores mínimos), sólo 355 con cuentas válidas con más de 100 publicaciones y en los últimos 12 meses y más de 100 followers se consideraron.

2. **Obtener Tweets**

   Para cada una de estas cuentas extrajimos los tweets publicados utilizando la API Twitter [34]. Estas consultas nos permitieron obtener hasta los 3200 tweets más recientes de cada cuenta.

3. **Filtrado**

Dado el objetivo del trabajo, seleccionamos los tweets relevantes y recientes. Por esta razón dejamos fuera los tweets de antigüedad mayor a 500 días y con menos de cinco palabras. Ya que estos tweets no aportan contexto (no es lo mismo un tweet de izquierda o derecha hace un par de años que hace un par de meses). También filtramos los Tweets que sean directo RT o citas de otras cuentas, Ya que esto no representan la forma de escribir del político.

4. **Obtención de características NLP**

Finalmente para cada uno de los tweets, extrajimos características del lenguaje natural: entidades, sentimientos y palabras claves de la publicación, que sirvieron para enriquecer nuestro set de datos.

Figura 16 Obtención de características: Sentimiento, Palabras clave y Entidades de los textos utilizando Amazon Comprehend

El flujo completo para la obtención del set de datos se muestra en el siguiente diagrama de la Figura 17.

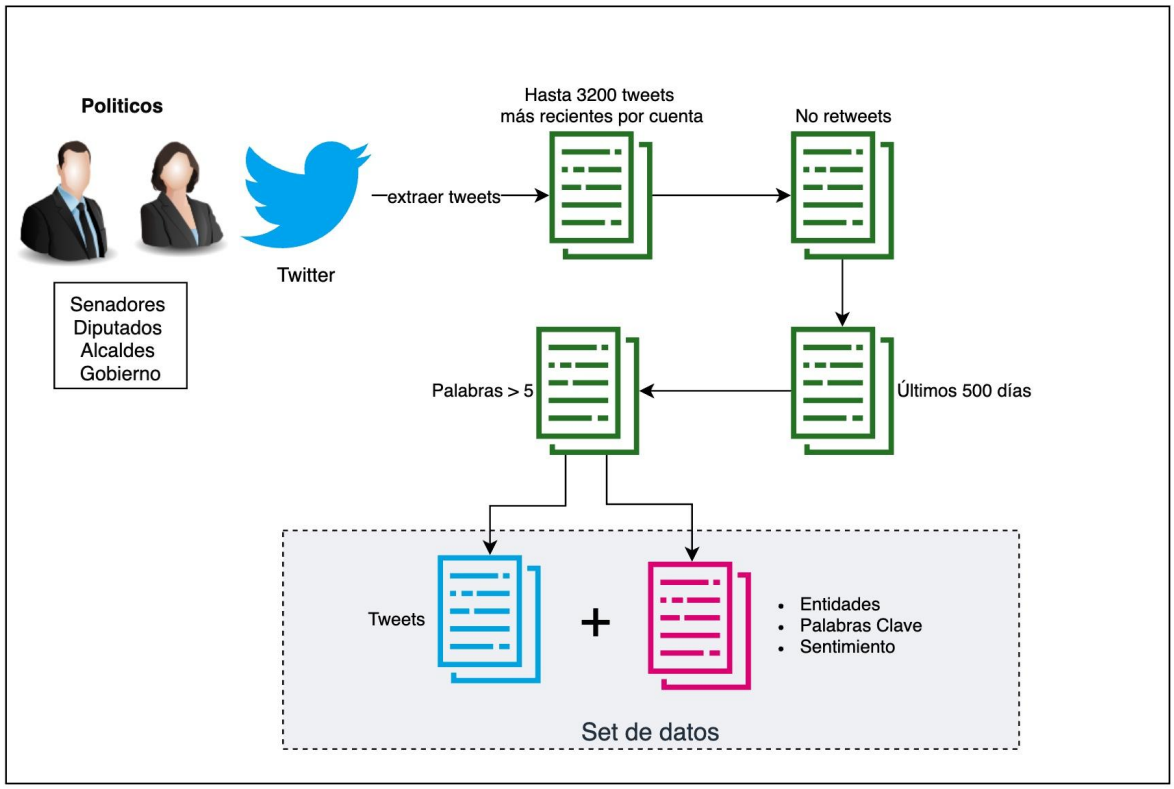


Figura 17 Construcción del dataset base

4.1.2. Descripción de los Datos

Para este proyecto utilizamos un set de datos principal correspondiente a las publicaciones de las autoridades políticas en Twitter. Cada observación corresponde un tweet. A continuación muestra el diccionario de los datos.

Datos relacionados al Político

Campo	Explicación	Ejemplo
Nombre	Nombre del político	Sebastián Piñera Echenique
Cargo	Cargo del político	Presidente de la República
Coalición	Coalición a la que pertenece el político	Chile Vamos
Partido	Partido al que pertenece el político	RN
screen_name	Cuenta twitter del político	sebastianpinera

Datos relacionados a la publicación

Campo	Explicación	Ejemplo
screen_name	Autor/a de la publicación	@sebastianpinera
id_str	ID único de la publicación	9928282873883
created_at	Momento de La publicación	2020/05/28 18:57:50
hashtags	Hashtags de la publicación	[#Coronavirus, #cuidemonos]
quoted_text	Texto que se está citando	Un saludo a todos los que...
retweeted_text	Texto que se está RT	Hoy nos encontramos...
text	Texto de la publicación	Seguimos con el plan paso a...

Twitter API Entrega muchos más atributos relacionados a la publicación (cantidad de RT, de favoritos, usuarios a los que se responde, que se retuitea, dispositivo utilizado y ubicación cuando está disponible) Nosotros solamente utilizamos los anteriores para este

proyecto. *quoted_text* y *retweeted_text* Solamente se utilizaron para validar si el tweet era original o no.

La combinación entre los atributos del político y los atributos de la publicación se realiza a través de *screen_name.*

Características de lenguaje natural

Adicionalmente al texto de la publicación se extrajeron características del lenguaje natural de este texto. Se complementa con entidades, palabras clave y sentimiento del texto del tweet.

Campo	**Explicación**	**Ejemplo**
id_str	ID único de la publicación	837348383844
entities	Entidades reconocidas	[Gobierno: ORGANIZATION]
key_phrases	Palabras Claves	["las soluciones", "el norte"]
sentiment	Sentimiento del texto	NEGATIVE

Estas características fueron obtenidas utilizando un modelo entrenado en español capaz de detectar sentimientos, entidades y palabras claves a partir de textos comunes de Amazon Comprehend. A través de este procedimiento realizamos un enriquecimiento de los datos que nos permite tener un mejor caracterizado del texto textos.

4.1.3. Exploración y Visualización

La cantidad de Tweets totales **1.173.079** fue filtrada a sólo publicaciones originales (no retweet y no citas), quedando en **527.485** tweets generados por los políticos, recordando que nuestro objetivo es sintetizar tweets de políticos y no de las publicaciones que citan o a las que le dan retweet.

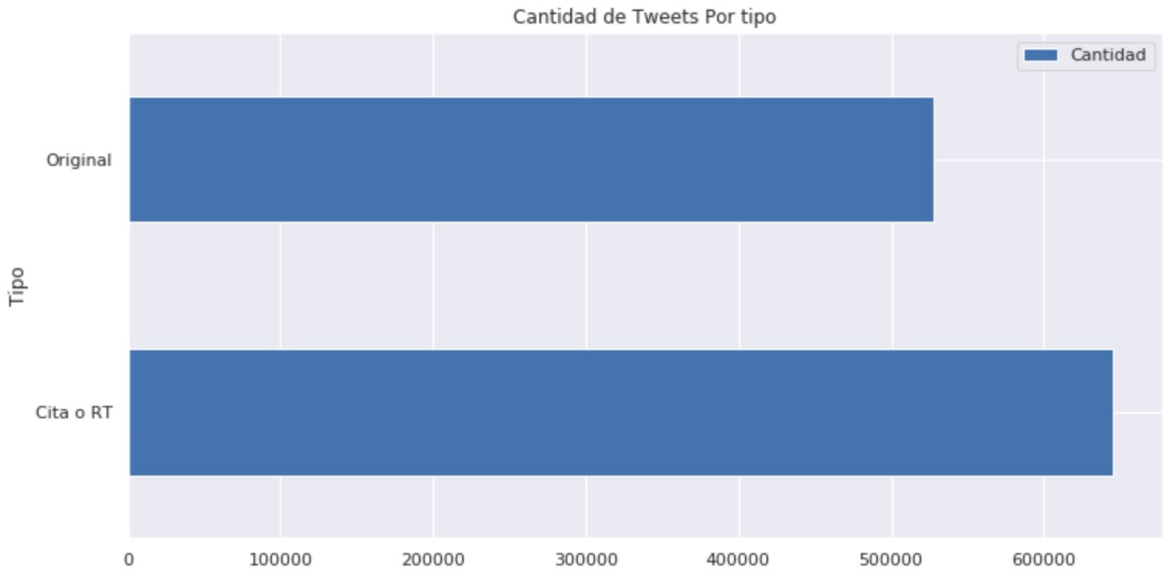


Figura 18 Publicaciones puras vs no originales

Nota: Las publicaciones originalmente pueden ser desde junio 2009 (primer tweet) hasta octubre de 2020. Para las próximas visualizaciones, y con el objetivo de visualizar un año completo, hemos filtrado los tweets los últimos 365 días con respecto al tweet más reciente, quedando en **226.280** publicaciones.

¿Cómo es el día en la red social twitter de los políticos?

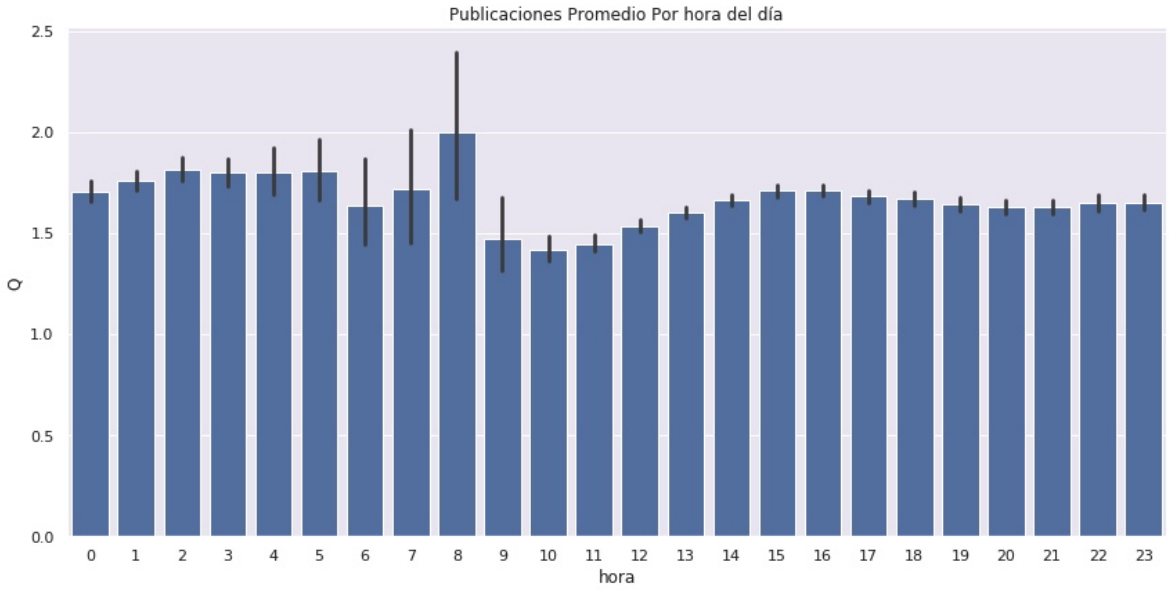


Figura 19 Promedio de publicaciones realizadas por hora del día para todos los políticos.

Dada la distribución de las publicaciones promedio durante el día que se ve en Figura 19, la actividad es permanente las 24 horas. Es interesante que la actividad también se realiza durante las horas de la madrugada. Un **85%** de los políticos de este estudio realizan publicaciones entre 2 y 7 de la madrugada además de las publicaciones que realizan durante el día. Es decir es una actividad permanente.

Adicionalmente a los promedios podemos observar las frecuencias de las publicaciones dentro del día. En la siguiente visualización (Figura 20) vemos por cada hora la dispersión de los tweets generados por cada político tomando solamente la hora (0-24) de cada día.

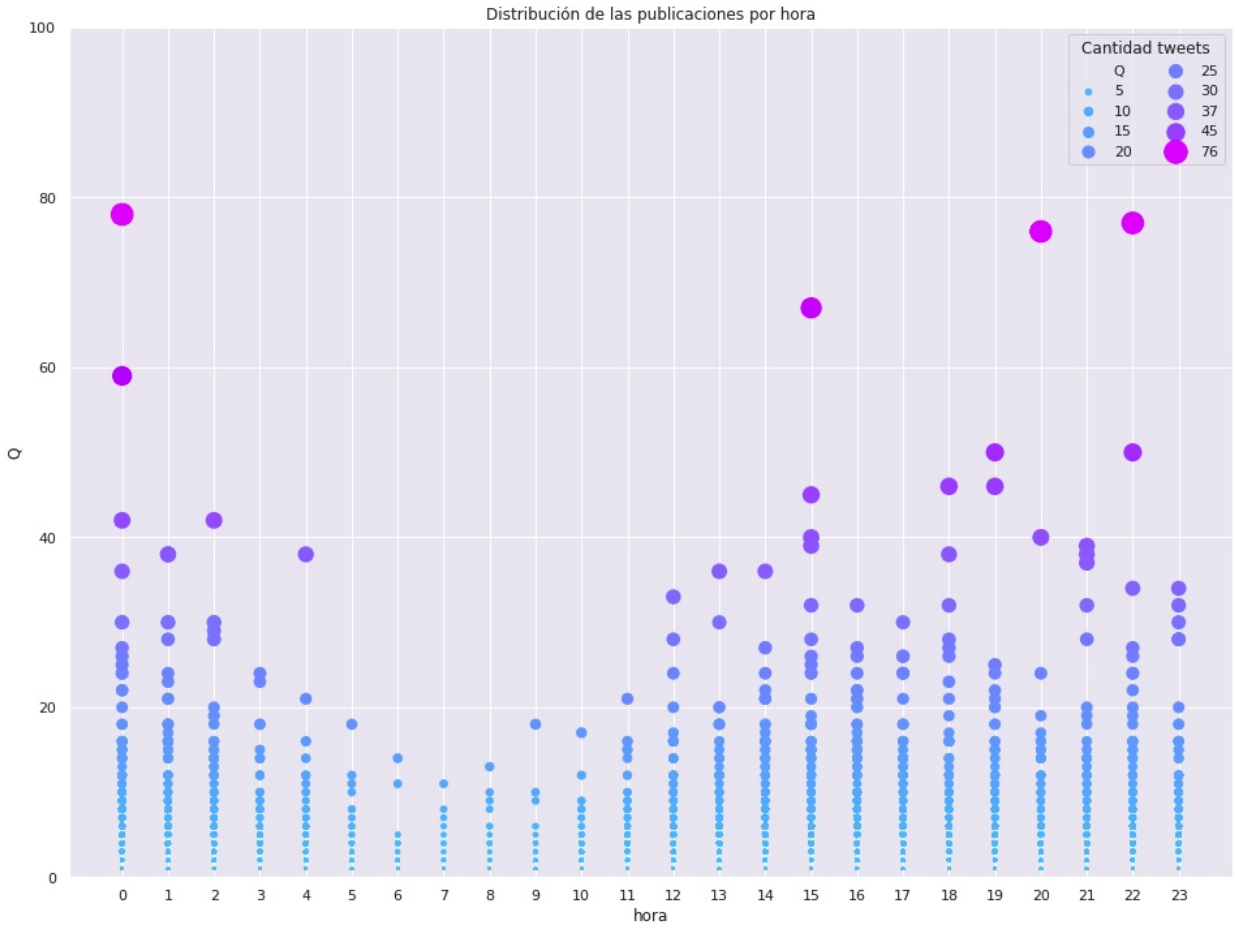


Figura 20 Bubbleplot con el conteo de tweets publicados por hora para las cuentas analizadas. El tamaño y color indica la cantidad de tweets para esa hora por usuario para los distintos días.

En la Figura 20 vemos cuentas que han publicado casi 80 tweets en una hora. Andrés Zarhi (alcalde UDI) realizó 78 publicaciones entre las 00:00 y las 00:55 del día 14 de Agosto de 2020. Por otro lado, Pamela Jiles (diputada PH) realiza 77 publicaciones entre las 22:05 y las 22:50 del 14 de agosto de 2020.

usario	dia	hora	Q
andreszarhi	121	0	78
PamJiles	227	22	77
Felixecologista	21	20	76
andreszarhi	75	15	67
GarinDiputado	225	0	59

Top Usuarios y las horas con más publicaciones del año

Por día en promedio los usuarios publican entre 3 y 4 tweets. Sin embargo los usuarios más activos publican en promedio sobre 20 tweets diarios.

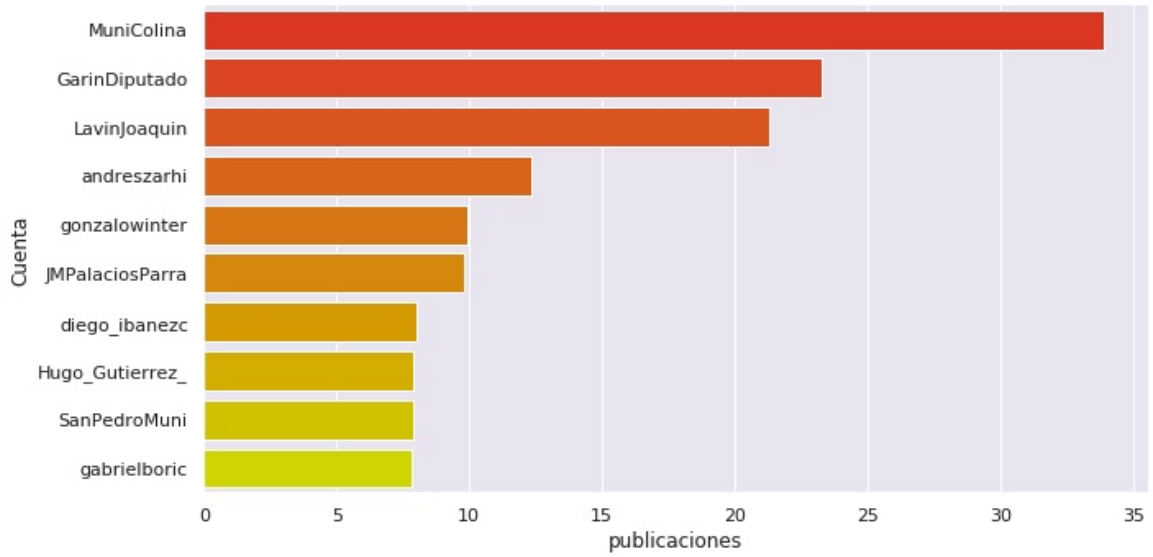


Figura 21 Top 10 Promedio de publicaciones realizadas diariamente por los políticos.

La imagen en los últimos 365 días es interesante, en la Figura 22 se muestra un bubbleplot que indica en el eje x la fecha desde 13 de octubre de 2019 al 13 de octubre de 2020.

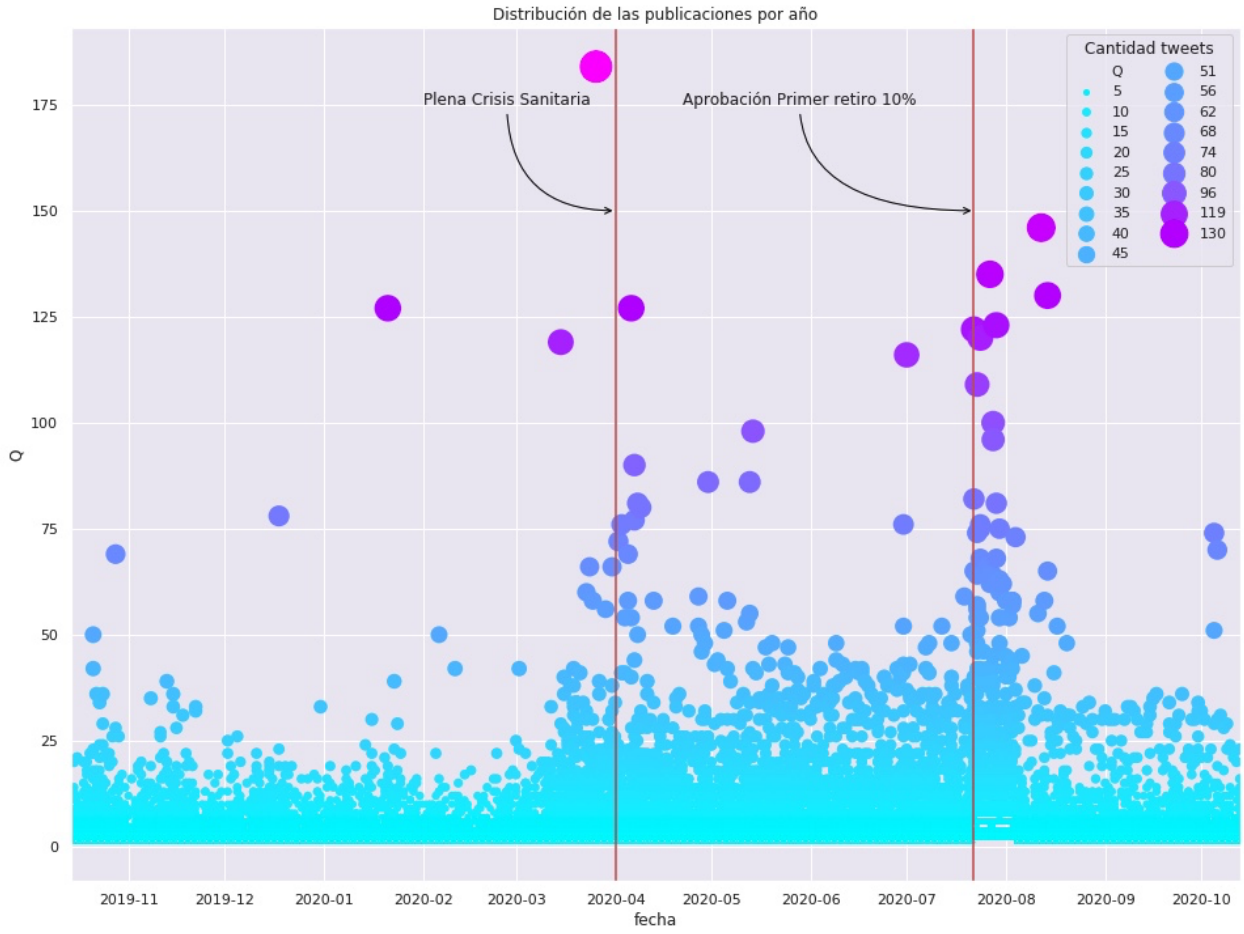


Figura 22 Bubbleplot con el conteo de tweets publicados por dia para las cuentas analizadas. El tamaño y color indica la cantidad de tweets ese día para cada político

En la imagen, cada burbuja representa un político y el tamaño es la cantidad de publicaciones de ese político ese día. Observamos dos momentos cruciales, cuando comienza con fuerza la crisis sanitaria y cuando se discute el retiro de 10% de fondos de AFP en el parlamento. A continuación los días donde están los máximos de publicaciones individuales.

fecha	Cuenta	Q
2020-03-26	andreszarhi	184
2020-08-12	GarinDiputado	146
2020-07-27	GarinDiputado	135
2020-08-14	PamJiles	130
2020-04-06	LavinJoaquin	127

En la distribución de publicaciones a nivel de partidos políticos, en la Figura 23 vemos que RN y UDI son los que aportan más observaciones en nuestro set de datos. La razón es porque tienen más representantes (gobierno, parlamentarios y alcaldes) con cuentas de twitter activas y además, en general, publican con más frecuencia.

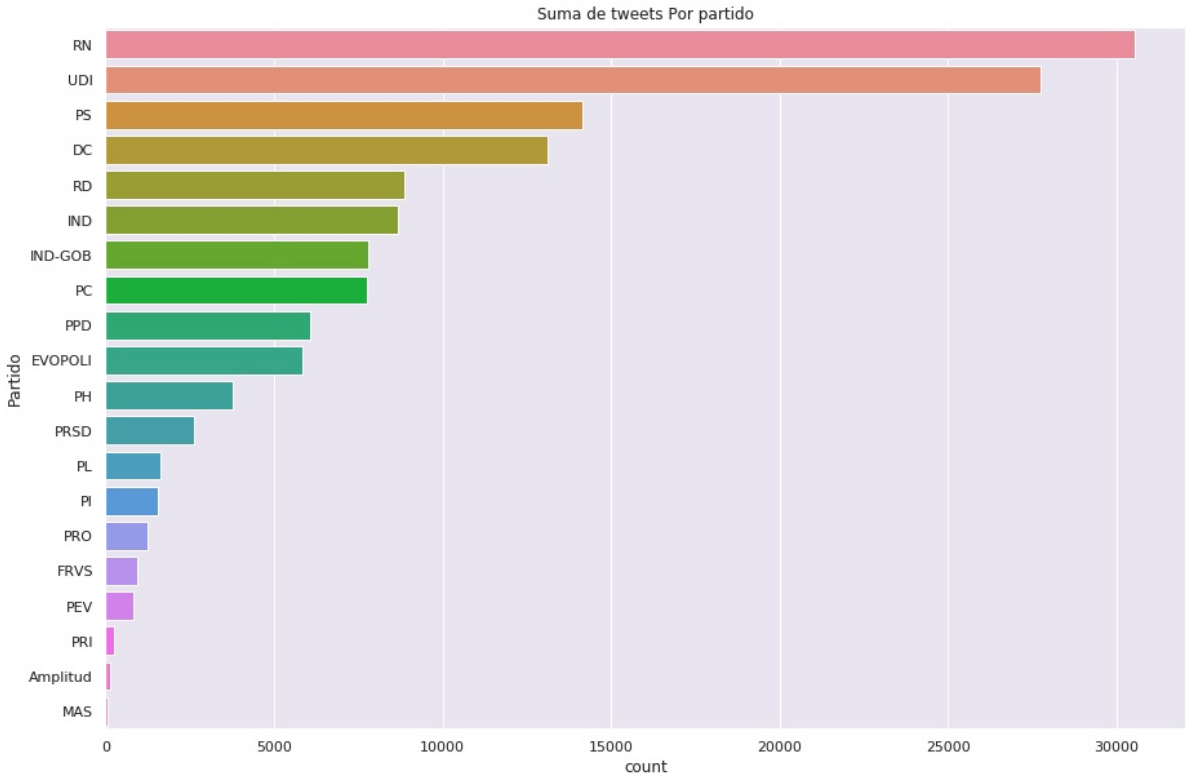


Figura 23 Distribución de tweets por partido en 365 días

Análisis preliminar del Texto de las publicaciones

Antes de comenzar nuestro trabajo decidimos explorar de manera simplificada la representación de los textos en los tweets de los Políticos. Utilizando la técnica de ***bag of words*** (removiendo stopwords previamente) entrenamos un modelo ***Word2Vec*** que permite representar de forma vectorial una secuencia de palabras. Para visualizarlo utilizamos una dimensioanalidad de dos vectores V1 y V2 los cuales se grafican a continuación:

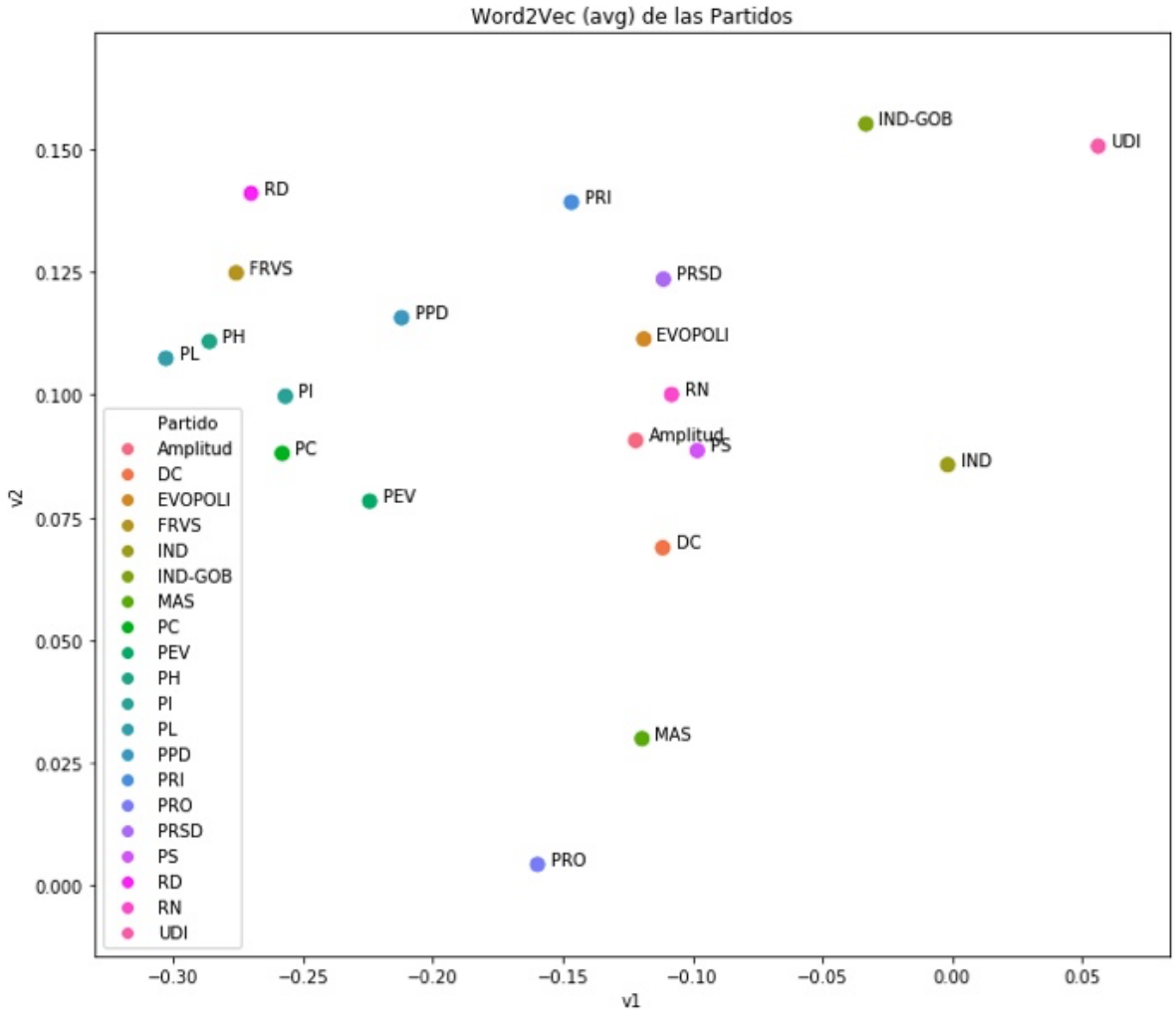
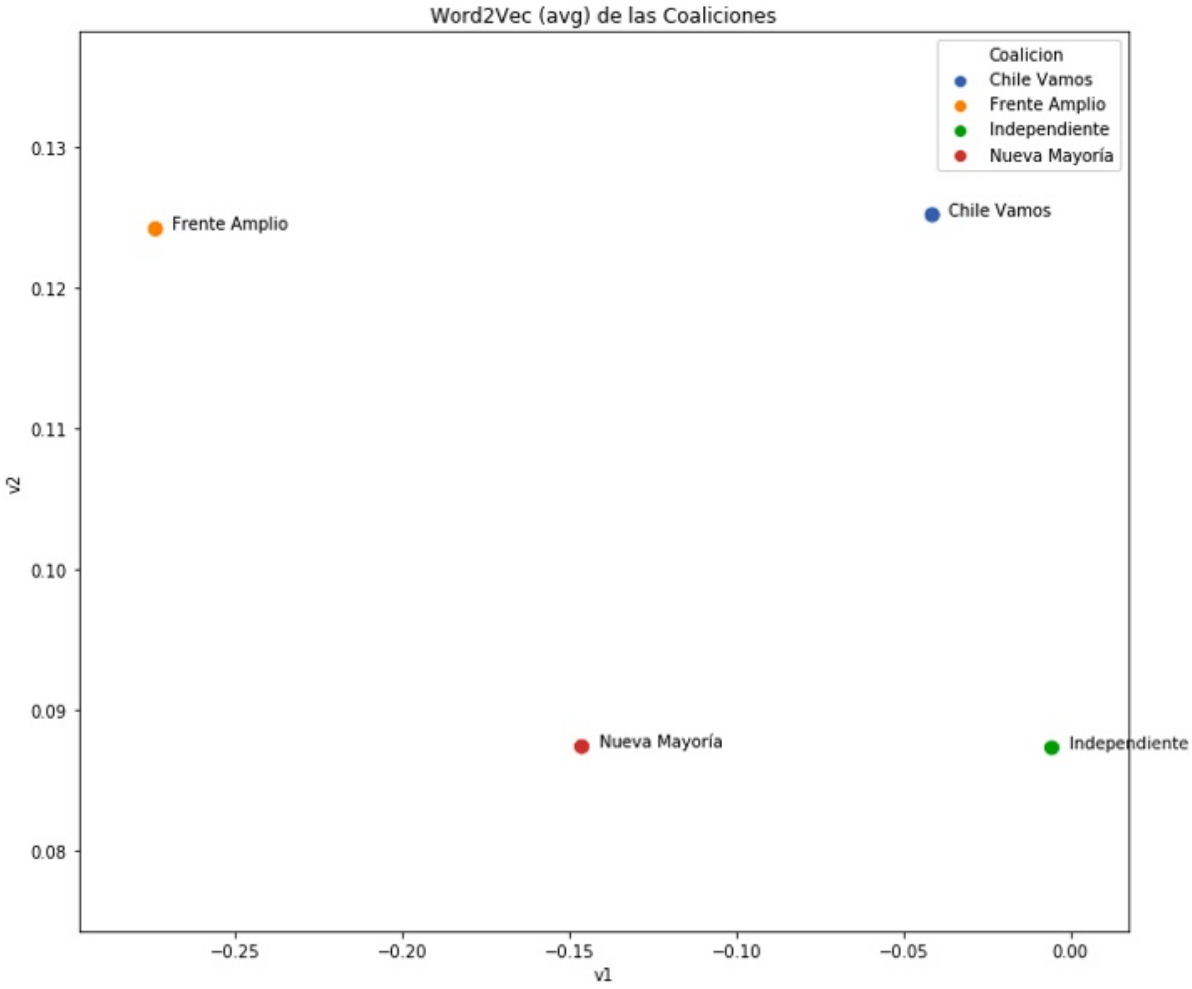


Figura 24 Modelo Word2Vect entrenado con la secuencia de los tweets del dataset aplicado a cada partido político.

Lo que nos muestra la Figura 24 es que los partidos se diferencian entre sí en el uso de las palabras, incluso podemos ver un partido de derecha como UDI muy alejado de los partidos de izquierda como RD, PC, PH, PL (coincidentemente la semilla del modelo *Word2Vect* calculó los vectores y no hubo intención de ubicarlos a la derecha o la izquierda).

Este hallazgo es promisorio, dado que nuestro objetivo es precisamente poder generar tweets con una orientación política.

Lo anterior es aún más evidente cuando agrupamos los tweets por coaliciones. Vemos un cuadrante bastante claro entre Las tres coaliciones y el grupo independientes.



Modelo Word2Vect aplicado a los textos publicados por las coaliciones

4.2. Metodología

A continuación se describen los experimentos realizados para obtener los resultados de este proyecto. En cada experimento se ha solicitado al modelo generar un grupo de 10 textos y hemos seleccionado una muestra de los mejores resultados o más relevantes del experimento. Para revisar los resultados completos y habilitar la reproducibilidad de los experimentos hemos publicado el siguiente repositorio https://github.com/ensamblador/este-politico-no-existe, donde se encuentran los notebooks y librerías utilizadas como también las tablas con todos los resultados completos obtenidos. Debido a las restricciones de tiempo no ha sido posible realizar una encuesta con personas para validar el nivel de naturalidad o mimetizado de las publicaciones sintetizadas a gran escala, y dejamos esto para trabajo futuro.

4.2.1. Ajuste fino de GPT-2 base para generación de nuevos Tweets

Durante el desarrollo del trabajo no encontramos un modelo de lenguaje casual entrenado en español que nos permitiera realizar un ajuste fino para los tweets a partir de un modelo en español. A modo de experimento, nuestra primera aproximación fue realizar un ajuste fino utilizando el modelo original GPT-2 base pre-entrenado inglés. Utilizamos la base de datos de los textos en los tweets para que el modelo aprendiera de ejemplos la manera de escribir de los políticos.

El procedimiento para ajuste fino de este modelo se representa en la siguiente imagen. Iniciamos concatenando todo el dataset de forma aleatoria en un gran corpus (1). Luego de dividir entre entrenamiento y pruebas (2), el set de entrenamiento se divide en partes iguales (*3*) que son codificadas con el tokenizador (4). Se entrenó durante 10 epoch (5) con evaluación cada 500 steps (6)

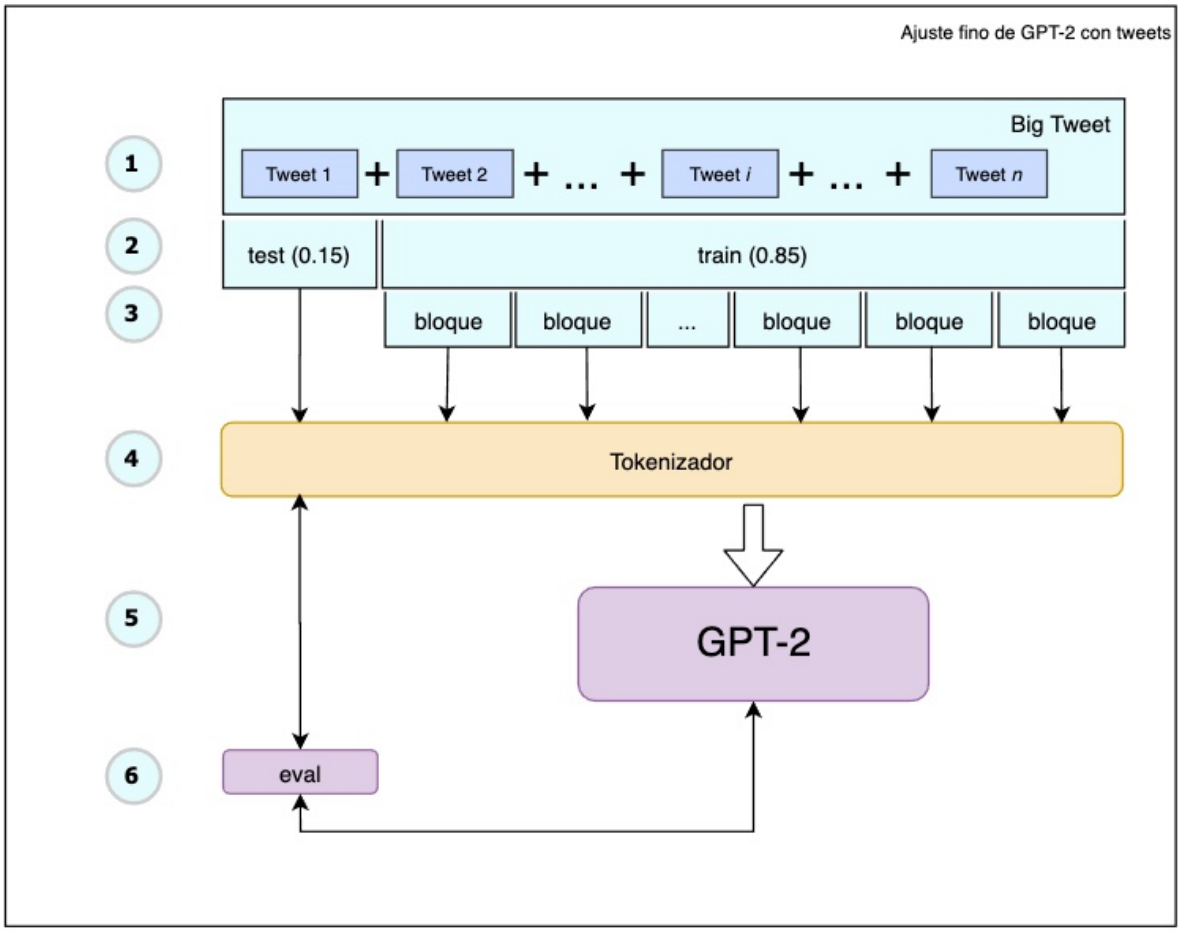


Figura 25 Ajuste fino de GPT-2 con tweets de políticos chilenos.

Para nuestra sorpresa, este experimento tuvo resultados positivos inmediatos. En la siguiente tabla se muestran algunos ejemplos de textos generados con el modelo base ajustado utilizando los tweets de políticos chilenos de derecha. No obstante la buena performance, si utilizamos un inicio muy pequeño el modelo genera texto que no mantiene

un hilo conductor y tiene pobre redacción. Por otro lado, utilizando un comienzo más amplio (más contexto) las generaciones mejoran.

Con los vecinos #CuidaTuVida @CorpColinaSalud informa las atenciones que se realizarán, a partir del lunes 8 de junio, en los CESFAM de Alpatacal y Esmeralda, de manera de enfocar sus prestaciones en las personas que surgen en esta situación.
Con los vecinos ¿#CuidaTuVida Si estás buscando un regalo para tu papá, te invitamos a la Vitrina Virtual del #CentroDelEmprendedor para el #DiaDelPadre. Encuentra ese obsequio perfecto para este día tan especial. Te dejamos también a algunos participantes (1 de 3).
Con los vecinos de la comuna estamos trabajando Con los vecinos de la comuna estamos trabajando para evitar contagios de #COVID—19 #QuedateEnCasa #UsaMascarilla @mario_olavarria @CorpColinaSalud #CuidaTuVida DIDECO va a tu barrio: Un servicio para que hagas consultas y resuelvas tus dudas sobre esta dirección del municipio.
Con los vecinos de la comuna estamos trabajando Con los vecinos de la comuna estamos trabajando #CuidaTuVida @CorpColinaSalud informa las atenciones que se realizarán, a partir del lunes 8 de junio, en los CESFAM de Alpatacal y Esmeralda, de manera de enfocar sus prestaciones en las personas que más lo requieran.

Tabla 1 Ejemplos de los tweets generados con Ajuste fino GPT-2 original

En la Tabla 1 se muestran ejemplos de los textos generados después del entrenamiento utilizando “Con los vecinos” y “Con los vecinos de la comuna” como texto inicial y dejando que el modelo complete el resto. Mientras más contexto mejor es la generación. Resultados

completos en https://github.com/ensamblador/este-politico-no-existe/tree/main/ajuste-fino-gpt2-tweets

4.2.2. Utilizando un nuevo modelo GPT-2 español con Tokenizador español.

Como siguiente paso entrenamos un Tokenizador y un modelo GPT2 con la misma arquitectura de la publicación original [24]. La metodología de entrenamientos que utilizamos es similar a [29] utilizando un corpus en español de artículos recientes de Wikipedia.

Tokenizador

El tokenizador utilizando por GPT-2 es un *Byte Pair Encoder* (BPE). Está entre una codificación a nivel de caracter y a nivel de palabras. Esto quiere decir que aunque tenga una representación de un token por palabra, mantiene un vocabulario base que le permite codificar palabras no vistas previamente en el entrenamiento mediante de la composición de varios códigos. De esta forma el tokenizador puede ser utilizado codificar otros textos, mientras se puedan componer con el vocabulario base. Esta es la codificación que se utiliza en la publicación original GPT-2.

A modo de ejemplo vea la Tabla 2, una secuencia simple como "*Buenos días a todos*" se podría codificar con Toekenizador BPE entrenado en español directamente palabra a palabra. Pero una secuencia desconocida como *"yu8ausy"* igualmente puede ser codificada y descodificada como si fuera una composición de varios tokens.

Secuencia	Tokens
Buenos días a todos	- Buenos -> 12339 - días -> 2004 - a -> 256 - todos -> 1378
yu8ausy	- yu -> 7756 - 8 -> 28 - a -> 69 - us -> 358 - y -> 93

Tabla 2 Ejemplo de una codificación BPE para palabras no conocidas

GPT-2 en español

Para el experimento realizamos el entrenamiento del modelo y tokenizador utilizando la metodología expuesta en [29]. Utilizando un corpus de Wikipedia 4.8 GB de datos (más de mil millones de palabras) [35] entrenamos un Tokenizador (***GPT2Tokenizer)*** y posteriormente un modelo GPT-2 Para la generación de lenguaje casual (***GPT2LMHeadModel)*** disponibles en librería transformers [12] para Tensorflow y Pytorch.

GPT2LMHeadModel es un modelo transformer con un cabezal de modelamiento de lenguaje al final, es decir devuelve un vector con probabilidades de elementos del vocabulario (tokens) de un largo parametrizable *n_ctx*.

Entrenamos el modelo utilizando un largo máximo de 512 tokens (*n_positions=n_ctx=512*) y 16 cabezales de atención (*n_heads=16*) la arquitectura utilizada de 12 capas interiores (*n_layer=12*) y dimensionalidad interna de 768 (*n_embd=768*).

El modelo se entrenó a 5 epochs utilizando 4 GPU en paralelo (Tesla V100) con un tamaño de lote de 16 por GPU. El proceso total tuvo una duración de 45.42 horas.

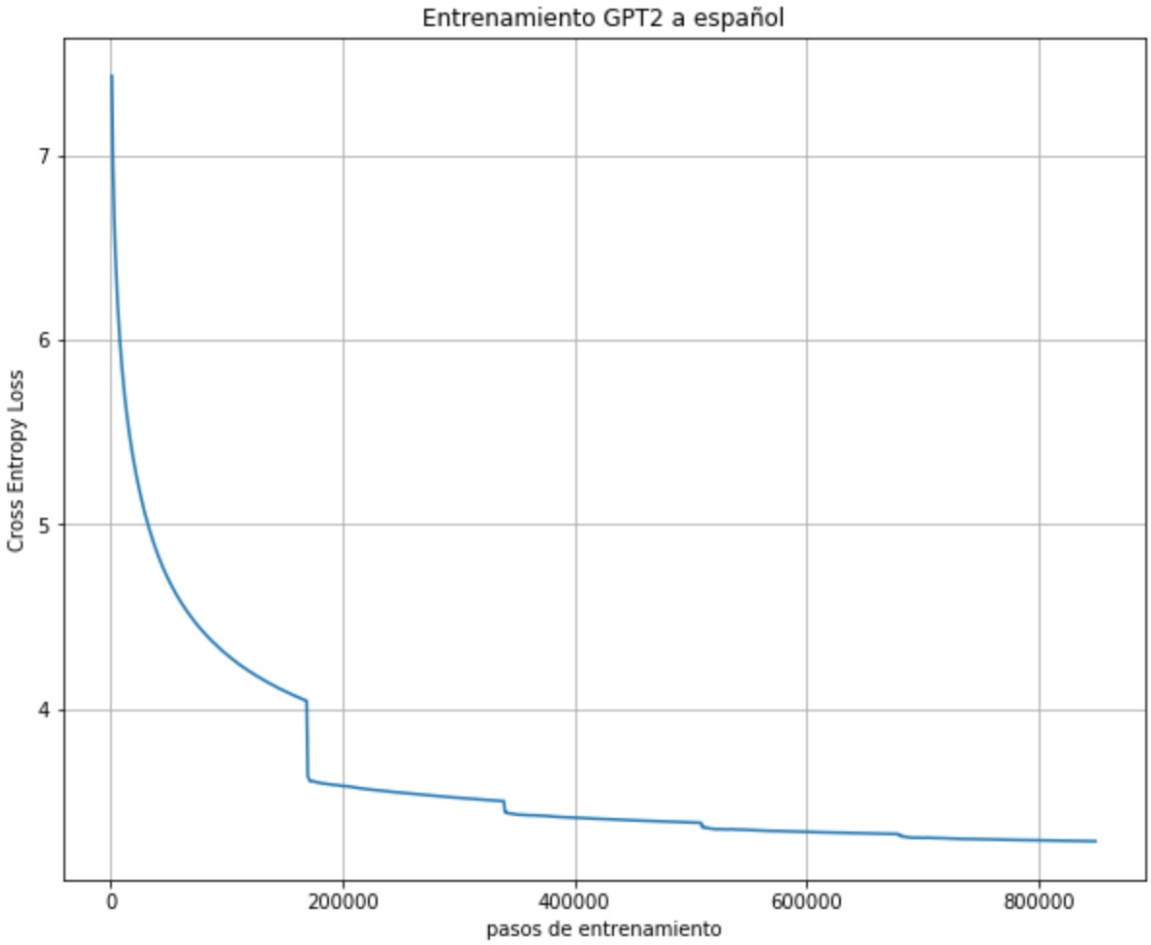


Figura 26 Entrenamiento GPT-2 en español. El modelo alcanza un loss (cross entropy) cercano a 3.28 y se estabiliza después de 5 epochs

Ahora que tenemos un modelo entrenado para español, podemos evaluar si sintetiza nuevos textos en base a texto inicial: ***"El 27 de enero del año 2010"***

El 27 de enero del año 2010, se anunció que la serie había sido renovada para una segunda temporada. La serie fue estrenada el de febrero de 2011 en Estados Unidos por la cadena The CW. El 11 de mayo de 2012, ABC renovó "The Walking Dead" como una serie de televisión basada en los hechos reales de la muerte de John F. Kennedy. En abril de 2013, AMC ordenó un spin-off

titulado "American Horror Story: Covenant", que se estrenó el 19 de julio de 2014. Esta es la primera serie derivada de esta serie, con el mismo nombre, mientras que salió al aire el 14 de junio de marzo de ese año 2011. También se emitió el 12 de su tercera temporada el 15 de noviembre de esa
El 27 de enero del año 2010, el papa Benedicto XVI lo nombró obispo auxiliar de la Arquidiócesis de Buenos Aires. Recibió la ordenación episcopal el 19 de febrero del mismo año, en la Basílica de Nuestra Señora de Luján (Buenos Aires). Su consagración episcopal tuvo lugar el 21 de marzo de 2011, manos del Nuncio Apostólico en Argentina, Mons. Juan Pablo II. Tomó posesión oficial de su diócesis el 20 de abril de ese año. El de mayo de 2012 fue nombrado por el Papa Francisco como co-consagruptor titular de las diócesis de Paraná, Santa Fe, San Isidro, Mar del Valle de los partidos bonaerenses de La Plata, Gualeguaychú, Diamante, Tres Arroyos, Chivillarreale, Nueve de Villaguay,
El 27 de enero del año 2010, la banda anunció en su página oficial de Facebook que estaban trabajando en un nuevo álbum. El 11 de febrero de 2011, el grupo anunció través de sus redes sociales que habían firmado un contrato discográfico con Nuclear Blast Records. La fecha de lanzamiento del álbum se fijó para el 12 de marzo de 2012, pero fue pospuesto hasta el 30 de abril de 2013 debido la pandemia de enfermedad por coronavirus de 2019-2020. En una entrevista con "The Wall Street Journal", los miembros dijeron que el álbum sería lanzado el 13 de mayo de ese mismo año siguiente. Sin embargo, también anunciaron que comenzarían el 15 días después de nuevo el 29 de junio de septiembre de 2020. Durante las mismas fechas finales del mismo mes

Tabla 3 Texto libre generado con un modelo GPT-2 Entrenado para español. Resultados completos en https://github.com/ensamblador/este-politico-no-existe/blob/main/train-gpt2-es/4-test.ipynb

Es interesante como el modelo aprendió nuevas relaciones del dataset que es posible aplicar en la síntesis. Para la primera generación relaciona la serie “The Walking Dead” con La muerte de John F. Kennedy (en Wikipedia de John F. Kennedy la palabra “muerte” aparece 14 veces) luego también lo relaciona con “American Horror Story” (efectivamente es una Historia de Horror Americana).

En el segundo y tercer ejemplo de la Tabla 3, se nota que el hilo del tema recorre el texto, ya sea de nombramientos de la Iglesia o lanzamiento de un disco.

4.2.3. Ajuste fino de GPT-2-español para generación de tweets

Después de entrenar un nuevo tokenizador y modelo GPT-2 que sea capaz de trabajar el español por defecto, volvemos a correr el ajuste fino con tweets pero ahora sobre nuestro modelo español.

Con los vecinos 👍 #CuidaTuVida El @MinDesarrollo presentó la página, para consultar por el pago del #IngresoDeEmergencia, una importante ayuda en la pandemia de #COVID19. Recuerda que ya no es necesaria la #ClaveUnica para solicitar este beneficio.
Con los vecinos 👍 #CuidaTuVida Si estás buscando un regalo para tu papá, te invitamos a la Vitrina Virtual del #CentroDelEmprendedor para el #DiaDelPadre. Encuentra ese obsequio perfecto para este día tan especial.

Con los vecinos 👨‍💼 @gaby_alcalde #ColinaSomosTodos Ahora son tres formas de donar a quien lo necesita: arma tu caja y entrégala en la oficina de Los Ingleses; llámanos y recogemos tu bolsa en tu casa; o entra a y sigue el link para comprar una caja en sencillos pasos

Tabla 4 Ejemplos de los textos generados después del entrenamiento utilizando un modelo y tokenizador para español. Resultados completos en https://github.com/ensamblador/este-politico-no-existe/tree/main/ajuste-fino-gpt2-tweets

Observamos los textos generados por el modelo ajustado a partir de GPT-2 español (Tabla 4) son muy similares a los generados con el modelo GPT-2 Base considerando el mismo ajuste fino con tweets (una tarea específica). De hecho, nuestro modelo en español se acerca bastante a las métricas de validación del modelo inicial (inglés).

De lo anterior se desprende que es posible utilizar directamente un modelo GPT2 base y vía transfer learning entrenar una tarea específica en español. Cabe destacar que este modelo GPT-2-Español fue entrenado con un volumen de datos equivalente al 10% de la cantidad de datos con los que se estrenó el modelo GPT-2 original [24].

Una interesante línea de trabajo futuro puede ser el entrenamiento de GPT2-large con un set de datos mayor, por ejemplo libros en español, blogs, noticias. Y comparar si el desempeño en tareas específicas en español mejora con respecto a la versión inglés.

4.2.4. Orientando la forma en que se genera el texto

Al momento de la predicción, el modelo entrenado expone atributos adicionales para modificar la forma en que se sintetizan los textos. Estos parámetros permiten modificar la

libertad en la elección de los tokens de la secuencia: conservador o creativo, largo, penalización por repetición, entre otros. A modo de ejemplo, es posible configurar la estrategia en la búsqueda de la secuencia con mayor probabilidad total (maximizar la probabilidad de la secuencia) o por el contrario, elegir siempre el token con más alta probabilidad. Es decir, podemos alterar el cómo se genera el texto, pero no existen opciones para orientar el contenido o temática de los textos más que con el prompt inicial.

4.2.5. Orientando el contenido del tweet: Izquierda y Derecha

Para generar tweets con orientación política, entrenamos 2 modelos distintos utilizando la metodología anterior derivados desde GPT-2: **GPT-2-Twitter-Derecha** y **GPT-2-Twitter-Izquierda.** Este ajuste fino lo realizamos utilizando Datasets de UDI por la Derecha y PC, RD, FRVS, PH, PI y PL por la Izquierda.

Selección de Datos de Izquierda y Derecha

Para dividir el set de datos, tomamos el set de datos de tweets etiquetado con el partido político y coalición. Luego ejecutamos el siguiente procedimiento:

1. Remover stop words de los textos.
2. Generación de bag of words de 1-gram
3. Entrenamos un modelo de Word2vect con 2 dimensiones

4. Promediamos v1 y v2 para cada partido político.
5. Visualización (siguiente figura)

Como vimos en las exploración y visualizaciones, a nivel de vocabularios y su uso hay una diferencia entre el partido UDI y un cúmulo de partidos que podemos llamar de izquierda (RD, FRVS, PH, PL, PI y PC).

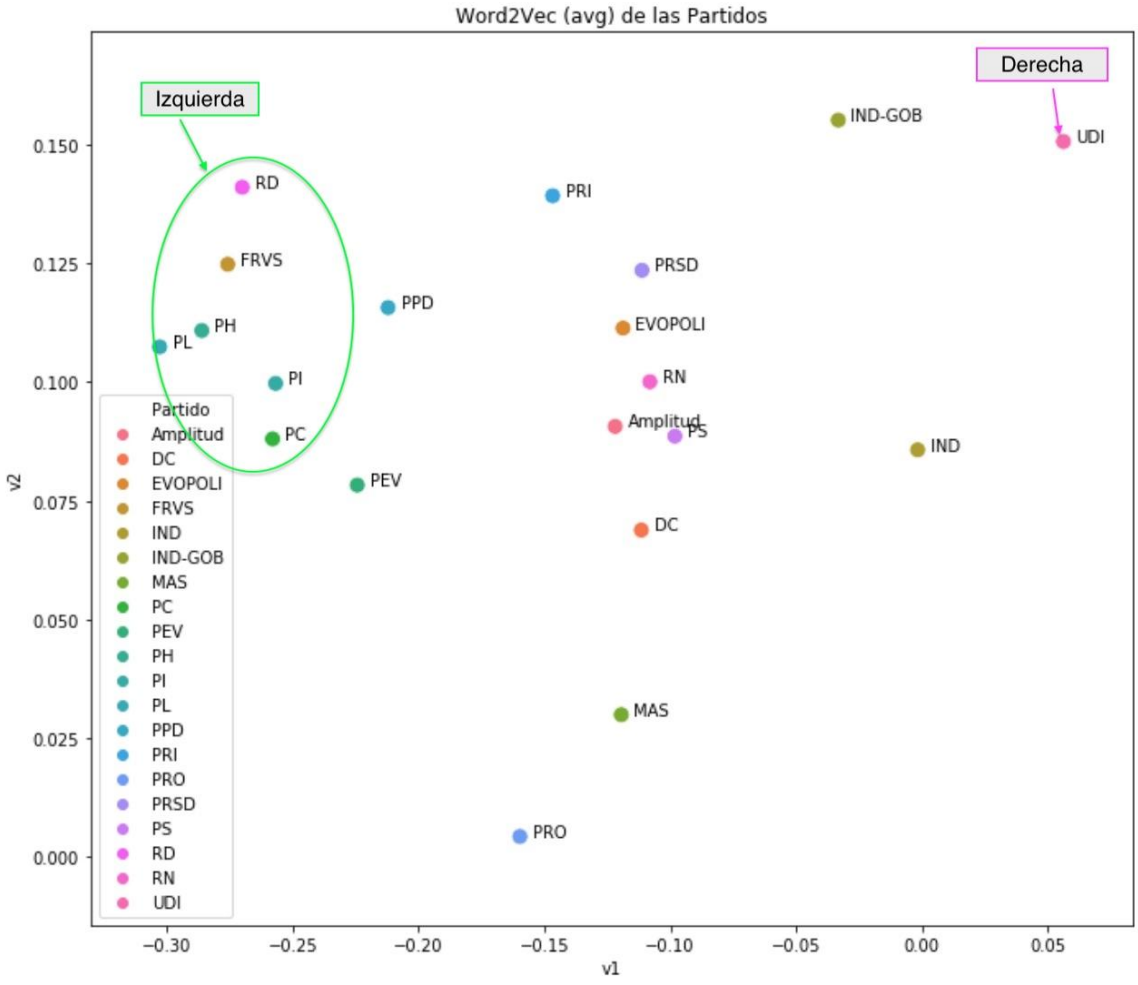


Figura 27 Representación Word2Vec de los tweets de los partidos y la separación entre Partidos Opuestos

La cantidad de tweets de Derecha es **31.631** y la por otro lado Izquierda es **22.571,** son volúmenes que nos permiten entrenar dos modelos independientes.

Luego de entrenar ambos modelos, se les solicita completar 10 veces los siguientes textos: ***"Con los vecinos ", "Vamos con todo ", "El presidente" y "El 27 de enero del año 2010"***. A continuación se muestran algunos de los resultados (los resultados totales se pueden ver en el repositorio Git de esta tesis)

Inicio	Tweet Derecha	Tweet Izquierda
Con los vecinos	**Con los vecinos** □□❤️ de la Provincia de #Melipilla y el país, nuestra Gobernadora @mpazsante junto a la Alcaldesa Carmen Bou, encabezaron la entrega de más de 4000 cajas de mercadería para ayudar a más familias afectadas por la #cuarentena por el #COVID19. #CuidémonosEntreTodos #PlanCoronavirus	**Con los vecinos** □□✊ #NoAlPagoDeLosServiciosBásicos para los + vulnerables @Diputados_PC Hoy se vota la admisibilidad del #PostnatalDeEmergencia en la cámara de diputados y diputadas Ahora pasa al Senado donde espero que podamos darle certezas a l@s vecinxs Necesitamos una #ConstituyenteParitaria y #PuebloqueCrece
Vamos con todo	**Vamos con todo** 💪 #CuidaTuVida El @MinDesarrollo presentó la página, para consultar por el pago del #IngresoDeEmergencia, una importante	**Vamos con todo** 💜✊ El proyecto de #40horas mejora la calidad de vida de las y los trabajadores y cuenta con apoyo transversal de diputad@s de Chile Vamos

	ayuda en la pandemia de #COVID19. Recuerda que ya no es necesaria la #ClaveUnica para solicitar este beneficio.	y el gobierno de @sebastianpinera Hoy más que nunca debemos avanzar hacia un #PlebiscitoConstituyente para q el pueblo decida si quiere o no una #NuevaConstitución
El presidente	**El presidente** @sebastianpinera anunció hoy la #NuevaAgendaSocial con mayores pensiones, aumento del ingreso mínimo, freno al costo de la electricidad, beneficios en salud, nuevos impuestos para altas rentas y más medidas que pueden ver aquí:	**El presidente** @sebastianpinera debe responder políticamente por las violaciones a los DDHH ocurridas bajo su mandato #PiñeraCulpable
El 27 de enero del año 2010	**El 27 de enero del año 2010** 👏 a partir de las 22:00 hrs. entramos en funcionamiento la #ComisaríaVirtual de @Carabdechile, que atenderá a los vecinos que requieran de su ayuda.	**El 27 de enero del año** El 27 de enero del año 2010 @sebastianpinera promulgó la ley que rebaja la jornada laboral a #40horas semanales □□✊ Este es un avance en justicia y reparación para las víctimas de violaciones a los DDHH durante la dictadura cívico militar #PiñeraCulpable

Tabla 5 Generación de tweets casuales de Izquierda y DerechaResultados en https://github.com/ensamblador/este-politico-no-existe/tree/main/ajuste-fino-gpt2-tweets

Observamos la diferencia diametral en los tweets generados entre políticos de UDI e Izquierda (es lo esperado, ya que refleja la diferencia del set de datos utilizado). También es destacable que aunque sólo una tarea implica hablar del presidente, en los tweets de la izquierda el tema presidente puede aparecer espontáneamente las síntesis. En otras palabras, el tema es muy frecuente en sus publicaciones.

Hasta el momento hemos logrado el objetivo de generar publicaciones de izquierda o derecha solamente haciendo separación del set de datos de entrenamiento. De igual manera, si quisiéramos generar tweets de izquierda con un sentimiento negativo tendríamos que subdividir el set datos de izquierda en los tweets con sentimiento negativo, y a partir de ellos, entrenar el modelo.

Esta metodología es útil a la hora de copiar la forma en que un sector o un político realiza las publicaciones, pero para este proyecto buscamos generar contenido del texto temático que no requiera tener un modelo por tema. No obstante como caso de uso se puede explorar esta metodología aplicaciones de lenguaje casual en tareas con jerga específicas. Por ejemplo, asistentes virtuales de un rubro específico, redacción de contratos, generación de descripciones de productos.

4.2.6. Control fino en la generación de tweets

Un mecanismo más sofisticado para controlar la salida del texto es entregar un contexto detallado como input y esperar que el modelo aprenda las relaciones entre el texto

y ese contexto gracias a la atención múltiple. Esto permite que el texto generado sea una especie de respuesta a ese input [28]. Para aumentar ese control en base a parámetros se debe modificar el dataset de entrenamiento para establecer esos controles y permitir al modelo aprender de ellos [26]. En [36] se plantea una metodología novedosa para poder controlar la generación de texto utilizando distintos elementos y aprovechando la capacidad de los transformers de poner atención en varias formas a la secuencia.

Preprocesamiento del set de datos

Para generar los contextos, utilizamos las características ya obtenidas en el set de datos base:

- **Coalición**: Coalición política del autor de la publicación
- **Partido**: partido político al cual pertenece el autor de la publicación.
- **Sentimiento**: sentimiento extraído desde el texto con herramientas de Sentiment Analysis.
- **Entidades**: entidades encontradas en el texto de la publicación, separadas por un espacio.
- **Hashtags**: Hashtags de publicación separadas por un espacio.
- **Frases**: frases clave encontradas en el texto de la publicación, separadas por espacios.
- **Tweet**: Texto de la publicación.

COALICION	PARTIDO	SENTIMIENTO	ENTIDADES	HASHTAGS	FRASES	TWEET
Chile Vamos	IND-GOB	NEUTRAL	CarolCBown s_villarrealb sebastianpinera	CuentaPública ChileenMarcha		Ya estamos en el Congreso con los subses @Caro...
Chile Vamos	RN	NEGATIVE		Araucanía CuentaPública		⭕ "Combatir con máxima voluntad y firmeza, sie...
Chile Vamos	RN	NEUTRAL	Presidente Ministerio de Agricultura y Aliment...	CuentaPública	#CuentaPública ANUNCIO Nuestro Presidente la c...	#CuentaPública\n📣 ANUNCIO| Nuestro Presidente a...

Tabla 6 Muestras del dataset utilizado para el ajuste fino con parámetros de control.

En este entrenamiento, todo lo que no es el texto del tweet es considerado un contexto que lo describe. Entonces mediante el uso de tokens especiales incorporamos este contexto como prefijo para las muestras de entrenamiento.

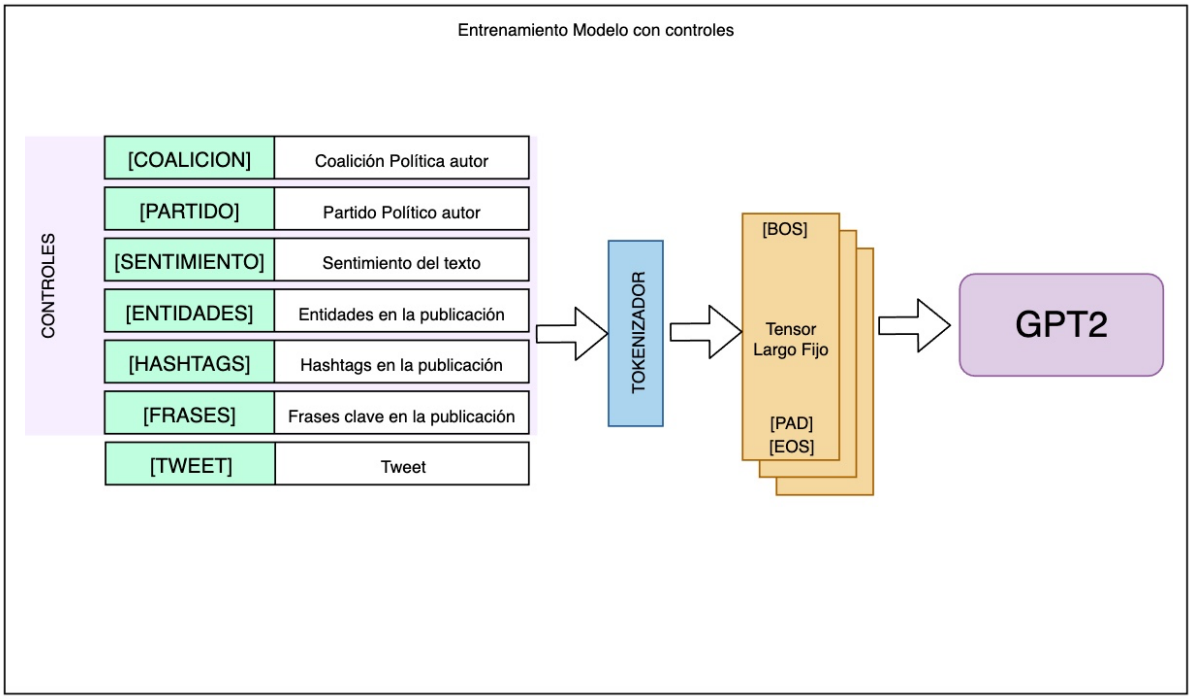


Figura 28 Construcción de las muestras de entrenamiento.

Los tokens especiales (*[COALICION], [PARTIDO], [SENTIMIENTO], [ENTIDADES], [HASHTAGS], [FRASES], [TWEETS]*) marcan el inicio del segmento

correspondiente en la secuencia. Manteniendo siempre la misma estructura, el texto total (tweet + contexto) es convertido en un vector con el tokenizador.

Verificamos el largo del vector resultante, en caso de no contar con el largo requerido por el modelo se agregan caracteres de relleno (*[PAD]*), por otro lado si el vector es más largo que la admitido por el modelo, el vector se trunca (primero se eliminan atributos frases clave, luego entidades y hashtags). Finalmente se le agrega el carácter de fin de secuencia (*[EOS]*).

Para que el modelo aprenda estas relaciones entre el contexto y la publicación es importante mantener la estructura y largo de las muestras para entrenamiento y validación.

Aplicamos esta transformación al set de datos completos de los tweets (170.306 muestras) utilizando un 90% para entrenamiento y un 10% para pruebas. Entrenamos el modelo GPT-2 utilizando como batch_size=6, learning_rate=5e-5, durante 3 epochs para 153.275 muestras. El entrenamiento tomó aproximadamente 80 minutos por epoch (cerca de 4 horas en total)

Terminado el entrenamiento, se le pide al modelo a generar tweets utilizando un contexto definido. En la Tabla 7 se observan los resultados, se le indica al modelo los parámetros de contexto (coalición, partido, sentimiento, entidades, hashtags, frases) aparecen en primera fila, el tweet es generado por el modelo en base a esos parámetros. Note que no se requiere dar un inicio al texto de la publicación, sólo los contextos.

COALICION	PARTIDO	SENTIMIENTO	ENTIDADES	HASHTAGS	FRASES
Chile Vamos	UDI	POSITIVE	sebastianpinera		
Tweet generado	*"Muy buena reunión de trabajo con el Pdte @presidencia_cl para avanzar hacia una mejor calidad de vida de nuestros vecinos y vecinas en esta emergencia sanitaria que afecta a nuestro país."*				

COALICION	PARTIDO	SENTIMIENTO	ENTIDADES	HASHTAGS	FRASES
	PC	POSITIVE		constitucion	25 de Octubre
Tweet generado	*"Aprobamos en general el proyecto de ley que establece una nueva Constitución para la ciudadanía."*				

COALICION	PARTIDO	SENTIMIENTO	ENTIDADES	HASHTAGS	FRASES
	PS	NEGATIVE	AFP		AFP
Tweet generado	*"El problema de las AFP es que las familias más vulnerables no pueden seguir esperando"*				

Tabla 7 Tweets políticos generados utilizando parámetros que definen la síntesis.

Otros ejemplos con un tema común, donde cambiamos el sentimiento y coalición en el contexto:

COALICION	PARTIDO	SENTIMIENTO	ENTIDADES	HASHTAGS	FRASES
Chile Vamos		NEGATIVE		coronavirus	coronavirus
Tweet generado	*"Estas son las medidas necesarias para enfrentar esta pandemia, por eso debemos cuidarnos entre todos y todas"*				

COALICION	PARTIDO	SENTIMIENTO	ENTIDADES	HASHTAGS	FRASES
Chile Vamos		NEUTRAL		coronavirus	coronavirus
Tweet generado	*"Para prevenir el contagio del virus, debemos cuidarnos entre todos y todas."*				

COALICION	PARTIDO	SENTIMIENTO	ENTIDADES	HASHTAGS	FRASES

Nueva Mayoría				coronavirus	coronavirus
Tweet generado	*"Nos preocupamos por la prevención del contagio del Covid-19"*				

COALICION	PARTIDO	SENTIMIENTO	ENTIDADES	HASHTAGS	FRASES
Nueva Mayoría		**POSITIVE**		coronavirus	coronavirus
Tweet generado	*"Gracias a todos quienes nos ayudan a prevenir el contagio del virus"*				

Tabla 8 Tweets políticos generados con un tema común, cambiando coalición y sentimiento.

Los resultados estan disponibles para su revisión en el repositorio de este trabajo https://github.com/ensamblador/este-politico-no-existe/tree/main/contextual-training como también los pasos para reproducir los experimientos.

Observe el caso anterior, la diferencia de contexto es sólo el sentimiento y la síntesis del tweet refleja esto.

Es interesante como este modelo es capaz de modificar la síntesis en base a los distintos elementos de contexto, una especie de parámetros del contenido generado. Lo que estamos haciendo es en la práctica utilizar un vector de mensaje (entidad, palabra clave, hashtags, sentimiento, coalición y partido) y dejar que el modelo resuelva con su aprendizaje cuál es la mejor redacción para eso.

5. Resultados

En este trabajo logramos sintetizar exitosamente tweets de autoridades políticas utilizando un modelo GPT-2 pre entrenado, y ajustado usando un set de datos específico mediante de un mecanismo de atención al contexto. Al ser una tarea muy específica, el desempeño de este modelo es mucho mejor que un generador casual.

5.1.1. Perplexity

Una de las formas de medir un modelo de lenguaje es utilizar la métrica Perplexity. Ésta se define como la probabilidad de generar una sentencia con las palabras correctas, normalizado por el número de palabras. Esta métrica se calcula sobre un set de datos no visto por el modelo (set de datos de pruebas).

$$Perplexity = P(W_1, W_2, \ldots W_N)^{-\frac{1}{N}}$$

Donde: W_1, W_2,... W_N Son las palabras que componen la sentencia y N es el largo de la sentencia.

En términos prácticos, Perplexity (PPL) indica cuán bueno es el modelo para predecir texto no conocido en el entrenamiento, a menor PPL, el modelo muestras mejores probabilidades en las sentencias de testeo (dado que es el inverso de la probabilidad). En otras palabras, el modelo queda menos “perplejo” ante las secuencias no conocidas. Al obtener la raíz N se normaliza por el largo de la sentencia.

Aunque es difícil definir un nivel de perplexity adecuado para tareas específicas, utilizaremos esta métrica por que está presente en la publicación original de GPT-2 y la podemos calcular de forma intrísica en todos los modelos que trabajamos es este proyecto, lo que nos permite compararlos entre ellos.

5.1.2. Métricas

En la siguiente tabla podemos ver la métrica Perplexity del modelo final comparado con el modelo base:

Modelo	Tarea	Perplexity	Cross Entropy Loss
GPT-2 Reportado en [24]	WIkiText2 sin ajuste fino (zero shot)	22,76	3,125
GPT-2 Base Ajustado con tweets	Generación de Tweets Casuales (izq y der)	3,411	1.1227
GPT-2 Pre-entrenado Español con ajuste fino de tweets	Generación de Tweets Casuales (izq y der)	5,02	1.615
GPT-2 Base Ajuste Fino de tweets	Generación controlada de Tweets usando prefijos de contexto	3,6026	1,2816
GPT-2 Pre-entrenado Español con ajuste fino de tweets	Generación controlada de Tweets usando prefijos de contexto	3,9470	1,373

Tabla 9 Comparación de Perplexity para las distintas tareas realizadas en este trabajo

En el primer caso corresponde a una generación sin ejemplos, mientras que nuestro proyect se realiza un ajuste fino. Aún así, la diferencia en las métricas muestra un muy buen desempeño del modelo final de este trabajo. Cabe destacar que el la mejor métrica siempre lo obtiene el ajuste fino sobre GPT-2 Original.

5.1.3. GPT-2 Base para tareas en español.

Un aspecto interesante para destacar es que en este trabajo se evidencia es que el modelo GPT-2 pre entrenado inglés puede ser utilizado para tareas específicas en español, siempre cuando se realice un ajuste fino con el set de datos en español.

En la Figura 29 se muestra el desempeño (cross entropy loss para los datos de validación) para la generación de tweets casuales, es decir sin contexto más que una secuencia inicial. El modelo base tiene mejor capacidad de generar nuevos tweets desde el

principio.

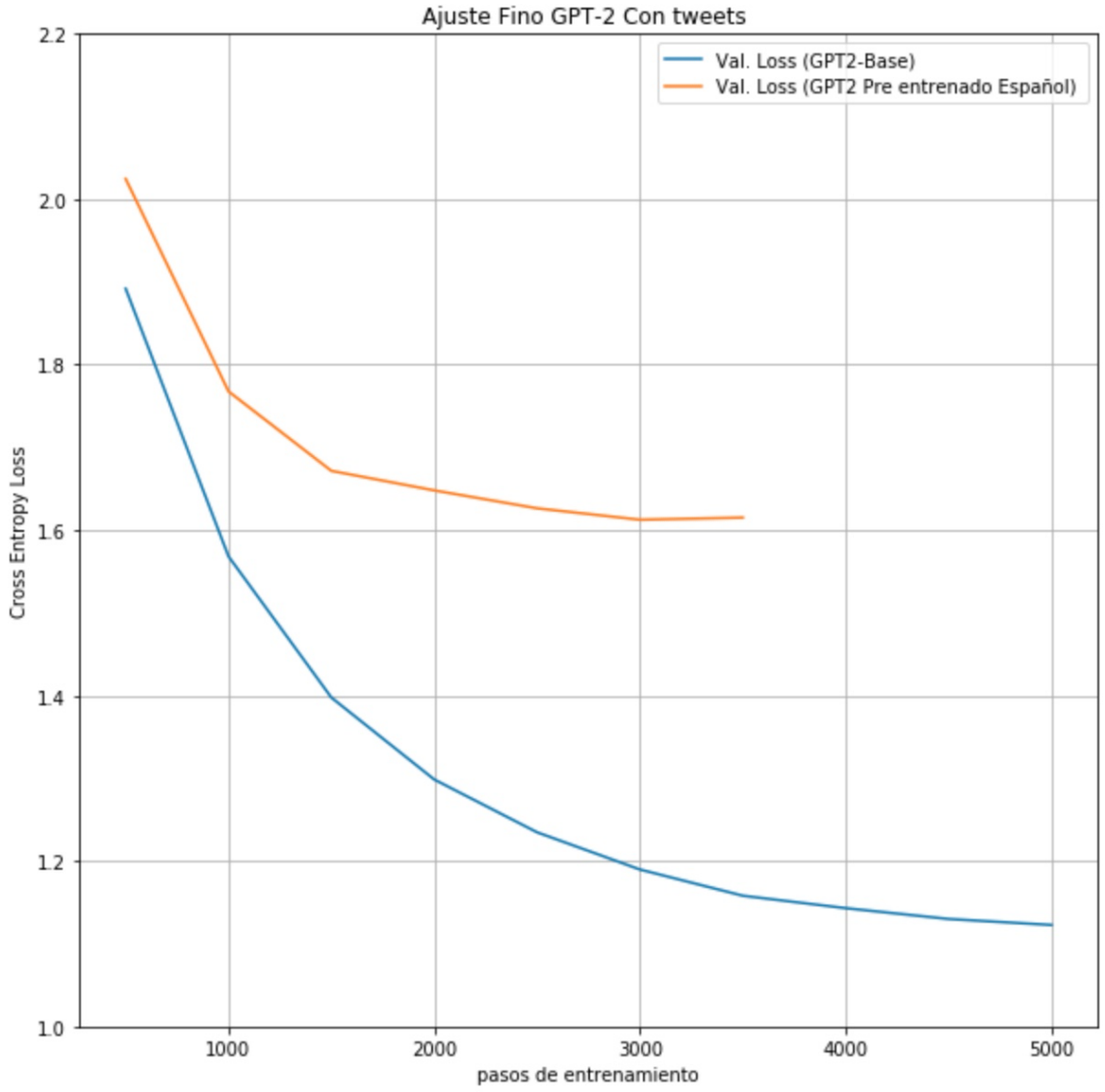


Figura 29 Desempeño del modelo GPT2-Base-tweets vs GPT2-ES-tweets. Utilizando el mismo dataset y parámetros.

Aunque sólo utilizamos un 10% del tamaño de corpus con el que se entrenó el modelo original, los resultados que alcanza el modelo pre-entrenado en español para la tarea específica son levemente inferiores. Entrenar un modelo en español para una tarea específica no representa una ventaja sobre el modelo base. Sólo se justificaría para ejecutar tareas de generación casual en español o para mejorar tareas de ajuste fino.

6. Conclusiones

Este trabajo muestra que, utilizando las nuevas arquitecturas innovadoras para el modelamiento y generación de lenguaje, es posible entrenar un modelo para realizar tareas específicas de generación de texto casual y generación texto controlado en español, partiendo de un modelo pre entrenado GPT-2 (ya sea en inglés o español) y haciendo el ajuste con un dataset especifico vía transfer learning.

Hemos elegido como experimento los tweets de los políticos en Chile para explorar los modelos con arquitecturas transformers, pero las aplicaciones pueden ir más allá de este simple caso de uso. Un modelo casual puede aprender el idioma como también puede aprender de una jerga específica, considerando que su característica más importante es la capacidad de manejar la atención de distintas formas dentro de la secuencia. La aplicación de esta innovación podría ampliar nuevas capacidades en servicios o definitivamente habilitar nuevas tecnologías.

6.1.1. Aplicaciones

Un ejemplo aplicado puede ser un contact center. Las llamadas podrían ser transcritas y las conversaciones entre clientes y operadores llevadas a texto para luego ser almacenadas en un gran set de entrenamiento. Extrayendo sentimiento y entidades podríamos entrenar un modelo que aprenda de las conversaciones, sus contextos, preguntas, respuestas para modelar secuencias complejas de respuestas con niveles de confianza. Estas respuestas mantendrían

un contexto (preguntas y respuestas anteriores) atendiendo a los clientes de forma personalizada.

Este modelo sería la base para un contact center de atención inicial, automatizado y capaz de atender a muchos clientes de forma simultánea y las 24 horas del día y, utilizando neural machine translation (otra vez, podemos aprovechar transformers) dar servicios de contact center a clientes en otros idiomas.

Otra aplicación es la generación de documentos contractuales y notariales, los cuales dependen de varios contextos y utilizan una jerga muy específica, solo se requiere un buen set de datos de entrenamiento. Hay casos donde se ha entrenado un modelo transformer para la reclamación de patentes nuevas en la Oficina de Patentes de Estados Unidos [30].

Pensemos en otras aplicaciones tal vez no empresariales, pero que podrían cambiar la forma en que hacemos las cosas: ¿Qué tal una respuesta de correo automática en base a los correos anteriores y al texto del nuevo correo? Podríamos entrenar un modelo GPT-2 transformer con nuestro buzón de correo para que aprenda a responder de forma básica por nosotros.

Puede tener también un nicho en las fábricas de software. Muchas empresas han externalizado los servicios de desarrollo sencillos y cotizan a empresas (muchas veces de la India, Rusia o LatAm) piezas de código en base a requerimientos funcionales. Si tenemos acceso a ejemplos de código podemos perfectamente entrenar un modelo que confeccione un programa en base a una solicitud (y en Github incluso el código ya está comentado con lo

que hace). Ya hay aplicaciones donde un modelo genera códigos HTML y de estilos de sitios web en base a instrucciones semánticas.

Hemos entrenado un modelo que es capaz de crear texto en base a un contexto. Podríamos, utilizando el mismo método, entrenar uno que sea capaz de responder tweets o menciones de otros usuarios, un AI-Comunity Manager

Las aplicaciones no sólo se limitan a la generación de texto, otras creaciones propias del ser humano también tienen características secuenciales y cuentan con un contexto: La música. Recientemente ya se ha incorporado Transformers en modelos de secuencias musicales [9], los resultados están por verse aún.

Desafortunadamente el avance aún no permite escribir libros de tesis. ¿Pero que ocurriría si podemos darle los temas y esperar por la redacción? El caso estrella de GPT-3 es la redacción de una publicación de blog completa utilizando sólo una breve introducción.

GPT-2 es una arquitectura moderna, existen otras arquitecturas para otros objetivos, pero ésta no es la más reciente. A la fecha ya se han publicado los resultados de GPT-3, los cuales se alejan de lo que hoy conocemos como modelos de aprendizaje y se mimetizan con las capacidades humanas. A pesar de sus asombrosos resultados no es más que un escalado en tamaño, sigue manteniendo la base GPT-2.

6.1.2. Trabajo Futuro

Como trabajo futuro se recomienda probar los límites de GPT-2 en español, hemos entrenado un modelo básico con un 10% del tamaño de corpus con el que se entrenó el modelo original, obteniendo una performance levemente por debajo de este último.

Si bien es cierto hemos probado que es factible técnicamente, como un siguiente paso deberíamos comprobar que cuán distinguibles son estas generaciones con respecto a publicaciones originales. Someter un listado de publicaciones sintetizadas con este método en conjunto con publicaciones reales y consultar la opinión de usuarios de redes sociales podría permitirnos medir el nivel de mimetizado. Con un dataset etiquetado podríamos entrenar un clasificador que nos ayude a mejorar el entrenamiento de nuevas versiones de GPT-2-Político.

Bibliografía

[1] «Projects», *OpenAI*. https://openai.com/projects/ (accedido nov. 19, 2020).

[2] A. Vaswani *et al.*, «Attention is All you Need», *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 5998-6008, 2017.

[3] T. B. Brown *et al.*, «Language Models are Few-Shot Learners», *arXiv:2005.14165*, 2020.

[4] «arXiv Dataset». https://kaggle.com/Cornell-University/arxiv.

[5] «The Scalable Neural Architecture behind Alexa's Ability to Select Skills», *Amazon Science*, jun. 07, 2018. https://www.amazon.science/blog/the-scalable-neural-architecture-behind-alexas-ability-to-select-skills (accedido nov. 20, 2020).

[6] «Understanding searches better than ever before», *Google*, oct. 25, 2019. https://blog.google/products/search/search-language-understanding-bert/ (accedido nov. 20, 2020).

[7] «Recent Advances in Google Translate», *Google AI Blog*. http://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html (accedido nov. 20, 2020).

[8] «How Amazon Translate Works - Amazon Translate». https://docs.aws.amazon.com/translate/latest/dg/how-it-works.html (accedido nov. 20, 2020).

[9] «Using Transformers to create music in AWS DeepComposer Music studio», *Amazon Web Services*, nov. 18, 2020. https://aws.amazon.com/blogs/machine-learning/using-transformers-to-create-music-in-aws-deepcomposer-music-studio/ (accedido nov. 19, 2020).

[10] «List of languages by number of native speakers», *Wikipedia*. oct. 26, 2020, Accedido: nov. 10, 2020. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=List_of_languages_by_number_of_native_speakers&oldid=985620308.

[11] J. Cañete, G. Chaperon, y R. Fuentes, «Spanish Pre-Trained BERT Model and Evaluation Data», *PML4DC ICLR 2020*, [En línea]. Disponible en: https://users.dcc.uchile.cl/~jperez/papers/pml4dc2020.pdf.

[12] «huggingface/transformers», ago. 22, 2020. https://huggingface.co/transformers/ (accedido ago. 21, 2020).

[13] S. Hochreiter y J. Schmidhuber, «Long Short-Term Memory», *Neural Comput.*, vol. 9, n.º 8, pp. 1735-1780, nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[14] «Understanding LSTM Networks». https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accedido ago. 21, 2020).

[15] S. Schuchmann, «History of the first AI Winter», *Medium*, may 16, 2020. https://towardsdatascience.com/history-of-the-first-ai-winter-6f8c2186f80b (accedido nov. 08, 2020).

[16] S. Vosoughi, P. Vijayaraghavan, y D. Roy, «Tweet2Vec: Learning Tweet Embeddings Using Character-level CNN-LSTM Encoder-Decoder», *Proc. 39th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR 16*, pp. 1041-1044, 2016, doi: 10.1145/2911451.2914762.

[17] T. Mikolov, K. Chen, G. Corrado, y J. Dean, «Efficient Estimation of Word Representations in Vector Space», *arXiv:1301.3781*, sep. 2013.

[18] M.-T. Luong, H. Pham, y C. D. Manning, «Effective Approaches to Attention-based Neural Machine Translation», *arXiv:1508.04025*, sep. 2015.

[19] J. Alammar, «The Illustrated Transformer». http://jalammar.github.io/illustrated-transformer/ (accedido oct. 11, 2020).

[20] J. Devlin, M.-W. Chang, K. Lee, y K. Toutanova, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding», *arXiv:1810.04805*, may 2019.

[21] Y. Liu *et al.*, «RoBERTa: A Robustly Optimized BERT Pretraining Approach», *arXiv:1907.11692*, jul. 2019.

[22] V. Sanh, L. Debut, J. Chaumond, y T. Wolf, «DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter», *arXiv:1910.01108*, feb. 2020.

[23] A. Radford, K. Narasimhan, T. Salimans, y I. Sutskever, «Improving Language Understanding by Generative Pre-Training», *OpenAI*, jun. 2018, [En línea]. Disponible en: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, y I. Sutskever, «Language Models are Unsupervised Multitask Learners», *OpenAI*, 2019, [En línea]. Disponible en: https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[25] G. Lample y A. Conneau, «Cross-lingual Language Model Pretraining», *arXiv:1901.07291*, ene. 2019.

[26] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, y R. Socher, «CTRL: A Conditional Transformer Language Model for Controllable Generation», *arXiv:1909.05858*, sep. 2019.

[27] «Summary of the models — transformers 3.4.0 documentation». https://huggingface.co/transformers/model_summary.html (accedido nov. 09, 2020).

[28] R. Ressmeyer, S. Masling, y M. Liao, «"Deep Faking" Political Twitter using Transfer learning and GPT-2», *Stanford*, 2019, [En línea]. Disponible en: http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26647402.pdf.

[29] A. Kayal, «Train GPT-2 in your own language», *Medium*, sep. 24, 2020. https://towardsdatascience.com/train-gpt-2-in-your-own-language-fc6ad4d60171 (accedido oct. 15, 2020).

[30] J.-S. Lee y J. Hsiang, «Patent Claim Generation by Fine-Tuning OpenAI GPT-2», *arXiv:1907.02052*, ene. 2019.

[31] A. Wang y K. Cho, «BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model», *arXiv:1902.04094*, feb. 2019.

[32] «Políticos de Chile», *Wikipedia*. Accedido: jul. 10, 2020. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Categor%C3%ADa:Pol%C3%ADticos_de_Chile&oldid=64591553.

[33] «GET users/search». https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-users-search (accedido nov. 11, 2020).

[34] «GET statuses/user_timeline». https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/api-reference/get-statuses-user_timeline (accedido nov. 11, 2020).

[35] «Wikimedia Downloads». https://dumps.wikimedia.org/ (accedido nov. 11, 2020).

[36] D. Alexandre, «How to build a controllable writing assistant for novel authors», *Medium*, oct. 24, 2020. https://towardsdatascience.com/how-to-build-a-controllable-writing-assistant-for-novel-authors-a9fa15b57c6a (accedido nov. 06, 2020).