
Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales

Maestría en Explotación de Datos y Descubrimiento de Conocimiento

# Tesis de maestría

**presentada para optar al título de**

## Magíster en Explotación de Datos y Descubrimiento de Conocimiento

## Generación de resúmenes de texto en español

por
Leonardo Ignacio Córdoba

Director: Leandro Ezequiel Lombardi

17 de Agosto, 2022
Buenos Aires

**Resumen**

En los últimos años han habido grandes avances en las tareas de generación de texto. Especialmente, el desarrollo y generalización de los modelos preentrenados permite el entrenamiento de modelos para tareas especializadas a bajo costo computacional. Por otro lado, el resumen abstracto de texto es una tarea para la cual son relativamente pocos los conjuntos de datos disponibles en español. Por este motivo, este trabajo se enfoca en el entrenamiento de modelos de aprendizaje automático para resumen en español y la construcción de conjuntos de datos apropiados para esta tarea. Como parte del mismo se disponibilizan 5 modelos entrenados a partir de arquitecturas basadas en Transformers y 2 conjuntos de datos obtenidos de Common Crawl. Además, se comparan métricas calculadas automáticamente con puntajes obtenidos mediante etiquetado manual.

**Palabras claves:** Resumen abstracto, generación de texto, procesamiento del lenguaje natural.

# Generation of text summaries in Spanish

**Abstract**

In the last few years there has been several advances in text generation tasks. Especially, the development and generalization of pretrained models allows the training of models for specialized tasks -a.k.a. downstream tasks- at low computational cost. On the other hand, abstract text summarization is a task for which relatively few data sets are available in Spanish. For this reason, this work focuses on the training of machine learning models for summarization in Spanish and the construction of appropriate datasets for this task. As part of it, 5 models trained from architectures based on Transformers and 2 data sets obtained from Common Crawl are available. Additionally, automatically calculated metrics are compared to scores obtained through manual tagging.

**Keywords:** Abstract Summary, Text Generation, Transformers, Common Crawl.

**Agradecimientos**

A *Fran Gaska*, por haberme alentado a comenzar este camino, ser mi compañera durante el mismo y apoyarme para concluirlo.

A *Tomi*, por hacer que cada día sea único e inspirarme a ser una mejor persona.

A mis viejos, *Raúl* y *Ro*, por ser mis guías en la vida.

A *Leandro Lombardi* por haber dirigido este trabajo y enriquecerme con sus reflexiones en el proceso.

A *Diego Kozlowski* y a *Juan Manuel Barriola*, por haber compartido incontables horas aprendiendo juntos.

A *Marcelo Soria*, por la buena predisposición que siempre ha tenido, dentro y fuera del ámbito de la maestría.

A mis amigos, especialmente a *Nicolás Monner Sans* y *Olga Khamidulina* por su apoyo y cariño.

A todas aquellas personas que hacen de la Universidad de Buenos Aires un mejor lugar.

# Índice

## Índice de figuras

## Índice de cuadros

# 1 Introducción

En esta sección se comienza introduciendo el área dentro del procesamiento del lenguaje natural (PLN) al cual se aboca este trabajo. Luego, se define el objetivo principal del mismo y se continúa dando un marco general de los enfoques que se desarrollan posteriormente. Por último, se comenta la organización del resto del trabajo.

## 1.1 Modelos de secuencia a secuencia

En el campo del procesamiento del lenguaje natural (PLN) existen distintas formas de estructurar la información para modelos de aprendizaje automático. En este trabajo nos interesan especialmente los llamados modelos de secuencia a secuencia (abreviados Seq2Seq en inglés). Éstos se caracterizan por recibir como entrada una secuencia de texto y producir como salida una nueva secuencia de texto.

Estos modelos son capaces de resolver diversos problemas, algunos ejemplos son:

- Modelado del lenguaje: esta es la tarea más conocida, consistente en predecir el próximo token (palabra, parte de palabra u otro tipo de símbolo) en un documento. Los modelos de lenguaje en ocasiones son entrenados para generar texto de cualquier dominio y, en otros casos, estos modelos son especializados en alguna área en particular.
- Traducción: en esta tarea el modelo recibe como entrada una frase y genera la misma frase en otro idioma.

- Resumen (summarization): puede ser abstracto, consistente en generar un texto nuevo que resuma un texto más largo, o resumen extractivo, cuando se resume un texto a partir de una porción de ese mismo texto.
- Simplificación: en este caso el texto original es simplificado, por ejemplo, omitiendo términos poco importantes, aclarando o ejemplificando términos inusuales, cambiando estructuras sintácticas, etc.

Si bien han habido distintas maneras de modelar lenguaje natural, en la actualidad los métodos con mejores resultados se enmarcan en una familia de modelos llamados *transformers* [58]. Recientemente este campo de investigación ha suscitado gran interés, en particular por los sorprendentes avances logrados para generar texto libre a partir de GPT-2 [45].

Por lo general, existen dos instancias en la aplicación de modelos tan grandes como éstos. En una primera instancia el modelo es **preentrenado**, en esta etapa el modelo ajusta sus parámetros de modo de ubicarse en algún lugar del espacio de parámetros que se estima cercano al objetivo que tiene después y, por lo tanto, es la tarea más costosa. Para poder aplicar el modelo en una tarea en particular es necesario continuar con el entrenamiento pero adaptando la salida del modelo para el caso, el llamado **ajuste fino** del modelo.

La gran mayoría de estos desarrollos suelen publicarse acompañados con el código para reproducir los experimentos y con los modelos ya entrenados. Sin embargo, gran parte estos esfuerzos están orientados al inglés y, en menor

número, a otros idiomas.

### 1.2 Objetivo

A partir de lo dicho anteriormente se proponen tres objetivos a desarrollar.

En primer lugar, se desea investigar y desarrollar modelos de resumen abstracto de texto en español, a partir de los avances más recientes que existen en el área. Se selecciona esta tarea ya que tiene diversas ramificaciones y existen conjuntos de datos disponibles. Por ejemplo, sobre una noticia de un periódico se pueden generar resúmenes largos, cortos o, incluso, se pueden considerar como tareas de resumen a la generación de títulos o copetes.

Para llevar a cabo esta tarea se comienza investigando los métodos recientes de resumen abstracto y se definen los enfoques y la metodología a seguir.

En segundo lugar, se propone analizar los resultados de los modelos empleando tanto métricas calculadas automáticamente como métricas basadas en anotaciones humanas, ya que ésto permite inspeccionar los resultados en dimensiones que no serían posibles de otra manera. Por este motivo se recopilan otros trabajos donde se realizan análisis similares para poder definir una metodología de evaluación, especialmente en lo que respecta a la evaluación mediante etiquetado manual.

En tercer lugar, este trabajo tiene como objetivo disponibilizar los modelos y conjuntos de datos creados como parte del mismo. Ésto tiene por fin poder ayudar a la difusión de estas técnicas en idioma español. Para ello, se propone revisar los canales actuales para divulgar estos recursos y com-

patibilizar los mismos al formato requerido. Como parte de este objetivo también se incluirá código donde se muestre cómo acceder o utilizar a estos recursos.

### 1.3 Síntesis de la metodología

Para el entrenamiento de los modelos de resumen se proponen dos enfoques. En primer lugar, una posibilidad es preentrenar un modelo en español y luego realizar un ajuste fino a la tarea de resumen.

Considerando el enorme costo que tiene el preentrenamiento de estos modelos [53] y la poca disponibilidad de modelos en español, se vuelve relevante encontrar formas de reaprovechar los modelos existentes, para evitar el entrenamiento desde cero. Por este motivo, en este trabajo se propone adaptar el modelo Beto [9] para la generación de texto. Para ello, se inicializa un modelo Encoder-Decoder [50] con el conjunto de parámetros de Beto tanto en el encoder como en el decoder y luego se entrena este modelo de lenguaje sobre un conjunto de datos construído para este fin. Posteriormente, se propone especializar al modelo en la tarea de resumen abstracto.

Para entrenar este modelo se construye un conjunto de datos a partir de noticias disponibles en Common Crawl.

En segundo lugar, otro enfoque posible es emplear un modelo multilenguaje ya preentrenado (que incluya el idioma español) y especializarlo en la tarea mencionada. Actualmente el modelo más grande con estas características es MT5 [64].

Se proponen dos tareas de resumen abstracto. En primer lugar, se toma como benchmark la parte en español

del conjunto de datos de resumen abstracto MLSUM [52]. Este conjunto de datos cuenta con noticias y sus resúmenes obtenidos del diario El País de España.

En segundo lugar, se construye un conjunto de datos de noticias y títulos a partir de Common Crawl. En este caso, el resumen que el modelo aprende a generar es el título de la noticia.

Al finalizar este trabajo se espera poder disponibilizar públicamente:

- Modelo de lenguaje EncoderDecoder preentrenado en noticias
- Modelo EncoderDecoder especializado en resúmenes de noticias
- Modelo EncoderDecoder especializado en títulos de noticias
- Modelo MT5 especializado en resúmenes de noticias
- Modelo MT5 especializado en títulos de noticias
- Conjunto de datos 1: noticias de Common Crawl para entrenar el modelo de lenguaje
- Conjunto de datos 2: pares de noticias y títulos para entrenar modelo de resumen

### 1.4 Organización de la tesis

El resto de la tesis está organizada como se indica a continuación.

En la segunda sección se desarrollan las ideas básicas de las técnicas de PLN recientes. Se comienza revisando lo que

es un modelo de lenguaje probablístico y embeddings. Luego se relaciona ésto con las arquitecturas de redes recurrentes, el mecanismo de atención y, por último, transformers.

En la tercera sección se ahonda en las arquitecturas de transformers y la diferencia entre preentrenar modelos y hacer realizar ajustes finos sobre una tarea específica. En especial, la sección se enfoca en BERT, en T5 y en cómo reaprovechar modelos preentrenados para el problema de la tesis.

En la cuarta sección se explican los aspectos metodológicos del trabajo. En particular, se refiere a qué modelos se entrenan y cómo. A su vez, se detallan los conjuntos de datos empleados, algunos de los cuales han sido creados para este trabajo y otro es un benchmark de la tarea de resumen abstracto.

En la quinta sección se describen las métricas que se emplean en el trabajo y por qué no es suficiente emplear una evaluación automática para medir la calidad de los modelos.

En la sexta sección se describen los resultados obtenidos. Para ello se comienza mostrando los resultados obtenidos automáticamente para cada uno de los modelso y se los compara con las referencias disponibles. Además, se analizan los resultados obtenidos mediante anotaciones manuales. Posteriormente se ejemplifica con textos generados por los modelos y se los compara con los equivalentes humanos. Finalmente, se describe cómo se abren los modelos y datos a través de la plataforma HuggingFace.

En la séptima sección se plantean las conclusiones y líneas de investigación futuras.

# 2 Marco teórico

En la presente sección se comienza explicando qué es un modelo de lenguaje y cómo se puede entrenar un modelo de lenguaje usando redes neuronales. De este modo, nos encontramos con el concepto de *embedding*. En la segunda subsección extendemos la explicación sobre embeddings y comentamos cómo se relaciona con las redes recurrentes, especialmente en lo que respecta a la generación de texto. Finalmente, en la tercera subsección llegamos al llamado *mecanismo de atención* y, especialmente, introducimos la arquitectura conocida como *transformers*.

## 2.1 Modelo probabilístico de lenguaje

La generación de resúmenes se enmarca en la tarea más amplia conocida como generación de texto o modelos de lenguaje.

Podemos definir un modelo probabilístico de lenguaje como la probabilidad de la siguiente palabra dadas las anteriores:

$$\hat{P}({w_1}^T) = \prod_{t=1}^{T} \hat{P}(w_t|{w_1}^{t-1}) \tag{2.1}$$

donde $w_t$ es la t-ésima palabra y ${w_i}^j = (w_i, w_{i+1}, ..., w_{j-1}, w_j)$

Las palabras en una oración ocupan una posición que es relevante para poder modelar la probabilidad de la próxima palabra. Teniendo en cuenta ésto y, debido a la necesidad de reducir la complejidad del problema, surgen los modelos de n-gramas, en el cual se reduce el contexto a las últimas n - 1

palabras y se construyen tablas de probabilidad condicional con sus combinaciones. Entonces, en este caso definimos:

$$\hat{P}(w_t|w_1^{t-1}) \approx \hat{P}(w_t|w_{t-n+1}^{t-1}) \tag{2.2}$$

Estos modelos encuentran dos grandes limitaciones. La primera es que el contexto es muy acotado, generalmente las últimas 2 o 3 palabras. Y en segundo lugar, la capacidad de generalización se ve reducida por no tomar en cuenta la similaridad entre las palabras. En este sentido, la frase "La economía argentina no para de crecer"debería ayudarnos a completar, por ejemplo, la frase "La economía chilena ...", ya que *argentina* y *chilena* se encuentran cerca de *economía* y juegan un rol similar [3].

Entre los intentos de sortear este segundo problema podemos mencionar el agrupamiento de palabras en clusters según si son parecidas o no en algún sentido, como por ejemplo, empleando n-gramas de clases de palabras [7].

En [3] se plantea un enfoque distinto y que inspira a posteriores trabajos en el área. Allí se asigna cada palabra a un vector en $\mathbb{R}^m$ para luego estimar la probabilidad conjunta de las secuencias de palabras en función de sus respectivos vectores. La representación de símbolos en un espacio vectorial en el contexto de redes neuronales también se puede encontrar en otros trabajos de la época pero en otros usos, como en la predicción de la estructura secundaria de proteínas [49] o en problemas de texto a fonema [25].

Dos elementos importantes a remarcar son:

- cada palabra queda asignada como un punto en el espacio vectorial

- la dimensión de dicho espacio es mucho menor que el tamaño del vocabulario

Estas dos condiciones permiten ajustar una red neuronal para estimar la función de probabilidad conjunta de un cierto número de términos previos para predecir el siguiente. Además, es interesante mencionar que en ese mismo trabajo se plantea que si bien los parámetros de la red son inicializados al azar podrían ser inicializados usando valores ya aprendidos anteriormente.

El modelo es la función $f$ definida como:

$$f\left(w_t, \cdots, w_{t-n+1}\right) = \hat{P}\left(w_t | w_1^{t-1}\right) \tag{2.3}$$

Con la restricción de que las estimaciones sean probabilidades:

$$\sum_{i=1}^{|V|} f\left(i, w_{t-1}, \cdots, w_{t-n+1}\right) = 1 \tag{2.4}$$

donde $V$ es el vocabulario.

En la práctica, la ecuación 2.3 se descompone en dos partes:

1. Una asignación desde una palabra hasta un vector mediante la matriz $C$. Esta matriz tiene parámetros libres con dimensión $|V|.m$ .

2. A continuación, una función $g$ asigna la secuencia de $\left(C\left(w_{t-n+1}\right), \cdots, C\left(w_{t-1}\right)\right)$ a una distribución de probabilidades condicionales sobre la próxima palabra da-

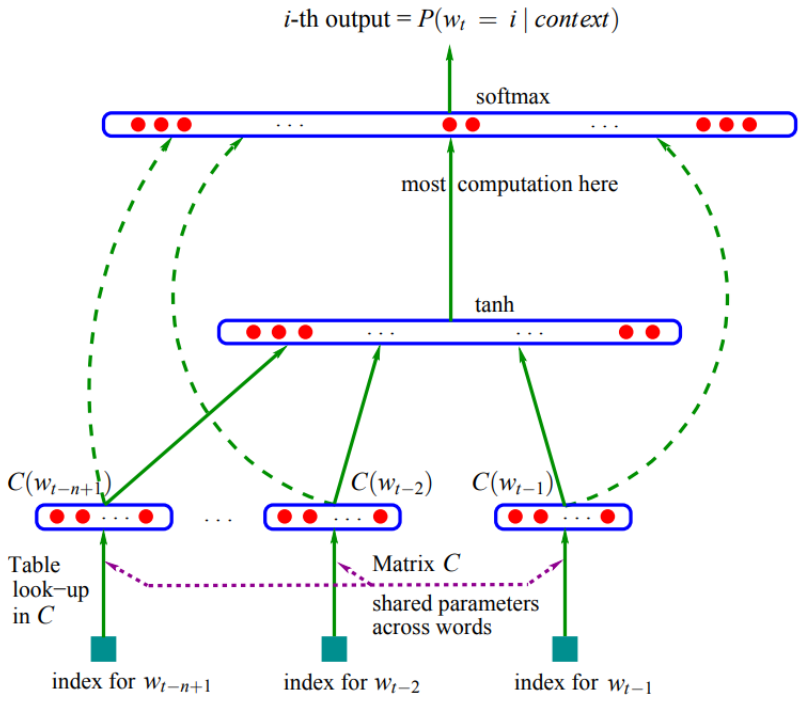


Figura 1: Arquitectura del *Modelo de lenguaje neuronal probabilístico*

das las anteriores, es decir:

$$f\left(i, w_{t-1}, \cdots, w_{t-n+1}\right) = g\left(i, C\left(w_{t-1}\right), \cdots, C\left(w_{t-n+1}\right)\right) \tag{2.5}$$

Esa función $g$ no es más que la red neuronal. La arquitectura de este modelo se puede ver en la figura 1 [3]. Nótese que los parámetros de $C$ son compartidos para cada palabra.

Los parámetros $\theta$ se entrenan maximizando la función de log-verosimilitud promedio en el corpus de entrenamiento y agregando un factor de penalización $R$.

$$L = \frac{1}{T} \sum_{t} \log f\left(w_t, w_{t-1}, \cdots, w_{t-n+1}; \theta\right) + R(\theta) \tag{2.6}$$

El modelo toma una ventana fija de palabras y trata de predecir la siguiente. La primera capa (la proyección) no presenta no linealidades y es conocido como *embedding* del modelo, mientras que la segunda capa es la capa oculta y emplea funciones de activación de tipo tangente hiperbólica. En cuanto a la capa de salida, usa una función *softmax*. La función *softmax* es elegida ya que genera como salida un vector de probabilidades, es decir, que suma 1 y que no presenta valores negativos. De esta manera, la salida del modelo es:

$$\hat{P}\left(w_t | w_{t-1}, \cdots w_{t-n+1}\right) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{2.7}$$

Se puede ver en la ecuación 2.7 cómo se normaliza el valor de $y$. Por su parte, $y$ queda definido como:

$$y = b + Wx + U \tanh(d + Hx) \tag{2.8}$$

y donde $x$ es:

$$x = \left(C\left(w_{t-1}\right), C\left(w_{t-2}\right), \cdots, C\left(w_{t-n+1}\right)\right) \tag{2.9}$$

Siendo $b, W, U, d, H, C$ los parámetros del modelo.

La idea de aplicar redes neuronales al modelado de lenguaje se extiende desde entonces y muchos de los siguientes avances en los modelos de lenguaje se relacionaron a:

- Reaprovechar la información contextual de las palabras mediante el uso de *embeddings*.
- Extender la ventana de palabras que considera o 'recuerda' el modelo.

### 2.2 Embeddings y redes recurrentes

La idea de entrenar un *embedding* como forma de representar una palabra en un espacio vectorial 'manejable' cobra mucha fuerza, especialmente con el desarrollo de modelos entrenados sobre corpus muchísimos más grandes que los que se usaban previamente. El primero de éstos modelos, Word2Vec [38], es entrenado sobre "Google News". A este modelo le siguen Glove, entrenado sobre Wikipedia, Gigaword, Twitter y CommonCrawl [43] y FastText, entrenado en Wikipedia y CommonCrawl [4]. En este último caso, vale agregar que el modelo no está entrenado sobre palabras enteras sino sobre partes de palabras, lo cual permite capturar nueva información. Una descripción detallada de estos modelos se puede encontrar en el Apéndice 8.1.

Algunos avances algorítmicos que permiten el entrenamiento de embeddings son el uso de softmax jerárquico [40] en vez de la función softmax, la actualización parcial de pesos mediante el muestreo negativo y el muestreo de palabras en relación inversa a la frecuencia [39].

Dada una palabra, su embedding captura la información contextual, es decir, qué palabras tienden a acompañarle. Se puede demostrar [29] que, por ejemplo, el skip-gram Word2Vec con negative sampling factoriza una matriz de PMI [12] (corrida por una constante global) :

$$M_{ij}^{\text{SGNS}} = W_i \cdot C_j = \vec{w}_i \cdot \vec{c}_j = PMI\left(w_i, c_j\right) - \log k \quad (2.10)$$

En [39] se plantea una nueva tarea para evaluar los embeddings, consistente en tratar de adivinar la mejor analogía para distintos tipos de relación bajo la pregunta "¿qué

palabra es similar a C, en el mismo sentido (o en la misma proporción) en que lo son A y B?". Por ejemplo, "¿qué palabra es similar a Berlin en el mismo sentido en que Francia y Paris son similares? La respuesta esperada en este caso es Alemania.

Para responder esta pregunta se pueden emplear los vectores del embedding haciendo la siguiente operación:

$$X = vector(biggest) - vector(big) + vector(small) \quad (2.11)$$

El segundo punto mencionado anteriormente se refiere a cómo mantener la dependencia de largo plazo en los modelos de lenguaje. En [3] este problema es especialmente importante porque el modelo toma como entrada una ventana fija. Un gran avance frente a ésto se produjo con el desarrollo de redes neuronales recurrentes [32], especialmente las LSTM [22] y GRU [11]. Esta clase de redes se caracteriza por tomar como entrada información secuencial y por mantener una cierta memoria de la información anterior. De esta manera, los datos previos pueden afectar a las predicciones del modelo.

Sin extendernos demasiado, esta información puede pasar de izquierda a derecha siguiendo un orden temporal, por ejemplo, de la misma manera que se lee una frase en español, o bien puede ser bidireccional, es decir, en el orden temporal y en el inverso. Así, pueden existir redes Bi-LSTM o Bi-GRU.

A diferencia de los embeddings, las redes recurrentes permiten aprender relaciones contextuales en base a la posición, ya que comprimen la información de las palabras an-

teriores a una palabra determinada. Si además la red es bidireccional, entonces toma tanto información previa como posterior. A este proceso por el cual comprimen la información se lo conoce como encoding. El paso siguiente es generar un resultado, a ello se lo conoce como decoding. En la figura 2 se puede ver una arquitectura simplificada de un ejemplo de traducción con una red recurrente. Puede apreciarse que cada token de la entrada ingresa de a uno por vez a la red, el primer bloque arriba del token es la representación vectorial (word vector) y en el segundo nivel se encuentra la capa oculta (o hidden state). Se puede ver, también, que las capas ocultas a partir del primer token toman como entrada la salida de la capa oculta anterior. El encoder finaliza con el token *eos* (end of sentence), a partir de allí el decoder comienza a generar las salidas (arriba de los bloques amarillos), cada uno es a su vez la entrada de la etapa siguiente hasta generar otro *eos*.

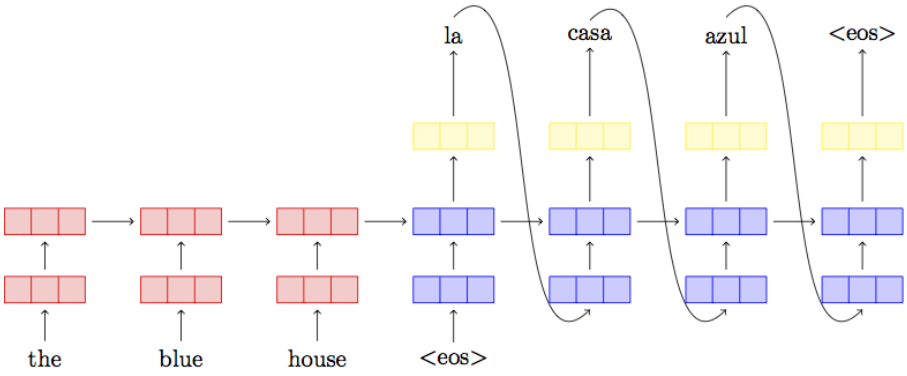


Figura 2: Ejemplo de red recurrente. En rojo el encoder, en azul el decoder.

La naturaleza secuencial de estas redes hace que, por un

lado, sea posible recordar información previa trasladándola, con lo cual han sido ampliamente empleadas en tareas de generación de texto, como traducción [10] [55]. Sin embargo, también limita qué tan lejana puede ser esa información, ya que para trasladar la información previa es necesario realizar muchísimas operaciones en las cuales se puede caer en el problema del "desvanecimiento del gradiente". Un segundo limitante que se deriva de la naturaleza secuencial es que el entrenamiento no es completamente paralelizable, lo cual lo vuelve lento y poco escalable.

### 2.3 El mecanismo de atención y transformers

Uno de los grandes avances para extender y mejorar la precisión de la "memoria" de las redes recurrentes es el mecanismo de atención [2] [63]. Este mecanismo lo que hace es permitir asignarle un peso relativo a cada uno de las entradas de los T períodos previos.

Intuitivamente podemos interpretar que el decoder puede darle mayor o menos importancia a distintas palabras de la secuencia ingresada, en cierta medida disminuyendo la necesidad de compresión del encoder. Para ello se pesa cada embedding contextual $h$ por el peso $\alpha$, donde éste es el resultado de una *softmax*:

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \tag{2.12}$$

donde

$$e_{ij} = a\left(s_{i-1}, h_j\right) \tag{2.13}$$

es un modelo de alineación que asigna un valor basado en el estado oculto previo $s_{i-1}$ y el embedding en esa posición

$h_j$. Gráficamente el mecanismo se puede describir como en la figura 3.

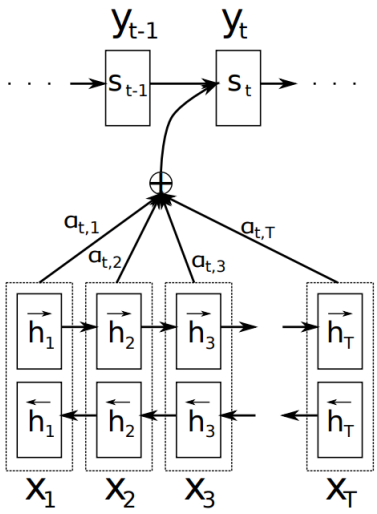


Figura 3: Ejemplo de mecanismo de atención, tomado de [2]

Esta idea es la base de un gran cambio en la arquitectura de los modelos, especialmente a partir de [58], cuando se introduce la arquitectura conocida como Transformer. En ella se elimina la recurrencia en las redes (y por lo tanto, la secuencialidad en la computación) y se basa la red exclusivamente en mecanismos de atención. La arquitectura propuesta es la de la figura 4 que detallamos a continuación.

La información de entrada se incorpora toda simultáneamente a la red y no secuencialmente como en las redes recurrentes. Lo mismo es cierto para la salida (hasta la posición actual). Ésto introduce un primer problema que, aunque no es central en el modelo, sí es relevante mencionar. Dado que el proceso no es autorregresivo el modelo requiere alguna manera para identificar la posición de los tokens y, por lo

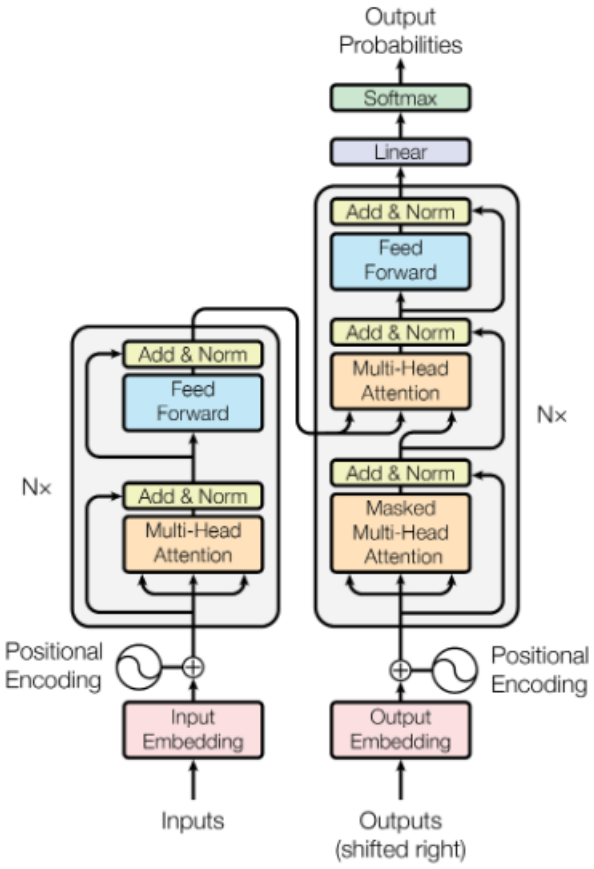


Figura 4: Transformers original, tomado de [58]

tanto, aprender una representación para cada token según su posición. Como vemos en la imagen, la información pasa por los embeddings y luego por un **encoding posicional**. Este encoding posicional viene justamente a resolver el problema. Sea *pos* la posición del token e $i$ la dimensión, se define el **Positional Encoding** ($PE$) como:

$$
\begin{aligned}
PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \\
PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\text{model}}}\right)
\end{aligned}
\tag{2.14}
$$

Luego del encoding posicional vemos que la información pasa por los mecanismos de atención, que son la clave de la arquitectura. Una función de atención puede ser descripta como una relación entre una *query* y un conjunto *llave-valor* (*key-value*) a una salida. La *query* es la pregunta, que codifica qué información a recuperar. El conjunto *llave-valor* es una forma de almacenar información, en la cual la llave indica la posición y el valor guarda la información de interés.

El mecanismo de atención empleado en Transformers se conoce como **Scaled Dot-Product Attention**:

$$
\text{Attention } (Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.15}
$$

Donde $Q$ es una matriz de queries, $K$ una matriz de keys y $V$ una matriz de valores.

Como se ve en la ecuación 2.15 y en la figura 5, el primer paso en el cálculo es el producto punto entre $Q$ y $K$. Esto es similar a la distancia coseno entre $Q$ y $K$, con lo que tendremos valores más altos allí donde los vectores sean más similares:

$$
\text{similaridad } = \cos(\theta) = \frac{\mathbf{Q} \cdot \mathbf{K}}{\|\mathbf{Q}\|\|\mathbf{K}\|} \tag{2.16}
$$

Luego, dividimos por $\sqrt{d_k}$, siendo $d_k$ la dimensión de las *queries* y *keys*. Este escalamiento se hace para evitar que

los valores caigan en rangos extremos de la *softmax*. Por último, la *softmax* nos da una distribución sobre los *values* $V$, pudiendo de esa manera recuperar información de la *memoria* del modelo.

Es preciso aclarar que en *Transformers* se usan capas de atención con múltiples cabezas (o *multi-head attention layers*). Estas capas no son más que la concatenación de muchas capas de atención:

$$\begin{aligned} \text{MultiHead}\,(Q,K,V) &= \text{Concat}\,(\text{head}_1,\ldots,\text{head}_h)\,W^O \\ \text{where head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2.17}$$

En la versión original de este trabajo usan 8 capas de atención ($h = 8$) y $d_k = d_v = d_{\text{model}}/h = 64$

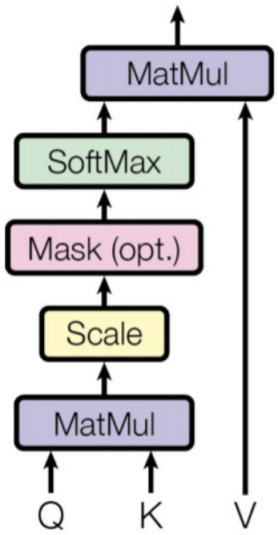


Figura 5: Mecanismo de atención, tomado de [58]

El modelo emplea los mecanismos de atención de tres maneras:

- En el encoder (lado izquierda de la arquitectura) se emplea una capa de autoatención (*self-attention layer*). Esto quiere decir que toda la información proviene del mismo lugar, el paso anterior en el modelo.
- En el decoder también se emplea una capa de autoatención pero enmascarada (*masked multi-head attention*). Ésto se debe a que toma la información del texto objetivo *hasta e incluyendo* esa posición.
- Por último, el modelo cuenta con una capa *encoder-decoder*, en la que las *keys* y los *values* provienen del *encoder* mientras que las *queries* provienen del decoder.

Posteriormente han habido diferentes variantes de esta arquitectura, implementando mejoras para aplicar el modelo en imágenes, para mejorar la performance en secuencias más largas, estabilizar el entrenamiento de modelos de reinforcement learning, entre otras tareas [61].

# 3 Modelos preentrenados y ajuste fino

Modelos como ULMFit [23], BERT[15] o GPT-2[45] dividen el entrenamiento en dos etapas. A la tarea de entrenar un modelo en un conjunto de datos extremadamente grande de manera no supervisada se la conoce como preentrenamiento. En esta etapa se espera que el modelo aprenda representaciones generales del lenguaje que sean de utilidad para más de una tarea específica. Luego, este modelo preentrenado puede ser adaptado en tareas específicas (a veces llamadas *downstream tasks* en inglés) que, generalmente, requiere un cambio menor en la arquitectura del modelo, a ésto se lo conoce como ajuste fino (o fine-tuning en inglés). Si bien existen distintas técnicas para realizar un ajuste fino, la idea principal es reutilizar el conocimiento existente en los parámetros aprendidos previamente.

El surgimiento y extensión de este tipo de modelos genera un cambio de paradigma en el mundo de la Inteligencia Artificial. Por este motivo, en [5] se los conoce como **modelos fundacionales**. Allí se resume la relevancia de estos modelos en dos palabras: emergencia y homogeneización. La emergencia implica que el aprendizaje de estos modelos surge implícitamente de los patrones de datos, antes que explícitamente. Esto hace referencia a que durante el preentrenamiento no se emplea un objetivo supervisado sino semi-supervisado, por ejemplo, predecir la siguiente palabra en un modelo de lenguaje. En cuanto a la homogeneización, en este caso el término capta la consolidación de metodologías para construir modelos de Machine Learning en una gran variedad de aplicaciones.

Esta sección comienza describiendo el modelo conocido como BERT. En segundo lugar, se estudian técnicas para reaprovechar modelos preentrenados, lo cual se toma como base para uno de los enfoques para generar resúmenes implementados en este trabajo. En la subsección siguiente describimos el modelo T5, el cual cuenta con una versión multilenguaje que empleamos en un segundo enfoque para generar resúmenes.

### 3.1 BERT

GPT-2 toma el *decoder* de Transformers, con lo cual usa el método de preentrenamiento "natural"para esta arquitectura, es decir, tratar de predecir la siguiente palabra.

En BERT se implementa un modelo basado prácticamente sólo en bloques de **encoder** de Transformers, difiere levemente ya que en vez de usar activaciones ReLU usa activaciones GELU. Para poder lograr un entrenamiento bidireccional (como en [44]) BERT se preentrena con nuevas tareas de no supervisadas:

- Masked Language Model (MLM): en vez de predecir la siguiente palabra, en esta tarea se enmascaran (se ocultan) al azar una serie de tokens (el 15 % en BERT), reemplazándolos por el token [MASK]. A este tipo de tarea también se le llama *cloze task*. Ahora bien, en la tarea de ajuste fino el token [MASK] no aparece, con lo cual se está introduciendo un sesgo. Para mitigar este sesgo, al 80 % de los tokens elegidos se los reemplaza por [MASK], a un 10 % por un token al azar, y a otro 10 % se lo deja sin intercambiar. Finalmente, la tarea

consiste en predecir ese token. Por ejemplo, dada la frase 'El gato está [MASK] la caja', el modelo debería predecir un vector con la probabilidad de corresponder a la máscara de cada palabra.

- Next Sencence Prediction (NSP): ciertas tareas se basan en entender relaciones entre dos frases, lo cual no está capturado en una tarea de modelado de lenguaje, por ejemplo, en las tareas de QA y Natural Language Inference (NLI). Por este motivo, los autores crean la tarea NSP que consiste en tomar al azar dos frases, en la mitad de los casos la segunda frase es la frase que le sigue en el texto original. En la otra mitad de los casos la segunda frase es una frase de otro lugar del testo. El objetivo de la tarea es predecir la segunda frase le sigue o no le sigue a la primera.

El preentrenamiento se realiza sobre BooksCorpus (800 millones de palabras) y sobre English Wikipedia (2.500 millones de palabras).

Por otro lado, BERT es capaz tanto de representar una sola frase como pares de frases en una sola secuencia. Para hacerlo realiza las siguientes representaciones:

- El primer token de cada secuencia es [CLS].
- Si hay más de una las frases se separan con [SEP].
- Se agrega un embedding identificando si un token pertenece a la frase A o a la frase B.

Así, dado un token, su representación se construye sumando el embedding del token correspondiente, el embed-

ding que identifica el segmento (A o B) y el embedding de la posición.

Una de las ventajas de BERT es que exceptuando la última capa el resto de la arquitectura (y por lo tanto de los parámetros) del preentrenamiento se reaprovechan a la hora de hacer un ajuste fino del modelo a una tarea específica, lo que puede verse en la Figura 6. Vale la pena aclarar que la entrada es de 512 tokens.

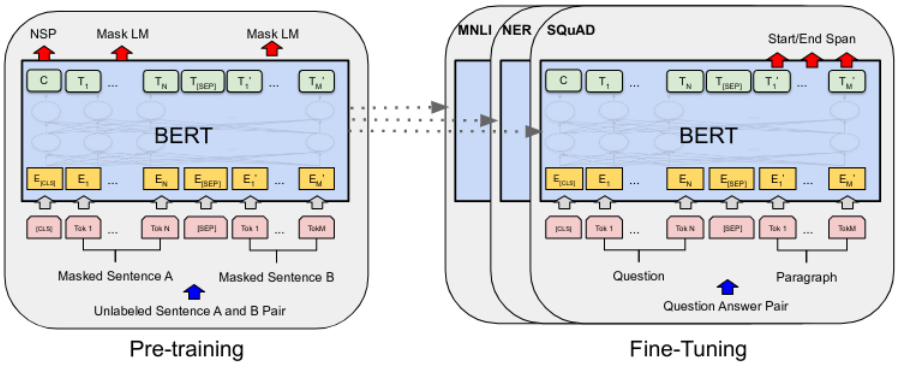


Figura 6: Preentrenamiento y ajuste fino, tomado de [15]

Además, los autores muestran que haciendo un ajuste fino de BERT se alcanzan resultados *state-of-the-art* en distintas tareas. El proceso de ajuste fino consiste sencillamente en pasar la información de entrada y salida para una tarea específica y entrenar todos los parámetros con ello, en algunos casos es necesario adaptar la última capa para que corresponda al formato de salida que espera. Sin embargo, este proceso es relativamente poco costoso.

Además de realizar ajuste fino en [15] también se muestra que es posible alcanzar resultados muy cercanos al *state-of-the-art* empleando alguna o algunas de las capas de BERT

como embeddings en un modelo Bi-LSTM.

Con los grandes resultados obtenidos por Transformers en general y por BERT en particular, surgen otras líneas de trabajo como:

- Análisis del funcionamiento interno de BERT [57] [37] [13].
- Modelos basados en BERT entrenados en otros idiomas [36] [51], entre los cuales nos interesa destacar Beto [9].
- Mejoras sobre la versión original de BERT, cambiando hiperparámetros y métodos de entrenamiento [34] o reduciendo la cantidad de parámetros requeridos y aumentando la velocidad de entrenamiento [28].

### 3.2 Reaprovechamiento de modelos preentrenados

Dado que entrenar este tipo de modelos es muy costoso pero, al mismo tiempo, los parámetros ya aprendidos encierran información muy valiosa, en [50] se propone una arquitectura con encoder y decoder similar al Transformer original que es inicializada con parámetros preentrenados en modelos basados en BERT, RoBERTa y GPT-2, que pueda seguir siendo preentrenada para tareas de generación de texto.

A diferencia de BERT esta arquitectura se adapta bien a problemas de secuencia a secuencia (seq2seq), como traducción y resumen abstracto.

En los casos en que se usa BERT como decoder, los parámetros faltantes (ya que BERT hereda la arquitectura del encoder de Transformer, no del decoder) son inicializados de manera aleatoria.

Los autores realizan más de 300 experimentos donde prueban distintas combinaciones para inicializar los parámetros. Como encoder prueban:

- Valores aleatorios
- BERT
- RoBERTa

Como decoder prueban:

- Valores aleatorios
- BERT
- GPT-2

Además, se prueban dos modelos donde se comparten los valores de encoder y decoder, BERTSHARE y RoBERTaSHARE. Por último, también se prueba una arquitectura puramente decoder inicializada con GPT-2. Todas las variantes se pueden ver en la Figura 7.

En los experimentos realizados se incluyen las siguientes tareas:

- Sentence Fusion: esta tarea consiste en agrupar varias frases en una sola frase coherente.
- Split and rephrase: esta es la tarea contraria a la anterior, consiste en separar coherentemente una frase en dos o más nuevas frases más cortas.
- Machine translation: en este caso la tarea consiste en traducir una frase de un idioma a otro.

| | total | embed. | init. | random |
|---|---|---|---|---|
| RND2RND | 221M | 23M | 0 | 221M |
| BERT2RND | 221M | 23M | 109M | 112M |
| RND2BERT | 221M | 23M | 109M | 26M |
| BERT2BERT | 221M | 23M | 195M | 26M |
| BERTSHARE | 136M | 23M | 109M | 26M |
| ROBERTASHARE | 152M | 39M | 125M | 26M |
| GPT | 125M | 39M | 125M | 0 |
| RND2GPT | 238M | 39M | 125M | 114M |
| BERT2GPT | 260M | 62M | 234M | 26M |
| ROBERTA2GPT | 276M | 78M | 250M | 26M |

Figura 7: Arquitecturas y número de parámetros probadas en [50]

- Abstractive summarization: resumen abstracto. En este caso los conjuntos de datos empleados son Gigaword [41], CNN and Daily-Mail [21] y BBC extreme [42].

En el caso de resumen abstracto los modelos se evalúan usando métricas automáticas como Rouge-1, Rouge-2, Rouge-L, extensión y repetición, y empleando anotaciones humanas. Todos los modelos basados en BERT superan a los modelos inicializados con parámetros aleatorios por mucha diferencia, y RoBERTaSHARE, BERTSHARE y BERT2BERT superan a GPT-2, en los 3 conjuntos de datos y en todas las métricas. Considerando tanto métricas automáticas como evaluadas manualmente por humanos RoBERTaSHARE es el mejor modelo de resumen abstracto y BERTSHARE el segundo mejor modelo.

Como conclusión, en ese trabajo se muestra que los modelos inicializados con parámetros de modelos preentrena-

dos obtienen mejores resultados que los modelos inicializados con valores aleatorios de parámetros. Por este motivo, resaltamos la importancia de reutilizar modelos ya preentrenados en nuestros experimentos.

### 3.3 T5

En modelos vistos hasta aquí se cuenta con un modelo preentrenado que, para usarlo en una tarea específica, requiere que se realice un ajuste fino. Ahora bien, si se quiere resolver una segunda tarea se debe volver a aplicar un ajuste fino sobre el modelo original generando así un nuevo modelo. De este modo contamos con un modelo ajustado a cada tarea específica (dos en total, en este ejemplo). Por supuesto, cada modelo puede tener distintos hiperparámetros.

En [46] se plantea un enfoque en el cual se preentrena un modelo que luego es ajustado en múltiples tareas pero siempre utilizando el mismo modelo, hiperparámetros y función de pérdida. Este modelo es llamado *Text-to-Text Transfer Transformer* o T5. El modelo emplea la arquitectura original de Transformer con cambios menores pero con mejoras en la representación de la información y en el entrenamiento.

En cuanto a cómo se representa la información, todos los problemas son tratados como problemas de secuencia a secuencia (o texto a texto), lo que justamente permite unificar los problemas. Para hacerlo se introduce un prefijo (prefix token, en inglés) en la entrada que identifica qué tipo de problema es llamado *token de control*.

Supongamos que se desea traducir de español a inglés la frase “El gato está en la caja.”, con la salida esperada

“The cat is in the box”. En este caso, se procesa la entrada para agregar un token de control, resultando, por ejemplo, “translate Spanish to English: El gato está en la caja.ƨ se entrena con la misma salida “The cat is in the box”.

Para tareas de clasificación, se entrena el modelo para que prediga un solo token, correspondiente a la salida esperada. Si se predice un token que no existe entre las clases esperadas, entonces se considera a la predicción como errónea. Por ejemplo, en la tarea de MNLI el objetivo es predecir si una premisa implica, contradice o nada de lo anterior, es decir, es neutral, con respecto a una hipótesis. Así, un caso con el formato adecuado para T5 se vería como “mnli premise: I hate pigeons. hypothesis: My feelings towards pigeons are filled with animosity.”, donde la salida esperada es “entailment”.

Pueden verse otros ejemplos en la Figura 8. Ahí, por ejemplo, se ve que en el caso de la tarea conocida como CoLa donde se predice la aceptabilidad de una frase, se agrega el token de control “cola sentence”. En la figura se observa que la entrada formateada queda como “cola sentence: The course is jumping well”, mientras que la salida esperada es “not acceptable”. Considérese que la salida también esta formatada, ya que originalmente “not acceptable”se representa numéricamente como 0.

Para poder entrenar un modelo que efectivamente resuelva múltiples tareas los autores de [46] construyen el conjunto de datos *Colossal Clean Crawled Corpus* (C4), tomando y limpiando 750 GB de información en inglés de *Common Crawl*.

El modelo luego es ajustado en las siguientes tareas:

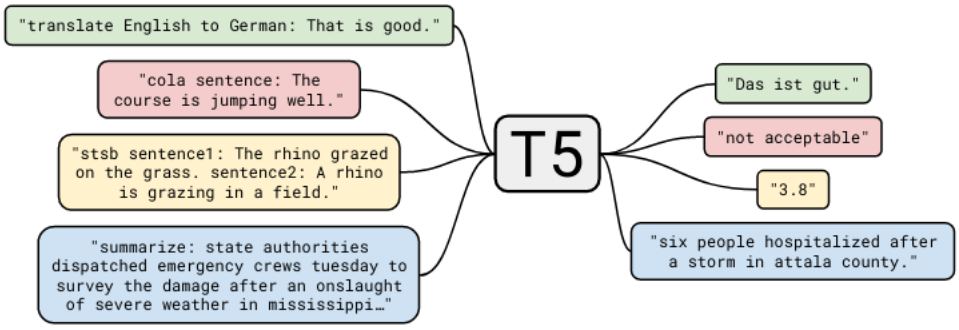


Figura 8: Diagrama de modelo unificado T5, tomado de [46]

- Los problemas de clasificación de Glue [60] y SuperGlue [59], y los autores agregan una tarea adicional, en total sumando 17.
- Resumen abstracto usando CNN/Daily-Mail.
- Question Answering en SQuAD [47].
- Traducción de inglés a alemán, fracés y rumano.

En [46] además realizan muchos experimentos donde analizan los siguientes aspectos de entrenar un modelo:

- **Arquitecturas**: comparan modelos transformer con las arquitecturas encoder-decoder de 12 capas, encoder-decoder con 6 capas, encoder-decoder compartiendo parámetros, puro decoder (que la llaman *language model*) y una arquitectura decoder con visibilidad completa del mecanismo de auto-atención de la entrada, llamada *prefix language model*. La arquitectura que mejor resultado obtiene es la encoder-decoder.
- **Objetivo no supervisado**: cómo preentrenar el modelo de manera no supervisada es crucial, ya que en

| Objective | Inputs | Targets |
|---|---|---|
| Prefix language modeling | Thank you for inviting | me to your party last week . |
| BERT-style Devlin et al. (2018) | Thank you <M> <M> me to your party apple week . | *(original text)* |
| Deshuffling | party me for your to . last fun you inviting week Thank | *(original text)* |
| MASS-style Song et al. (2019) | Thank you <M> <M> me to your party <M> week . | *(original text)* |
| I.i.d. noise, replace spans | Thank you <X> me to your party <Y> week . | <X> for inviting <Y> last <Z> |
| I.i.d. noise, drop tokens | Thank you me to your party week . | for inviting last |
| Random spans | Thank you <X> to <Y> week . | <X> for inviting me <Y> your party last <Z> |

Figura 9: Distintos esquemas de objetivos, tomado de [46]

esa instancia es cuando el modelo gana conocimiento de propósito general, lo que permite una mejor generalización en las tareas posteriores. Los distintos tipos de objetivos probados y ejemplos pueden verse en la figura 9. El objetivo que mejores resultados obtiene es llamado *i.i.d. denoising* ('i.i.d noise, replace spans' en la tabla). En este caso lo que se hace es tomar aleatoriamente palabras y enmascararlas con distintos tokens (a diferencia de BERT que sólo usa [MASK]). Luego, el modelo debe generar una frase con máscaras en las palabras originalmente visibles y predecir las palabras enmascaradas en la entrada. Además, no se enmascaran tokens individuales sino *ventanas*, es decir, conjunto de tokens sucesivos. Entonces, cuántos tokens sucesivos van a enmascararse y qué porcentaje del total de las palabras se enmascaran son hiperparámetros. Luego de diversos experimentos se determina que una longitud promedio de la ventana de 3 y un porcentaje de corrupción de 15 %. Este objetivo además de ser levemente mejor que los demás en cuanto a performance tiene un costo computacional algo menor gracias a que el uso de *ventanas* reduce la extensión de los *objetivos*.

- **Conjuntos de datos**: se analizan distintos conjuntos de datos para preentrenar el modelo. Se consideran C4 habiendo aplicado filtros para limpiarlo, C4 sin limpiar, un conjunto de datos de noticias, uno basado en Reddit, Wikipedia y, finalmente, Wikipedia sumado a un corpus de libros. Como es esperable, omitir la limpieza de C4 claramente empeora los resultados obtenidos. Sin embargo, no es tan clara la conclusión con respecto a los demás conjuntos de datos. En ciertos casos usar conjuntos de datos del mismo dominio que las tareas en las que se va a realizar un ajuste fino genera mejores resultados, a pesar de ser corpus mucho más chicos. Sin embargo, justamente su tamaño es una limitación ya que, como también muestran, cuando las observaciones se repiten en grandes cantidades los modelos tienden a memorizar y *sobreajustar* el conjunto de entrenamiento. 64 repeticiones es el número óptimo que los autores encuentran, y hasta 256 hay buenos resultados, pero más allá de eso se encuentran problemas serios de performance. Además, es preciso mencionar que este número está en relación directa al tamaño de la arquitectura que se está entrenando.
- **Estrategias de entrenamiento**: en primer lugar se analizan tres estrategias de ajuste fino:
  - Entrenar todos los parámetros de manera simultanea
  - Agregar capas *adaptadoras* (o *adapter layers*), consistente en bloques de capas densa-RELU-densa (*dense-RELU-dense*) que se agregan después de las

capas *feed-forward*. Con estos bloques se agrega un hiperparámetro que regula la dimensionalidad del mismo y, por lo tanto, cuánto parámetros nuevos se agregan. Finalmente, en este caso se dejan freezados los demás parámetros y sólo se entrenan las capas adaptadoras y las capas de normalización.

- Un tercer procedimiento que se considera es el llamado *gradual unfreezing* que consiste en entrenar primero la primera capa (la más exterior), luego de cierta cantidad de pasos o epochs agregar la segunda capa más exterior, y así sucesivamente hasta entrenar todas las capas.

La estrategia que obtiene mejores resultados es entrenar todas las capas, aunque el entrenamiento con capas adaptadoras también genera buenos resultados en ciertas tareas. En segundo lugar, hasta el momento nos hemos centrado en la estrategia de preentrenar (usando un objetivo no supervisado) y luego ajustar (usando los conjuntos de datos correspodientes a cada tarea). Sin embargo, también existe el llamado aprendizaje multitarea (*multi-task learning*, en inglés), por el cual se aprenden todas las tareas simultáneamente. Además, ya que T5 implementa un estrategia secuencia a secuencia para todas las tareas, aplicar aprendizaje multitarea es, en este caso, mezclar todos los conjuntos de datos, incluyendo a la tarea no supervisada como una tarea adicional. El mayor desafío en este caso es qué estrategia usar para balancear los conjuntos de datos. Por último, los autores también prueban preentrenar

usando aprendizaje multitarea y luego hacer un ajuste fino a cada tarea, y obtienen resultados similares a preentrenar de manera no supervisada y luego ajustar. Finalmente, eligen preentrenamiento multitarea y luego ajuste fino para las versiones finales de los modelos, ya que permite monitorear la performance de las tareas durante el preentrenamiento.

- **Scaling**: los autores del trabajo también analizan cuál es la mejor manera de aprovechar más capacidad o presupuesto de cómputo, a ello lo llaman *scaling*. Realizan distintas pruebas con 4 veces más cómputo y lo comparan con un modelo base, las pruebas que realizan son:
  - Usar el mismo modelo pero con 4 veces más *pasos*.
  - Usar el mismo modelo pero con un tamaño de lote (o batch size, en inglés) 4 veces mayor.
  - Usar un modelo con el doble de parámetros y entrenarlo el doble de *pasos*.
  - Usar un modelo con 4 veces más parámetros.
  - Preentrenar y ajustar 4 modelos diferentes y luego ensamblar sus predicciones. En este caso se usa 4 veces el cómputo del baseline de manera ortogonal.
  - Preentrenar un solo modelo y hacer cuatro modelos ajustados y luego ensamblar sus predicciones. En este caso se usa menos que 4 veces el cómputo del baseline de manera ortogonal.

  Los resultados muestran que no hay una única estrategia que sea estríctamente superior. Preentrenar un

modelo más grande se muestra efectivo y parece complementarse bien con un incremento en la cantidad de *pasos*. Por otro lado, preentrenar un modelo más grande también tiene un mayor costo a la hora de hacer un ajuste fino. A su vez, los autores muestran que ensamblar modelos también puede ser muy efectivo en ciertas tareas, incluso sin hacer preentrenamientos separados.

Los autores abren el modelo T5 en distintos tamaños, obteniendo *state-of-the-art* en casi todas las tareas y, como es esperado, el mejor resultado lo obtienen con el modelo más grande, T5 11B, de aproximadamente 11 mil millones de parámetros.

En total, superan el estado del arte (a octubre de 2019) en 18 de las 24 tareas en las que se prueba.

En **Glue** [60] T5 supera, en promedio, los resultados obtenidos por humanos. Además, supera los resultados humanos en todas las tareas excepto en WNLI y RTE. A modo de ejemplo, en Glue podemos encontrar tareas como **Quora Question Pairs**, en la cual se predice si dos preguntas significan lo mismo o no y **The Corpus of Linguistic Acceptability**, en donde se predice si una frase está bien formada o no.

En cuanto a **SuperGlue** [59], T5 obtiene, en promedio, un resultado levemente peor que los resultados humanos, sin embargo, mejora el estado del arte en la mayoría de las tareas. Entre las tareas de SuperGlue podemos mencionar **Choice of Plausible Alternatives (COPA)**, en la cual la tarea consiste en elegir la alternativa más lógica en base a una premisa. Por ejemplo:

- Premise: The man broke his toe. What was the CAUSE

of this?

- Alternative 1: He got a hole in his sock.
- Alternative 2: He dropped a hammer on his foot.

A modo de ejemplo, otra tarea de SuperGlue es **BoolQ**, en donde se presenta un texto, una pregunta (que se responde en base al texto) y el objetivo es predecir la respuesta, que puede ser verdadero o falso. Esta tarea es una versión más difícil de la tarea NLI de Glue.

Por otra parte, en [64] los autores entrenan el modelo MT5, una versión multilenguaje de T5. Para ello, construyen el corpus MC4, el equivalente multilenguaje de C4, con un total de 101 idiomas. Es interesante mencionar que el español aparece como el tercer lenguaje con más páginas en este conjunto de datos, por detrás el inglés y el ruso.

Para entrenar este modelo siguen lo expuesto previamente para T5, con un mayor tamaño, empleando sólo un objetivo no supervisado para entrenar y empleando activaciones GeGLU [54]. El vocabulario es de un total de 250.000 *wordpieces*.

En ese trabajo comparan los resultados de mT5 con otros modelos multilenguaje como mBERT [14], XLM-R [27], mBART [33] y MARGE [30].

# 4 Metodología

En esta sección se detallan los aspectos metodológicos del trabajo y se organiza en dos subsecciones. En la primera se explican las estrategias elegidas para entrenar los modelos de lenguaje, las arquitecturas elegidas y los modelos preentrenados que se usan. Además, se definen las tareas sobre las cuales se va a hacer el ajuste fino.

En la segunda sección se especifican los conjuntos de datos sobre los cuales se entrenan los modelos. Allí se mencionan dos conjuntos de datos creados a tal fin, que llamamos CC-NEWS-ES y CC-NEWS-ES-titles, y un conjunto de datos actualmente disponible que se emplea como benchmark, MLSUM.

## 4.1 Modelos

Según lo que hemos visto previamente, para desarrollar un modelo de resumen abstracto a bajo costo y empleando las técnicas más recientes analizamos dos alternativas viables:

- Emplear alguno de los modelos multilenguaje mencionados anteriormente y realizar un ajuste fino en español.
- Usar la estrategia de inicializar un modelo preentrenado en una arquitectura encoder-decoder como en [50]. En este caso, el único modelo disponible para hacerlo es Beto [9].

En cuanto a la primera estrategia, los modelos mBERT (reportado en [52]) y MARGE presentan benchmarks de

resumen abstracto en español sobre el conjunto de datos MLSUM. Sin embargo, MARGE no está disponible públicamente al momento de escribir este trabajo. Por otra parte, dado que mBERT es un modelo únicamente encoder para generar texto es necesario adaptar la salida, como se realiza en [16].

Por otra parte, tanto mBART como XLM-R usan un conjunto de datos que consta de 9.374 millones de tokens en la sección en español, mientras que el corpus de mT5 (mC4) consta de 433.000 millones de tokens en español.

En otro sentido, mT5 tiene distintas versiones, siendo la más chica mT5 Small, con alrededor de 300 M de parámetros.

| Modelo | Arquitectura | N° de parám. | N° de tokens en español | Conjunto de datos |
|---|---|---|---|---|
| mBERT | encoder | 180 M | 962 [1] | Wikipedia |
| XLM | encoder | 570 M | 2.412 | MultiUN [66] |
| XLM-R | encoder | 270-550 M | 9.374 | CC-100 |
| mBART | encoder-decoder | 680 M | 9.374 | CC25 |
| MARGE | encoder-decoder | 960 M | 9.780 [2] | Wikipedia + CC-NEWS |

| Modelo | Arquitectura | N° de parám. | N° de tokens en español | Conjunto de datos |
|---|---|---|---|---|
| mT5 | encoder-decoder | 300 M - 13.000 M | 433.000 | mC4 |

Cuadro 1: Estadísticas de modelos multilenguaje

En la tabla 1 se ven las estadísticas de los distintos modelos multilenguajes. Debido a que es uno de los modelos más recientes y que, además, es el modelo con el mayor corpus en español en el entrenamiento se elige a mT5 como modelo para realizar el ajuste fino. Además, para reducir los costos de entrenamiento se toma sólo mT5 small, ya que consta de una cantidad de parámetros que permite un entrenamiento accesible. Sin embargo, es probable que se obtuviesen mejores resultados con los modelos más grandes.

De esta manera, la primera estrategia será hacer un ajuste fino de **mT5 small** a las distintas tareas.

En segundo lugar, se propone utilizar Beto para inicializar un modelo encoder-decoder sin compartir parámetros. Siguiendo a [50] podemos llamar a este modelo Beto2Beto. Ahora bien, en el conjunto de datos sobre el cual Beto es entrenado no predominan ni noticias ni el español hablado en Latinoamérica. Entonces, para orientar el conocimiento general del modelo al lenguaje de noticias y, además, para dar mayor peso al español hablado en los países de América Latina se construye el conjunto de datos CC-NEWS-ES.

De esta manera, el procedimiento consta de dos instancias: en primer lugar, se inicializa el modelo encoder-decoder con Beto tanto en el encoder como en el decoder; en segundo lugar, preentrenamos ese modelo en CC-NEWS-ES. Luego, este modelo es ajustado a tareas específicas.

Los modelos son evaluados en dos tareas:

- Resumen abstracto en el conjunto de datos MLSUM
- Generación de títulos, que es también una tarea de resumen abstracto en un conjunto de datos construído a tal fin. Este conjunto de datos lo llamamos CC-NEWS-ES-titles

En resumidas cuentas, como parte de este trabajo se entrenan 5 modelos:

- MT5-small-mlsum: modelo mt5 especializado en MLSUM.
- MT5-small-cc-news-es-titles: modelo mt5 especializado en CC-NEWS-ES-titles.
- Beto2Beto: modelo encoder-decoder inicializado con Beto tanto en el encoder como decoder, preentrenado sobre CC-NEWS.
- Beto2Beto-cc-news-es-titles: Beto2Beto especializado en CC-NEWS-ES-titles.
- Beto2Beto-mlsum: Beto2Beto especializado en MLSUM.

### 4.2 Datos

En esta subsección se describen los datos empleados en el trabajo.

#### 4.2.1 CC-NEWS-ES

Common Crawl (CC) [3] es una organización que se dedica a recolectar información pública de la web, organizarla y hacerla accesible. En 2016 CC lanza un índice especial para noticias: CC-NEWS [4] .

Los datos de CC son varios órdenes de magnitud más grandes que otros corpus ampliamente usados en las tareas de NLP como Wikipedia y BookCorpus. Esto lo hace útil y ampliamente elegido para entrenar modelos como, por ejemplo, T5. Sin embargo, vale la pena advertir que este corpus contiene discursos de odio, contenido sexual explícito, contenido racista, entre otros tipos de expresiones que pueden generar sesgos inapropiados en los modelos entrenados, dando lugar a expresiones potencialmente dañinas o no seguras.

En [35] se realiza un estudio de este problema, encontrando que entre el 4 % y 6 % de las páginas web tienen discursos de odio, y que entre el 1 % y 2 % tienen contenido sexual explícito. También muestra que estas páginas inapropiadas permanecen en el conjunto de datos luego de haber limpiado el mismo mediante técnicas basadas en *perplexity*, técnicas usada comúnmente para mejorar la calidad de los datos antes de entrenar. Por último, el análisis allí desarrollado está basado en el índice de páginas web, no en el índice de noticias que usamos en este trabajo y presuponemos más curado, ya que hay una lista en la cual se explicitan las páginas en las cuales recolectar los datos.

En CC-NEWS la información se almacena en archivos

[3] https://commoncrawl.org
[4] https://commoncrawl.org/2016/10/news-dataset-available/

WARC de 1 GB aproximadamente cada uno y en total hay aproximadamente 4 TB de información por año.

En el caso de CC-NEWS la información no es almacenada ni por dominio ni por lenguaje, sino por fecha de creación. Tampoco consta de un índice para buscar por idioma. Por este motivo, el proceso efectuado es el siguiente:

1. Se recorren los archivos WARC de 2019
2. Para cada archivo se obtienen el dominio de nivel superior y el cuerpo de la noticia
3. Para cada párrafo se predice el lenguaje usando FastText y si es español con una probabilidad mayor a 0.5 entonces éste es incluído.
4. Si el texto tiene más de 50 palabras se incluye en el conjunto de datos

| Dominio | N° de textos | N° de palabras |
|---|---|---|
| ar | 532.703 | 145.127.369 |
| bo | 29.557 | 7.289.964 |
| br | 107 | 14.207 |
| cl | 116.661 | 33.463.315 |
| co | 78.662 | 19.264.943 |
| com | 3.650.951 | 844.093.906 |
| cr | 16.542 | 3.820.752 |
| es | 1.838.794 | 482.943.483 |
| gt | 4.833 | 838.121 |

| Dominio | N° de textos | N° de palabras |
|---|---|---|
| hn | 36.559 | 5.499.330 |
| mx | 724.908 | 162.198.054 |
| ni | 40.643 | 10.850.146 |
| pa | 18.447 | 4.347.245 |
| pe | 230.962 | 35.212.326 |
| pr | 7.756 | 1.663.300 |
| py | 30.651 | 20.807.689 |
| sv | 454 | 353.145 |
| uy | 80.948 | 27.256.206 |
| ve | 33.148 | 6.965.782 |
| total | 7.473.286 | 1.812.009.283 |

Cuadro 2: Estadísticas de CC-NEWS-ES

#### 4.2.2 CC-NEWS-ES-titles

En segundo lugar, se crea el conjunto de datos CC-NEWS-ES-titles consistente en el cuerpo de la noticia y su título, con la información de 2019 y 2020. Este corpus consta de 402.310 observaciones, distribuidos en:

Entrenamiento: 370.125 Validación: 16.092 Test: 16.092

Debido a que es común que los títulos tengan ciertos sesgos es poco probable que éstos se puedan obtener directamente del cuerpo de la noticia, lo cual hace que generarlos sea una tarea muy adecuada para modelos de resumen abstracto.

Puede verse un ejemplo a continuación:

- text: Instagram elimina un video de Madonna por desinformar sobre el coronavirus‘Tengo esa autoridad. Puedo hacerlo con una orden ejecutiva‘, afirmó el mandatario, quien detalló que planea tomar la decisión este mismo sábado como pronto.A principios de mes, el secretario de Estado de EE.UU., Mike Pompeo, ya dejó entrever que el Gobierno de Trump consideraba restringir el acceso a TikTok en Estados Unidos ante la posibilidad de que Estados Unidos supera las 150.000 muertes a causa de la pandemiaEn un evento organizado por el diario The Hill, Pompeo explicó que la Administración está valorando imponer sanciones y aseguró que ‘en breve‘ comunicarán al público ‘la serie de decisiones‘ que se han tomado.TikTok es una red social desarrollada por ByteDance, con sede en Pekín (China), en la que se comparten videos cortos y que ha logrado un gran éxito entre el público adolescente, pero que a la vez ha levantado
- output text: ¿Por qué Donald Trump anunció que prohibirá TikTok en Estados Unidos?

#### 4.2.3 MLSUM

En [52] se presenta MLSUM, el primer conjunto de datos multilenguaje orientado a resúmenes y el primero de gran escala en español. Este corpus provee de noticias en francés, alemán, español, ruso y turco, y se complementa muy bien con el conjunto de datos más famoso en su tipo CNN/Daily Mail, puesto que éste también tiene su fuente en noticias.

Los periódicos que toma son:

- Le Monde de Francia
- Suddeutsche Zeitung de Alemania
- El Pais de España
- Moskovskij Komsomolets de Rusia
- Internet Haber de Turquía

En este trabajo tomamos sólo la sección en español del mismo, cuyas estadísticas son:

1. Total de casos: 290,645
2. Training: 266,367
3. Extensión promedio del artículo: 800.50
4. Extensión promedio del resumen: 20.71
5. Ratio de compresión (4/5): 38.7
6. Novelty (unigramas): 15.34
7. Tamaño del vocabulario total: 1,257,920
8. Palabras que ocurren más de 10 veces: 229,033

Además, luego de un análisis manual se notó que el objetivo es predecir el subtítulo de las noticias.

# 5 Evaluación de modelos

Comenzamos esta sección describiendo brevemente las métricas de evaluación automáticas que se emplean en el trabajo, específicamente *perplexity* y las variantes de *rouge*. Luego, nos centramos en algunas críticas a este método de evaluación y revisamos autores que se refieren a la necesidad de una evaluación basada en anotaciones manuales. Ambas técnicas son posteriormente empleadas para analizar la calidad de los modelos.

## 5.1 Perplexity y Rouge

En los modelos de lenguaje la métrica generalmente elegida para evaluar la performance de los mismos es conocida como perplexity [24]. Dada una secuencia de tokens $X = (x_0, x_1, ..., x_t)$ entonces la perplexity de $X$ es:

$$\mathrm{PPL}(X) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta\left(x_i \mid x_{<i}\right)\right\} \tag{5.1}$$

donde $\log p_\theta\left(x_i \mid x_{<i}\right)$ es el logarítmo de la verosimilitud del token i-ésimo, condicional en los tokens previos.

Por otro lado, para evaluar la calidad en la generación de resúmenes se emplea la métrica conocida como Rouge [31], la cual mide el solapamiento entre palabras en el texto generado y el texto objetivo. Existen distintas versiones de esta métrica, las más comunes son Rouge-N y Rouge-L.

Rouge-N se define como el recall de n-gramas entre un texto candidato y múltiples textos objetivo, ya que para un mismo texto pueden haber múltiples resúmenes correctos:

$$\text{Rouge-N} = \frac{\sum_{S \in [\text{ ReferenceSummaries }]} \sum_{\text{gram }_n \in S} \text{ Count }_{\text{match}} \text{ ( gram }_n)}{\sum_{S \in [\text{ ReferenceSummaries }]} \sum_{\text{gram }_n \in S} \text{ Count } \text{ ( gram }_n)} \tag{5.2}$$

Se reportan Rouge-1, el recall de unigramas (tokens), y Rouge-2, el recall de bigramas.

Por otro lado, Rouge-L es el F-score de la subsecuencia común más extensa (LCS). Sea $X$ un texto de referencia de longitud $m$ e $Y$ un texto candidato de longitud $n$:

$$\text{Rouge-L} = 2 * precision * recall / (precision + recall) \tag{5.3}$$

siendo:

$$precision = \frac{LCS(X,Y)}{n} \tag{5.4}$$

y

$$recall = \frac{LCS(X,Y)}{m} \tag{5.5}$$

Rogue-L tiene las siguientes propiedades:

- Rouge-L es 0 si no hay nada en común, i.e. $LCS(X,Y) = 0$
- Rouge-L es 1 si $X = Y$
- No hay necesidad de predefinir la extensión de n-gramas
- Las medidas basadas en n-gramas no toman en consideración el orden, mientras que el LCS sí capta aspectos de la estructura de las oraciones.

Por último, Rouge-Lsum (LCS a nivel de resumen) computa la suma de LCS para todas las frases que forman parte de un mismo resumen.

### 5.2 Críticas a Rouge

En [56] se estudian algunos limitantes de Rouge, concluyendo que esta métrica presenta limitaciones para diferenciar resúmenes que son ciertos de los que no. Además, muestran que un cambio en el preprocesamiento como eliminar stopwords o aplicar stemming puede afectar la métrica.

Véase, por ejemplo, la Figura 10. Allí tenemos:

- Referencia: The rooms were neat and clean. En español: los cuartos estaban ordenados y limpios.
- Resumen 1: Clean room. En español: cuarto limpio.
- Resumen 2: The rooms were dirty. En español: Los cuartos estaban sucios.

Para cada resumen se aplican cuatro tipo de preprocesamientos (uno de los cuales es no hacer ningún preprocesamiento) y se calculan cuatro métricas de calidad. En la mayor parte de los escenarios el resumen 2 obtiene un puntaje superior o igual, a pesar de que el significado es contrario al del texto de referencia. Esto se debe a que la cantidad de términos que coinciden con el texto de referencia es mayor en términos relativos al resumen 1. A su vez, ésto también se debe a que el término que no coincide (*dirty*) está al final del texto, lo cual favore a rouge-2 y rouge-L. Por otro lado, el resumen 1 dice *room* en vez del *rooms*, esta diferencia mínima afecta las métricas en los casos en que no se aplica stemming.

En términos generales rouge no mide si el resumen refleja correctamente los hechos que se desean transmitir. Por

| **Reference:** The rooms were neat and clean. | | | **Summary1:** Clean room. | | | | **Summary2:** The rooms were dirty. | |
|---|---|---|---|---|---|---|---|---|
| **Configuration** | **R-1** | **R-2** | **R-L** | **R-SU4** | **R-1** | **R-2** | **R-L** | **R-SU4** |
| None | 0.250 | 0.000 | 0.250 | 0.087 | 0.600 | 0.500 | 0.600 | 0.400 |
| Stemming | 0.500 | 0.000 | 0.250 | 0.174 | 0.600 | 0.500 | 0.600 | 0.400 |
| StopWordRemoval | 0.400 | 0.000 | 0.400 | 0.222 | 0.400 | 0.000 | 0.400 | 0.222 |
| StopWordRemoval+Stemming | 0.800 | 0.000 | 0.400 | 0.444 | 0.400 | 0.000 | 0.400 | 0.222 |

Figura 10: Ejemplos de cálculo de Rouge, tomado de [56]

este motivo, muchas veces los trabajos acompañan sus resultados con trabajos de etiquetado manual [17][8].

En [26] se muestra que un problema importante en la tarea de resumen es la falta de restricciones en torno a en qué enfocar el resumen. Para ello realizan el siguiente experimento. Empleando anotadores construyen dos conjuntos de datos: en el primero se les pide a los anotadores resumir noticias sin ningún tipo de restricción, en el segundo se les pide que lo hagan de modo de contestar tres preguntas. Luego, se plantea la siguiente pregunta, ¿cuántas frases por artículo se considerarían relevantes si el mínimo de personas que acuerdan en ello es X?

| Human vote threshold | Sent. per article considered important | |
|---|---|---|
| | *Unconstrained* | *Constrained* |
| $= 5$ | 0.028 | 0.251 |
| $\geq 4$ | 0.213 | 0.712 |
| $\geq 3$ | 0.627 | 1.392 |
| $\geq 2$ | 1.695 | 2.404 |
| $\geq 1$ | 5.413 | 4.524 |

Figura 11: Cantidad de frases relevantes para los grupos restrictos e irrestrictos, tomado de [26]

Como se puede ver en la Figura 11 hay una amplia dife-

rencia en el acuerdo de si una frase es relevante o no, según si se restringe la tarea de resumen o no.

En [26] también lleva a cabo un análisis para entender si rouge es una métrica adecuada de evaluación. Para ello, toman 100 notas al azar y el resumen generado por 13 sistemas distintos (tanto extractivos como abstractos). Cada resumen es luego puntuado por 5 anotadores distintos en las siguientes dimensiones:

1. Relevancia: si se selecciona contenido importante.
2. Consistencia: si los hechos que marca el resumen están alineados con la fuente.
3. Fluidez: calidad de la frases individuales.
4. Coherencia: calidad del conjunto de todas las frases.

Luego, se obtiene el valor final como el promedio del puntaje dado por los 5 anotadores. Estos valores son luego comparados con el rouge y se calculan las correlaciones de Pearson y de Kendall. Además, se emplean 1, 5 y 10 resúmenes. En todos los casos se aprecia que hay una muy baja correlación entre las métricas automáticas y humanas. Además, se analizan muestras a mano y se concluye que en un 30 % de las mismas hay hechos incorrectos que, en muchos casos, podrían pasar como ciertos.

Con lo dicho previamente no se pretende descartar las métricas basadas en rouge sino remarcar que éstas son insuficientes para poder evaluar modelos y que, por lo tanto, es valioso incluir un proceso de evaluación humana en esta tarea.

En [26] también remarcan la importancia del llamado *layout bias* en las noticias. Este sesgo se refiere a que en las noticias es común incluir la parte más importante de la información en los primeros párrafos y en el resto se incluyen detalles o información de contexto. Como las noticias son una de las principales fuentes para este tipo de modelos, estos modelos a su vez pueden verse afectados por este tipo de señal.

Por último en [65] también realizan un análisis basado en anotaciones para profundizar en la calidad de los modelos. Para ello toman una muestra de 100 documentos de cada conjunto de datos y distintos anotadores puntúan 5 resúmenes (incluyendo los originales que en adelante denomino "humanos") de 1 (resumen pobre) a 5 (buen resumen). Luego, realizan un test-t para ver si las puntuaciones son significativamente diferentes o no.

Por lo aquí expuesto, en este trabajo también realizamos un análisis basado en anotaciones humanas para poder entender en mayor profundidad ventajas y desvantajas de los modelos entrenados.

# 6 Resultados

La sección comienza con comentarios acerca de las tecnologías empleadas y de los modelos que se entrenan. Además, se detalla la forma de entrenamiento de los mismos y los resultados obtenidos según las métricas previamente mencionadas. Luego esta sección continua con los resultados obtenidos a partir de las anotaciones humanas, allí se ahonda en la metodología empleada y se analizan los puntajes obtenidos para cada modelo. Continuamos mostrando algunos de los resúmenes empleados en el conjunto analizado y, luego, ejemplos de títulos generados. Por último, se describe cómo se disponibilizan los modelos y conjuntos de datos al público en general mediante HuggingFace y de qué manera se pueden usar.

## 6.1 Entrenamiento de modelos y métricas automáticas

Todo el trabajo es desarrollado usando el lenguaje de programación Python. Para la creación de los conjuntos de datos se usan paquetes comunes tales como `pandas`, `requests`, `multiprocessing`, `beautifulsoup`, `numpy`, `json`, etc. Además, el entorno elegido es AWS y se emplea S3 para almacenaje y SageMaker para el entrenamiento de los modelos. Por este motivo también se emplea el paquete boto3 (para interactuar con los servicios de AWS). Por último, para el entrenamiento de los modelos se eligen las librerías de *HuggingFace* `transformers` y `datasets`.

Como se menciona previamente, se entrenan 5 modelos:

- Beto2Beto

- Beto2Beto-cc-news-es-titles
- Beto2Beto-mlsum
- MT5-small-mlsum
- MT5-small-cc-news-es-titles

#### 6.1.1 Beto2Beto

En primer lugar, se preentrena Beto2Beto como en [50] tomando como modelo preentrenado tanto del encoder como del decoder el modelo BERT en español Beto [9].

Los parámetros empleados son:

- Encoder max length: 40
- Decoder max length: 128
- 3 epochs, un total de 68.000 pasos.

Beto2Beto se entrena sobre un conjunto de datos construído a partir de CC-NEWS-ES. Para ello primero se preprocesan los textos realizando los siguientes pasos:

1. Se pasa todo el texto a minúscula.
2. Se convierte a unidecode. Esta decisión por un lado reduce el vocabulario, ya que unifica textos en donde se incluyen tildes, por ejemplo 'había' y 'habia' o 'mañana' y 'manana' se convierten en el mismo token. Esta decisión por un lado facilita al modelo aprender embeddings para ese tipo de ejemplos y, por otro lado, disminuye la dimensión del problema. Sin embargo, esta decisión trae aparejado que el texto generado no

distinguirá entre palabras con tildes y sin tildes o 'ñ' y 'n' a la hora de generar nuevo texto.

3. Se eliminan los 'xa0'.

4. Se eliminan los caracteres no alfanuméricos.

5. Se eliminan múltiples espacios consecutivos.

En segundo lugar, se preparan los datos para entrenar bajo un esquema de *prefix language model*, como se vio anteriormente. Para ello se eliminan los documentos con menos de 64 palabras. Seguidamente cada documento se divide en una entrada (*text*) de hasta 40 tokens y una salida (*output text*) de hasta 128 tokens. Se eligen 128 tokens como máximo para que no crezca demasiado el costo computacional del entrenamiento.

Puede verse un ejemplo a continuación:

- text: El fin de esta serie medieval-fantástica exitosa en todo el planeta, que cuenta la historia de

- output text: familias que pelean por reinar en el mundo ficticio de Westeros, desató las pasiones de los fanáticos presentes en el pub Brennanś, que incluso tenía para la ocasión su propio Trono de Hierro (emblema del poder).El establecimiento, en el oeste de Los Ángeles, también sirvió un coctel inspirado en 'Juego de Tronos': el 'night king', o 'rey de la noche', el gran villano de la serie, y que consistía en un whisky sour con hielo azul, según aseguró a la AFP un aficionado a la bebida.La última temporada fue, en efecto, objeto de fuertes polémicas incluso antes de su episodio final.

> Las críticas se centraron sobre todo en la aceleración del ritmo de la serie del canal de pago HBO, que dio lugar a cambios apresurados y a que algunas escenas cruciales hayan durado apenas unos minutos, después de años de espera.Eso no molestó sin embargo a Corey Ben-David, seguidora de la 'Madre de Dragones' y quien se disfrazó de ella para participar en la fiesta de este domingo. 'Dany hizo lo que ella misma dijo. Su evolución es lógica', dice la mujer de 34 años, que ha leído todos los libros de George R. R. Martin que inspiraron la serie televisiva.

Así, con todos los textos en español de 2019 se construye un primer conjunto de datos, consistente en 5.959.789 de notas, con una mediana de 155 palabras por nota. Este conjunto de datos está dividido en: Entrenamiento: 5.893.175 Test: 66.614

Adicionalmente, se incluyen dos conjuntos de testeo. El primero (Test-1), con datos de inicios de 2020 que, presuntamente, debería tener pocas diferencias con la información usada para entrenar de 2019. Además, un segundo conjunto de testeo basado en información de todo 2020, el cual se espera que sea muy diferente al de 2019, por la preponderancia de la pandemia del coronavirus en las noticias.

El preentrenamiento se realiza tomando los primeros 40 caracteres del texto como entrada y los siguientes como objetivo. Se trunca el objetivo a los 128 tokens para evitar un mayor costo de entrenamiento, ya que el tamaño del encoder y del decoder tienen un impacto importante en el costo computacional.

En una máquina ml.p3.8xlarge, lleva aproximadamente

160 horas de cómputo y se alcanza una pérdida en test de 2.651. Puede verse la evolución de la pérdida en la Figura 12.

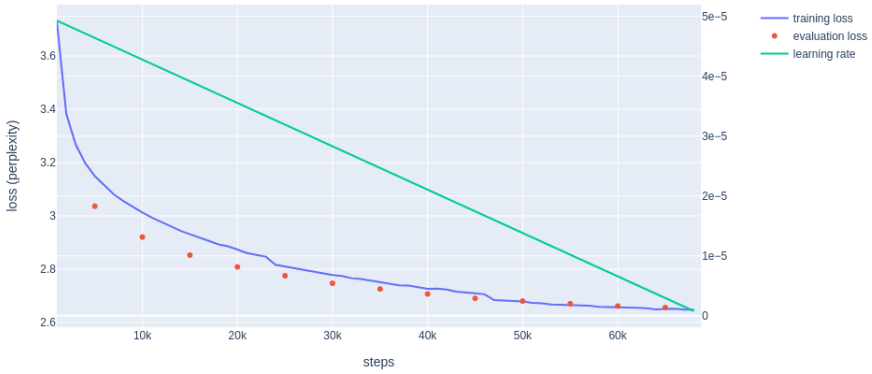


Figura 12: Perplexity del modelo Beto2Beto

### 6.1.2 Ajuste fino a MLSUM

A partir de Beto2Beto se entrena un modelo de resumen abstracto sobre el conjunto de datos MLSUM. También se hace lo mismo para esta tarea a partir de MT5-small. Ambos modelos usan los siguientes parámetros en el ajuste fino:

- Encoder max length: 512
- Decoder max length: 64
- 10 epochs

En la tabla 3 se pueden ver las métricas de performance de estos dos modelos junto a los modelos M-BERT [52] y a MARGE-NEWS [30].

[5]

| Métrica | MT5-mlsum | Beto2Beto-mlsum | M-BERT | MARGE-NEWS |
|---|---|---|---|---|
| eval loss | 1.925 | 2.502 | | |
| eval rouge1 | 26.4352 | 26.1256 | | |
| eval rouge2 | 8.9293 | 9.2552 | | |
| eval rougeL | 21.2622 | 21.4899 | | |
| eval rougeLsum | 21.5518 | 21.8194 | | |
| eval length | 33.9532 | 19.239 | | |
| test loss | 2.006 | 2.577 | | |
| test rouge1 | 26.0756 | 25.8639 | | |
| test rouge2 | 8.4669 | 8.911 | | |
| test rougeL | 20.8167 | 21.2426 | 20.44 | |
| test rougeLsum | 21.0822 | 21.5859 | | 22.72 |

[5]Si bien esos trabajos no lo aclaran presuponemos que las métricas reportadas son de test (y no de validation). Desafortunadamente al momento de escribir ésto ninguno de esos modelos se encuentra disponible al público como para replicar esos resultados.

| Métrica | MT5-mlsum | Beto2Beto-mlsum | M-BERT | MARGE-NEWS |
|---|---|---|---|---|
| test length | 33.803 | 19.2463 | | |

Cuadro 3: Performance en MLSUM

Como se puede ver, nuestros modelos alcanzan un mejor resultado en RougeL que M-BERT y un peor resultado que MARGE-NEWS en rougeLsum. El modelo basado en MT5 tiene un promedio en torno a los 33-34 tokens, el modelo basado en Beto2beto tiene un largo en torno a los 19 tokens. Esto se debe a que usan distintos tokenizadores (MT5 usa Sentencepiece y BERT está basado en WordPiece). Si se tokeniza sólo usando espacios entonces la diferencia se reduce drásticamente. En ese caso MT5 muestra una extensión promedio de 21.08 palabras y Beto2beto una longitud de 19.59, en una muestra al azar de 100 casos. Vale la pensa observar que la estadística reportada de MLSUM para su parte en español es de 20.71 palabras.

#### 6.1.3 Ajuste fino en CC-NEWS-ES-titles

Por otro lado, se realiza un ajuste fino de Beto2Beto y MT5 a la tarea de generación de títulos sobre el conjunto de datos antes mencionado, CC-NEWS-ES-titles.

Los parámetros empleados son:

- Encoder max length: 512
- Decoder max length: 36

- 10 epochs.
- batch size: 128 para MT5 y 256 para Beto2beto.

Los resultados obtenidos son:

| Métrica | MT5-small-cc-news-es-titles | Beto2Beto-cc-news-es-titles |
|---|---|---|
| eval loss | 2.879 | 4.54 |
| eval rouge1 | 22.6623 | 23.7478 |
| eval rouge2 | 7.7894 | 7.3616 |
| eval rougeL | 19.8015 | 20.6615 |
| eval rougeLsum | 19.8092 | 20.7371 |
| eval length | 17.1839 | 16.1806 |
| test loss | 2.878 | 4.515 |
| test rouge1 | 22.9263 | 23.7415 |
| test rouge2 | 7.9146 | 7.3548 |
| test rougeL | 20.0272 | 20.746 |
| test rougeLsum | 20.0387 | 20.8149 |
| test length | 17.1696 | 16.1926 |

Cuadro 4: Pérdida y Rogue en CC-NEWS-ES-titles

Tanto en MLSUM como en este último conjunto de datos podemos concluir que Beto2beto obtiene un mejor rouge pero una loss más alta.

Por lo visto anteriormente es deseable tener una medi-

ción más precisa de la performance de estos modelos mediante un etiquetado manual de las predicciones.

### 6.2 Evaluación con anotaciones

Por lo expuesto previamente además de emplear la métrica *rouge* se decide complementar el análisis de resultados usando anotaciones humanas. Siguiendo una metodología similar a la ya expuesta de [26] se toma una muestra de 98 notas al azar de MLSUM y cinco anotadores puntúan los resúmenes en tres ejes.

Los tres ejes elegidos son los mismos que en el trabajo mencionado pero se juntan fluidez y coherencia en un punto que llamamos 'coherencia y cohesión'. De este modo las aristas que se consideran para evaluar la calidad de los resúmenes son 'relevancia', 'consistencia' y 'coherencia y cohesión'.

Los puntajes para cada concepto pueden ir de 0 a 2 según una guía predefinida con la cual se capacita a los anotadores. En el apéndice 8.2 se puede ver la guía empleada.

Para cada nota se consideran 4 resúmenes:

- Humanos
- Beto2Beto
- MT5
- Pegasus

Debido a que al momento de desarrollar este trabajo no se encontró otro modelo de resumen abstracto en español basado en Transformers y que, por lo tanto, sea fácilmente aplicable a este conjunto de datos mediante un ajuste fino, se opta por emplear un modelo en inglés como comparación.

Para ello, se elige el modelo Pegasus [65] el cual está especializado en numerosos conjuntos de datos, uno de los cuales es CNN/Daily Mail. Debido a que la temática es similar a la empleada en este trabajo se elige esa versión del modelo. Para poder emplear ese modelo cada nota debe ser traducida a inglés, luego resumida con el modelo y, finalmente, traducida a español. Para la traducción se emplea el servicio AWS Translate.

En cuanto a la tarea de anotación en sí, estos cinco anotadores presentaban las siguientes características:

- Todos los anotadores cuentan con estudios terciarios o superiores.
- Una de las personas es profesora de literatura y otra es licenciada en actuación, con lo cual tienen cierta formación específica.
- Una de las personas es experta en aprendizaje automático.
- Ninguna de las personas tenía experiencia previa en anotaciones de datos en el área del procesamiento del lenguaje natural.

Por otro lado, la anotación fue una tarea remunerada para dos de las cinco personas, el resto no percibió remuneración. La tarea llevó a cada persona entre 10 y 12 horas, las cuales fueron repartidas en alrededor de 2 semanas. Todos los anotadores fueron solicitados etiquetar todos los casos pero hubo 5 datos faltantes, sobre un total de 1960 anotaciones. Este número resulta de multiplicar la cantidad de

anotadores (5) por la cantidad de notas (98) por la cantidad de modelos (4). Estos anotadores se desempeñaron de manera independiente, sin comunicación entre sí.

Para entender el grado de acuerdo entre los anotadores la métrica Kappa es comúnmente usada. Sin embargo, esta métrica mide el acuerdo sólo entre dos anotadores. En nuestro caso, al tener un número mayor es necesario emplear alguna de las generalizaciones de Kappa, como el Kappa de Fleiss o el Kappa de Rudolph. El Kappa de Fleiss [19] supone fijos los marginales, es decir, supone fija la cantidad de casos que caen en cada una de las posibles categorías. En este sentido, los jueces pueden votar la categoría deseada siempre y cuando al final del estudio las cantidades totales por categoría se mantengan fijas. Por su parte, Rudolph [48] propone una alternativa libre de esta condición De este modo, cada juez es libre de votar una categoría sin otro tipo de condiciones. Por último, el Kappa de Rudolph compara el nivel de acuerdo entre los jueces contra el nivel de acuerdo que ocurriría suponiendo una distribución uniforme de las categorías. Un Kappa igual a 1 indica un perfecto acuerdo y un Kappa igual a 0 indica un nivel de acuerdo igual al ocurrido mediante el azar.

Para nuestro estudio, el Kappa de Rudolph es de 0.469, lo cual indica un nivel de acuerdo sustancialmente superior al que sería esperable por el azar. Vale la pena mencionar que el nivel de acuerdo es más alto en veracidad (0.513) y en coherencia y cohesión (0.506) que en relevancia (0.389).

Una primera observación al respecto de los resultados es que los resúmenes generados mediante Pegasus son mucho más extensos. Esto se debe a que fue ajustado en otro

conjunto de datos. Por este motivo, los resultados de este modelo no son completamente comparables ya que, al ser los resúmenes más extensos, es esperable que incluyan más información relevante. La extensión mediana de los resúmenes es de:

1. MT5 - 20 palabras
2. Beto2Beto - 19 palabras
3. Humanos - 21 palabras
4. Pegasus - 58 palabras
5. texto base - 736 palabras

En primer lugar analizamos la *coherencia y la cohesión*. En la figura 13 podemos ver un conteo de todos los puntajes obtenidos. Se puede apreciar que "humanos"tiene la mayor cantidad de 2 (el puntaje más alto). Vale la pena remarcar que ésta es la categoría en donde Humanos saca el mayor puntaje de todas las dimensiones y mayor diferencia contra los modelos. Es seguido por MT5 y, más atrás, Beto2Beto. Pegasus aparece como el que obtiene menos puntajes 2 pero también menos puntajes 0 y mayor puntaje 1. Ésto indica que, si bien las frases no están completamente bien formadas, tampoco están completamente incorrectas. El origen de esta diferencia tiene que ver con que Pegasus casi nunca genera una sola frase sino varias frases. Por este motivo, es improbable que no haya al menos una frase que esté bien formada. En cambio, los otros dos modelos y el resumen humano en muchas ocasiones tiene una sola oración y falta de coherencia o cohesión llegando así a tener mayor proporción de 0.

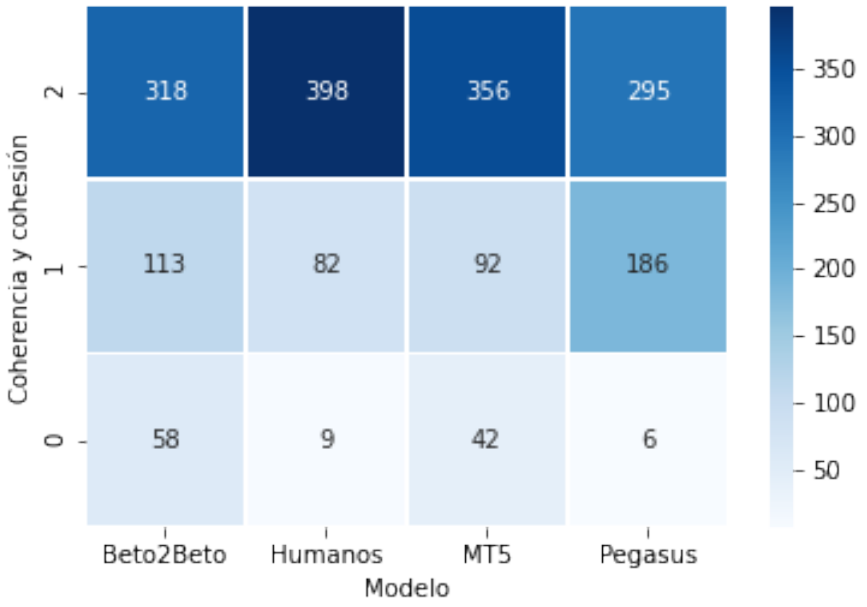


Figura 13: Coherencia y Cohesión

En cuanto a la *veracidad*, Pegasus es el modelo con mayor proporción de 2, en segundo lugar MT5 y, en tercer lugar, Humanos. Sin embargo, Humanos presenta mayor cantidad de unos que MT5, con lo cual es difícil concluir que uno es estrictamente mejor que otro. Beto2Beto está bastante más atrás que el resto, con una menor cantidad de 2 y una mayor cantidad de 0.

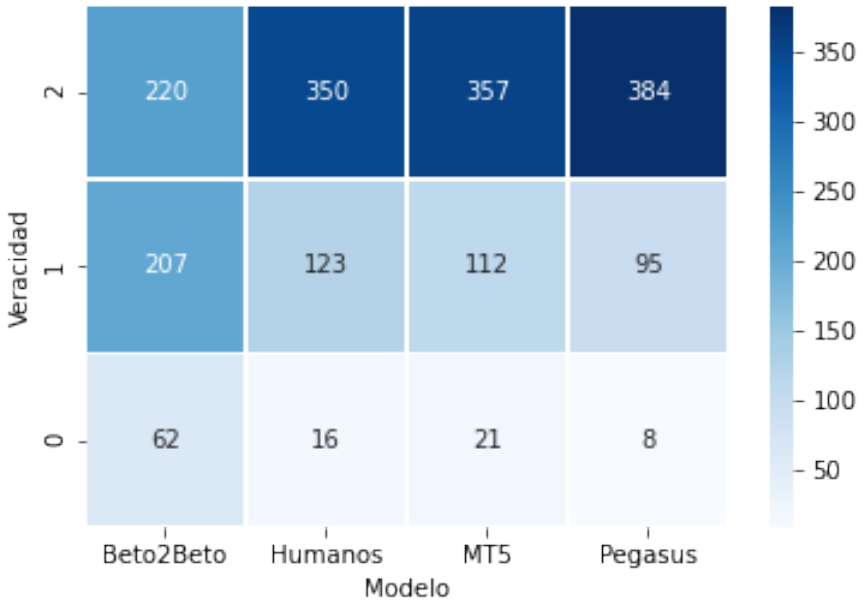


Figura 14: Veracidad

Por último, en lo que respecta a *relevancia* como es esperable Pegasus obtiene un resultado mucho mejor que los modelos y "humanos". Esto se debe, como ya se dijo, a que la extensión de los resúmenes es mucho mayor. Además, debe tenerse presente que lo que aquí llamamos Humanos, así como el objetivo que aprendieron los modelos MT5 y Beto2Beto en realidad son los subtítulos, no resúmenes propiamente dichos. Como tal, es esperable que MT5, Beto2Beto y Humanos tengan una peor relevancia en sus resúmenes. Más allá de eso, vemos que Humanos tiene mejores resultados que MT5 y éste, a su vez, que Beto2Beto.

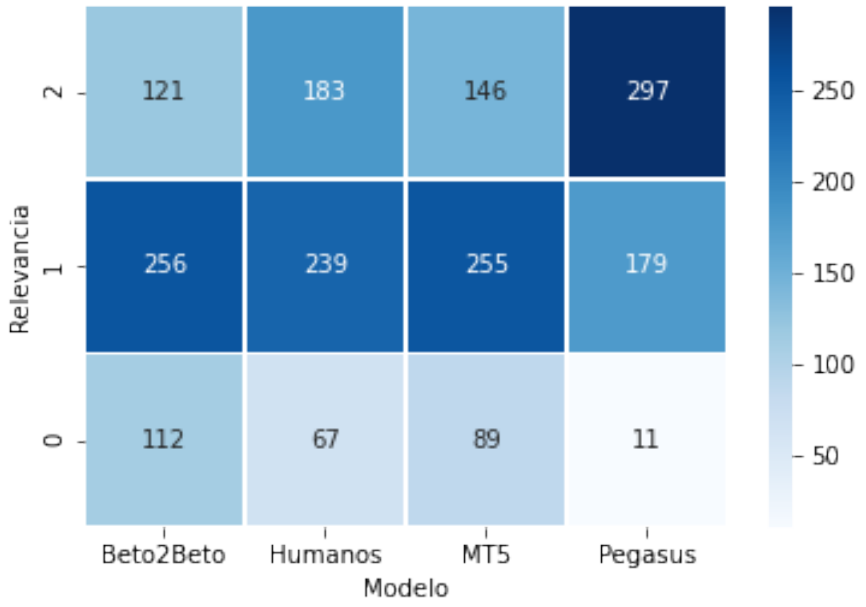


Figura 15: Relevancia

Hasta acá algunas conclusiones que se desprenden son:

1. *Humanos* obtiene el mejor resultado en coherencia y cohesión pero en segundo lugar y tercer lugar, están MT5 y Beto2Beto. Sin embargo, acá es necesario matizar este resultado, ya que si bien Pegasus obtiene una menor cantidad de 2, obtiene una mayor cantidad de 1 y una menor cantidad de 0.

2. Además, MT5 obtiene un resultado muy similar a *humanos* en relevancia (tiene una mayor cantidad de doses pero una menor cantidad de unos) pero un resultado bastante peor a *humanos* en veracidad.

3. Beto2Beto no muestra resultados comparables a los

demás en ninguno de los ejes analizados.

4. Pegasus obtiene indudablemente un mejor puntaje en relevancia pero ésto es difícilmente atribuible a que el modelo es mejor en sí sino que se debe a que fue ajustado sobre un conjunto de resúmenes más extensos y que, por lo tanto, tiende a predecir resúmenes más largos. A su vez, fue entrenado sobre un conjunto de datos de resúmenes propiamente dichos, y no de subtítulos como MLSUM.

5. Pegasus también obtiene un mejor resultado en veracidad pero ésto no se debe a la extensión del resumen sino probablemente a un mejor entrenamiento. Para poder concluir al respecto sería necesario hacer un análisis pormenorizado del objetivo con el cual se entrenó, una posibilidad es que éste tuviese gran cantidad de información, obligando al modelo a tener mayor precisión en la extracción de determinadas entidades.

Para poder tener una mayor precisión sobre los resultados se desean corroborar mediante test de hipótesis las diferencias obtenidas entre MT5 (el mejor modelo de los entrenados en este trabajo) y los resúmenes humanos (que los tomamos como benchmark). Para ello se emplea el test de los rangos con signos de Wilcoxon, el cual es un test no paramétrico y sirve para evaluar de si la distribución de las diferencias entre dos grupos es simétrica alrededor del 0. Para ello se emplea la implementación de `scipy`.

Las hipótesis planteadas y los resultados son:

1. Tomando la métrica de **coherencía y cohesión** para MT5 y Humanos:

$H_0$: la mediana de la distribución de las diferencias entre Humanos y MT5 es positiva.

$H_1$: la mediana de la distribución de las diferencias entre Humanos y MT5 es negativa.

P-valor: 0.99999

Este resultado es claramente esperable ya que Humanos obtiene un puntaje sumamente superior a MT5.

2. Tomando la métrica de **relevancia** para MT5 y "humanos":

   $H_0$: la mediana de la distribución de las diferencias entre Humanos y MT5 es positiva.

   $H_1$: la mediana de la distribución de las diferencias entre Humanos y MT5 es negativa.

   P-valor: 0.9982

   Vale la pena recordar para este caso que si bien Humanos obtiene una mayor cantidad de doses, MT5 obtiene una mayor cantidad de unos, pero esto no llega a compensar la diferencia.

3. Tomando la métrica de **veracidad** para MT5 y "humanos":

   $H_0$: la mediana de la distribución de las diferencias entre MT5 y Humanos es positiva.

   $H_1$: la mediana de la distribución de las diferencias entre MT5 y Humanos es negativa.

   P-valor: 0.520

   En este caso no podemos rechazar la hipótesis nula, sugiriendo que los resultados obtenidos entre MT5 y

humanos son relativamente similares, al menos en el conjunto de datos analizados.

En términos generales, con este conjunto de datos no se encuentra evidencia que indique que los resúmenes humanos tienen mayor veracidad que los de MT5 pero si mayor relevancia y coherencia.

Además, es de interés analizar si estos resultados se sostienen a nivel de caso. En particular, se desea saber en cuántos casos un modelo obtiene un mejor resultado que los resúmenes humanos. Para ello, se sigue el siguiente procedimiento. Para cada modelo, variable y caso se suma el total de puntajes obtenidos de los distintos jueces. Luego se compara el valor obtenido contra el resumen humano. Finalmente, se cuenta en cuántos casos obtiene un mejor puntaje el modelo (indicado por la variable 'Mejor') y en cuántos casos obtiene un mejor puntaje el humano (indicado por la variable 'Peor'). Este procedimiento sólo se realiza como complemento al análisis previo y no es de ninguna manera concluyente por sí mismo, ya que es bien sabido que la suma no es una operación válida para las variables ordinales.

En el eje vertical se ve la cantidad de casos en los que el modelo obtiene un mejor o peor resultado que "humanos". Pegasus obtiene un mejor resultado que los demás en Veracidad y Relevancia (incluso mejor que los humanos), pero obtiene un peor puntaje que MT5 (y que "humanos") en Coherencia y cohesión. Todos los modelos obtienen un peor resultado que humanos en Coherencia y cohesión. MT5 solamente supera a Humanos en el caso de Veracidad.

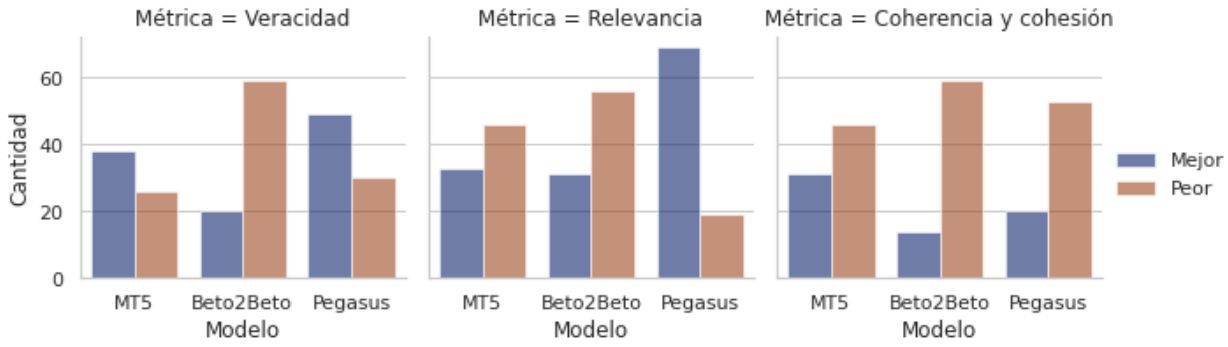


Figura 16: Modelos contra humanos

### 6.3 Discusión sobre Beto2Beto y MT5

Es interesante también preguntarse sobre la diferencia en performance entre Beto2Beto y MT5. Anteriormente se ve que ambos obtienen resultados parecidos en términos de rouge, sin embargo en las anotaciones humanas MT5 obtiene métricas sustancialmente mejores, especialmente en veracidad.

Entonces, ¿de dónde surge la diferencia de los modelos? En primer lugar, ambos son modelos Transformers que emplean la arquitectura original con diferencias menores.

En segundo lugar, Beto2Beto pesa 947 MB y tiene 248.101.146 parámetros mientras que MT5 pesa 1.12G y tiene 300.176.768 parámetros. Si bien existe diferencia, MT5 small es el modelo más parecido en cuanto a tamaño a Beto2Beto ya que MT5 también está disponibilizado en versiones mucho más grandes: Base (580M), Large (1.2 mil de millones), XL (3.7 mil de millones) y XXL(13 mil de millones).

Por otro lado, existe una gran diferencia en cuanto a los conjuntos de datos empleados en el entrenamiento de cada

modelo. Mientras que el conjunto de datos de entrenamiento de Beto consta de 3.000 millones de tokens a los que hay que sumar 1.8 mil millones de CC-NEWS-ES empleados en Beto2Beto, solamente la sección en español de MT5 incluye 433.000 millones de tokens.

Otra diferencia sustancial en los modelos es que mientras que Beto es entrenado como un **masked language model**, MT5 es preentrenado con un entrenamiento multitarea, mezclando la tarea semisupervisada llamada *i.i.d. noise, replace spans* con tareas supervisadas. Muchas de estas tareas obligan al modelo a buscar partes de la información en lugares relativamente distintos de cada texto de entrada, por ejemplo, en la tarea de resumen abstracto usando CNN/Daily-Mail. En mi opinión, tanto el uso de un conjunto de datos mucho más grande como el uso de tareas supervisadas variadas como parte del preentrenamiento son los factores que permiten que MT5 tenga un resultado sustancialmente mejor que Beto2Beto, especialmente en las métricas de veracidad y relevancia.

En cuanto a la facilidad de entrenamiento podemos decir que el proceso es similar para ambos modelos con la librería `transformers` de HuggingFace. Sin embargo, el hecho de que MT5 use prefijos para hacer ajustes finos a distintas tareas permite que, con más esfuerzo, uno puede incrementar la performance del modelo incluyendo diversas tareas en español como parte del preentrenamiento. De este modo, se mejora sobre una tarea al mismo tiempo que se agregan nuevas tareas en un mismo modelo.

### 6.4 Ejemplos de generación

En esta subsección presentamos algunos ejemplos tomados al azar de generación con los modelos entrenados. Primero, se encuentran algunos de los ejemplos de resúmenes generados sobre MLSUM y que fueron usados en la evaluación manual. Luego, se pueden observar títulos generados sobre noticias de distintos portales de noticias. Además, en el apéndice 8.3 se incluyen ejemplos generados con el modelo Beto2Beto.

En todos los casos se emplean los parámetros por default que implican, básicamente, una búsqueda *greedy* de la siguiente palabra, ya que éste es el enfoque usado al entrenar y testear. Los resultados no son generados con *beam search* ni muestreo de palabras. [6]

#### 6.4.1 Ejemplos de generación en MLSUM

A continuación se incluyen resúmenes generados por Beto2Beto y MT5 junto al resumen humano en MLSUM.

[6] Un abordaje detallado de los parámetros que implementa el paquete transformers se puede ver en https://huggingface.co/blog/how-to-generate

| Resumen humano | Beto2Beto | MT5 |
|---|---|---|
| Doce meses después de la moción de censura, sus protagonistas, que entonces dudaron, ven claro que fue un acierto¸ | la jugada que cambió la historia de españa parece ajedrez, todo calculado¸ | El presidente del Gobierno vuelve a ganar la votación de la moción de censura hace un año |
| Los Reyes presiden este sábado el desfile de las Fuerzas Armadas¸ | el desfile de las fuerzas armadas, presidido por margarita robles, se celebra este sábado en sevilla¸ | El tradicional desfile de las Fuerzas Armadas celebra el 30o aniversario de la intervención de España en operaciones de paz en el extranjero |

| Resumen humano | Beto2Beto | MT5 |
|---|---|---|
| La tradicional receta de Senegal se creó en la ciudad de Saint Louis en el siglo XIX y uniría el pescado local con el arroz de la antigua Indochina, la cebolla europea y el tomate de América importado por los colonos franceses. Se va a abrir un proceso ante la UNESCO para que sea declarado Patrimonio Cultural Inmaterial¸ | la cocinera seynabou dbiouge elabora su plato más famoso en senegal¸ | La cocinera Seynabou Dbiouge cocina en el restaurante Darou Salam de Saint Louis, en Senegal |

| Resumen humano | Beto2Beto | MT5 |
|---|---|---|
| ‘Bohemian Rhapsody’ y ‘Rocketman’ prefiguran otro olimpo para las figuras del rock: los que tienen película y los que no. En España, Fernando León prepara una serie sobre Sabina¸ | el interés por las series de televisión se dispara en estos tiempos. ¿ qué actores y series pueden desarrollar en las figuras de la música popular?¸ | El interés audiovisual en las estrellas de la música no se reduce al ámbito anglosajón, a la espera de más proyectos |
| Amadou Diaw es una persona clave en el desarrollo educativo de Senegal en el último cuarto de siglo y uno de los principales mecenas culturales de Saint Louis¸ | amadou diaw, impulsor de la revitalización de la ciudad senegalesa, reivindica la hospitalidad y la fluidez en las relaciones¸ | El promotor de la revitalización de Saint Louis, Amadou Diaw, abre cada día las puertas de su casa, situada en el sur de la isla de Saint Louis, en el barrio de Sindoné |

| Resumen humano | Beto2Beto | MT5 |
|---|---|---|
| El hijo jinete de la fallecida duquesa arremete contra su hermano Carlos y abre interrogantes sobre la situación financiera de la saga¸ | el primogénito de la duquesa de alba, carlos fitz - james stuart, se tambalea con la declaración de su hermano carlos¸ | El primogénito, Carlos Fitz-James Stuart, ha sido el primer duque de Arjona y XIV conde de Salvatierra |
| Dicen que lo quiere el pueblo. ¿Qué pueblo? Así llaman hoy a esa parte de la ciudadanía dispuesta a obedecer sin rechistar a quienes le mandan desobedecer las leyes¸ | los que se reconcilian demuestran magnanimidad, prudencia, empatía, virtudes sin duda humanistas¸ | La reconciliación es pasar de una querella pública a un acuerdo privado |

| Resumen humano | Beto2Beto | MT5 |
|---|---|---|
| Cuando has sido un personaje público muchas cosas propias de esa vida pública se quedan adheridas. Es difícil enfrentarse al día sin secretarias ni camareros¸ | este primer fin de semana de junio va a ser algo más que toros y pasar página¸ | El monarca emérito desea “dejar de desarrollar actividades institucionales a partir del 2 de junio” |
| En política no se trata de calcular lo que vas a ganar sino de tomar buenas decisiones, y ya ganarás poder real¸ | a los pactos hay que confundir a los oponentes, por impasibilidad o por gestualidad¸ | El PP tiene buenas opciones desde el bloquismo, pero también la ansiedad por mandar en Madrid |

| Resumen humano | Beto2Beto | MT5 |
|---|---|---|
| Las reformas más urgentes exigen un Gobierno sustentado por una mayoría¸ | la concre fórmula de gobierno que alcancen los partidos importa menos que garantizar la estabilidad parlamentaria¸ | La nueva legislatura no puede estar marcada por los sobresaltos y los envites parlamentarios llevados al límite |

Cuadro 5: Ejemplos de generación en MLSUM

#### 6.4.2 Ejemplos de generación en CC-NEWS-ES-titles

A continuación se incluyen resúmenes generados por Beto2Beto y MT5 entrenados en CC-NEWS-ES-titles junto al título generado por humanos. Estos ejemplos no son parte de CC-NEWS-ES-titles, por lo que no son conocidos por los modelos.

| Humano | Beto2Beto | MT5 | URL |
|---|---|---|---|
| Santiago Cafiero respondió las críticas por Formosa: “A nosotros no nos tienen que venir a decir qué hacer con los derechos humanos” | amnistia internacional pide intervencion del estado en formosa | Amnistía Internacional pide intervención del Estado frente a la situación en Formosa | url |
| Amado Boudou consiguió que le bajaran 10 meses de su condena por “estímulo educativo” y ya puede pedir salidas transitorias | sigue la discusion sobre si amado boudou debe volver a prision | El juez obliga a defender el beneficio del estímulo educativo | url |
| Entre Ríos: un grupo de pescadores se sorprendió con una especie del río Paraná | pescadores paranaenses captaron un ejemplar de manguruyu piraguazu | Otro grupo de pescadores entrerrianos sorprendieron por el tamaño de manguruyú piraguazú | url |

| Humano | Beto2Beto | MT5 | URL |
|---|---|---|---|
| Qué es y en qué consiste la Superliga europea: equipos, formato y otras claves | ¿ que es la superliga europea? | La Superliga europea, una competición elitista | url |
| La chocotorta fue elegida como el mejor postre del mundo | chocotorta, el mejor postre del mundo | La chocotorta, el mejor postre argentino de los argentinos | url |
| Cuáles son los cinco indicadores que el Gobierno busca remontar tras la derrota en las PASO y el tembladeral político | el gobierno espera que el presupuesto 2022 sea el mas robusto de los ultimos años | El Frente de Todos y la crisis política en el oficialismo | url |
| El impacto de las PASO: la economía y otros motivos para la bronca | el gobierno de alberto fernandez y el gabinete que quiere | El Gobierno deberá remontar cuestas a plena luz | url |

| Humano | Beto2Beto | MT5 | URL |
|---|---|---|---|
| Felipe Solá presentó su renuncia desde México | el sorpresivo pedido de renuncia de alberto fernandez a la cancilleria | El excanciller se enteró de su desplazamiento en El Salvador | url |
| Axel Kicillof viajó a El Calafate para reunirse con la vicepresidenta Cristina Kirchner | kicillof se reunio con cristina kirchner en un vuelo privado | El gobernador bonaerense regresa a Santa Cruz en un vuelo de línea | url |
| “Chino” Navarro: “Estábamos en caída hace 30 días, desde la foto de Sofía Pacchi” | el festejo en olivos fue la consecuencia de una crisis interna | Navarro: “No es una crisis institucional, es una crisis política” | url |

Cuadro 6: Ejemplos de generación en CC-NEWS-ES-titles

## 6.5 Disponibilización de modelos y conjuntos de datos en HuggingFace

HuggingFace es una organización enfocada en el desarrollo de NLP que cuenta con una plataforma en la cual se disponibilizan numerosos recursos, entre los cuales se destacan los modelos y los conjuntos de datos. Como parte de esta tesis se entrenaron 5 modelos y se crearon 2 conjuntos de datos, y se subieron a esa plataforma.

Los conjuntos de datos son:

- CC-NEWS-ES: https://huggingface.co/datasets/LeoCordoba/CC-NEWS-ES
- CC-NEWS-ES-titles: https://huggingface.co/datasets/LeoCordoba/CC-NEWS-ES-titles

Los modelos entrenados son:

- MT5-small-mlsum: https://huggingface.co/LeoCordoba/mt5-small-mlsum
- MT5-small-cc-news-es-titles: https://huggingface.co/LeoCordoba/mt5-small-cc-news-es-titles
- Beto2Beto: https://huggingface.co/LeoCordoba/beto2beto
- Beto2Beto-cc-news-es-titles: https://huggingface.co/LeoCordoba/beto2beto-cc-news-es-titles
- Beto2Beto-mlsum: https://huggingface.co/LeoCordoba/beto2beto-mlsum

Estos modelos pueden ser probados directamente desde la plataforma de HuggingFace (a excepción de Beto2Beto,

debido a la arquitectura). Además, los conjuntos de datos son accesibles públicamente y de manera sencilla.

Por ejemplo, el modelo **mt5-small-mlsum** se puede usar sencillamente como en el siguiente ejemplo en lenguaje Python:

```
article = """ La chocotorta, el tradicional y práctico antojo dulce de los
argentinos, fue elegida como el mejor postre del mundo por críticos de
restaurants internacionales, a casi 40 años de su creación. El ránking
Taste Atlas ubicó primero en su lista al postre insignia local de
galletitas, queso crema y dulce de leche, por delante del helado de
pistacho italiano y la tarta alemana de manzana. \Este postre argentino
sin hornear fue influenciado por la cocina italiana y se inspiró en el
famoso tiramisú italiano. Está elaborado con tres ingredientes básicos
argentinos: galletas de chocolate, dulce de leche y queso crema",
explica la página web que exhorta a los turistas de todo el mundo a
que prueben la chocotorta. En la votación, superó también a los waffles
belgas y el zserbó húngaro. A nivel local le sigue el alfajor, con
4,2 puntos contra los 4,7 de la torta. En el texto que acompaña al
listón dorado de \postre número uno", los expertos enseñan además cómo
se hacen las chocotortas, paso por paso. \Las galletas se ablandan en
leche y se cubren con una combinación de queso crema y dulce de leche.
Las formas de la chocotorta pueden variar, mientras que las galletas
se pueden remojar con leche con chocolate, café o incluso licor de
café", detallan. Por último, adjudican su creación a una \campaña de
márketing"  diseñada para promover las galletitas icónicas que le dan
su nombre. La chocotorta, infaltable en los cumpleaños argentinos, fue
creada en 1982 por una creativa de las agencias más importantes del
país, Marité Mabragaña."""

from transformers import pipeline

summarizer = pipeline("summarization", model="LeoCordoba/mt5-small-mlsum")

summarizer(article, min_length=5, max_length=64)

## Resultando en:
 [{'summary_text': 'El ránking Taste Atlas ubicó primero en su lista al
 postre  insignia local de galletitas, queso crema y dulce de leche'}]
```

Además, se puede acceder a los conjuntos de datos como se muestra a continuación:

```
from datasets import load_dataset

# cargamos el dataset CC-NEWS-ES-titles, el argumento 'split' es optativo
ccnews_titles = load_dataset("LeoCordoba/CC-NEWS-ES-titles", split="train")

print(ccnews_titles)

## Lo cual muestra:

Dataset({
    features: ['text', 'output_text'],
    num_rows: 370125
})

print(ccnews_titles[0])

# A continuación se puede ver un diccionario, reducido por simplicidad
{'output_text': 'MPT reconoce ardua labor de efectivos (...)',
 'text': ', se designó a la virgen Santa Rosa de Lima, como
 su patrona (...)'}


# De manera similar podemos acceder a CC-NEWS-ES
# Nótese que como 'name' se puede pasar el dominio del país deseado
ccnews = load_dataset("LeoCordoba/CC-NEWS-ES", name="py")

## Con print podemos ver la siguiente información
print(ccnews)

DatasetDict({
    train: Dataset({
        features: ['country', 'text', 'id'],
        num_rows: 30651
    })
})
```

### 6.6 Discusión en torno a la calidad de los modelos

Como se vio en la subsección *Evaluación con anotaciones*, MT5 obtuvo mejores resultados que Beto2Beto. En esta subsección vamos a rever los resultados del modelo pero con el foco puesto en la comparación con los resúmenes humanos. Para ello analizaremos distintos tipos de fortalezas y de errores.

En primer lugar, tomando las anotaciones humanas y ordenando de mayor a menor diferencia en puntaje, en la dimensión de relevancia podemos encontrar un caso en el que MT5 parece haber generado un resumen sustancialmente más relevante que el humano.

- MT5: El monarca emérito desea "dejar de desarrollar actividades institucionales a partir del 2 de junio"
- Humano: Cuando has sido un personaje público muchas cosas propias de esa vida pública se quedan adheridas. Es difícil enfrentarse al día sin secretarias ni camareros
- Enlace a la nota original

Es interesante mencionar también que en ese caso ni el título ni el subtítulo ni la imagen que acompañan son alusivos a que el rey emérito deja sus actividades institucionales. Esto parece indicar cierto sesgo editorial en no revelar toda la información en los titulares o, tal vez, cierta inclinación estética. En cualquier caso, en cuanto resúmen MT5 parece haber logrado un mejor resultado.

En segundo lugar, también entre los casos mejor puntuados en relevancia podemos mencionar el siguiente:

- MT5: El futbolista José Antonio Reyes, de 35 años, ha fallecido este sábado en un accidente de tráfico
- Humano: El jugador del Extremadura y ex del Sevilla, Madrid y Arsenal fallece en accidente de tráfico en la autovía A-376, que une Sevilla a Utrera. El jugador iba acompañado de su primo Jonathan Reyes, que también ha perdido la vida, y de Juan Manuel Calderón, que ha sido hospitalizado de extrema gravedad
- Enlace a la nota original

En este caso, si bien en una primera mirada MT5 parece ser más informativo que el humano, la diferencia de puntaje en este caso viene del hecho de que el subtítulo no contiene toda la información relevante del titular. Por el contrario, el título de esa noticia es *Los equipos de José Antonio Reyes*, lo que incluye el nombre, que es justamente la pieza de información faltante en el subtítulo. Esto mismo se complementa también con la imagen de la noticia.

Esta situación nos habla de la complejidad de la evaluación y de cómo, a veces, la construcción de un conjunto de datos puede ocultar determinadas decisiones. En este caso, la elección del subtítulo sin el título podría haber quitado parte relevante de la información, penalizando los resúmenes humanos, por un lado, y complejizando el aprendizaje de la tarea, por otro.

En tercer lugar, el caso donde MT5 obtuvo un peor resultado relativo al resumen Humano es el siguiente:

- MT5: El ministro de Finanzas asegura que la Dirección Central no tenía conocimiento de estas acciones
- Humano: Inspectores tributarios dan el alto a conductores con vehículos de lujo para revisar si tienen cuentas pendientes
- Enlace a la nota original

En este caso puede verse que MT5 genera una frase bien formada, con coherencia y cohesión y que además capta algo que está contenido en el texto y es, por lo tanto, verídico. Sin embargo, no logra captar el elemento central de la noticia, que hay un operativo de tránsito con fines recaudatorios.

Un cuarto y último caso para destacar la dificultad de analizar la relevancia es el siguiente:

- MT5: Los sondeos a pie de urna indican que se revirtió otra racha: la baja movilización juvenil, de solo el 28 % en 2014
- Humano: Las nuevas generaciones rompen con los partidos tradicionales en varios países y siguen dos caminos dispares: el ecologismo o el ultranacionalismo
- Enlace a la nota original

Esta noticia es mucho más larga y más compleja que la mayoría de las demás. Cuenta con una gran cantidad de cifras y con información de distintos países. Por un lado es probable que no toda la noticia haya podido entrar en el modelo, lo cual de por sí es una limitación. En segundo

lugar, debido a la estructura de pirámide invertida de las noticias es probable que el modelo haya intentado buscar lo más relevante en las primeras secciones, cuando en este caso se necesitaba una abstracción mucho mayor. Forzar un resumen más abstracto para evitar simplemente buscar en el texto de entrada es un desafío técnico que probablemente requiera un mejor corpus de entrenamiento o mejores técnicas de muestreo en ese corpus.

Sin embargo, las dificultades para generar buenos resúmenes no son tan peligrosas como un texto directamente falso. Por ejemplo, el siguiente:

- MT5: El secretario estadounidense Patrick Shanahan asegura que el gigante asiático puede dominar el gigante asiático
- Humano: Los titulares de Defensa de ambos países se reúnen en un foro de Seguridad en Singapur
- Enlace a la nota original

Si bien la frase esta mal formada, lo peligroso de la misma es que parece indicar que Patrick Shanahan aseguró que China (el gigante asiático) podría dominar algo, cuando lo que en verdad dijo es que *Ninguna nación puede,o debería, dominar el Indo-Pacífico.*

Por otro lado, entre los resúmenes humanos se destaca como uno de los peores puntuados en veracidad el siguiente:

- En este relato hay un viaje espacial, una esposa harta de no ver a su marido y una barra libre abierta las 24 horas para colmar los excesos de dos rockeros

- Enlace a la nota original

Ese resumen en cuestión es incorrecto y por dos motivos. Por un lado, porque no se hace mención a que el exceso de alcohol sea un problema en los rockeros sino el exceso de drogas. Por otro lado, porque en rigor el problema descripto no es de dos rockeros sino de uno solo, Elton John. Se menciona a su vez a su compañero, Bernie Taupin, pero no parecería haber tenido problemas de excesos con el alcohol ni con las drogas, según la nota.

Por último, en lo que respecta a la coherencia y cohesión, en algunos casos MT5 genera repeticiones sobre un o varios gramas. Si bien esto puede parecer problemático es, sin lugar a dudas, el inconveniente más fácil de resolver ya que se puede identificar algorítmicamente y usar otra de las ramas en la búsqueda.

Dicho lo anterior, ¿qué condiciones hacen falta para que estos modelos puedan funcionar en el mundo real?

Por un lado creo que es necesario definir un mejor conjunto de datos, el cual debe estar orientado a la línea editorial en cuestión. Por ejemplo, puede ser más informativa, más literaria o más orientada al *clicbait*. En este último caso, cierta información se mantiene deliberadamente oculta del titular para que quien lo lea sienta la necesidad de entrar a la nota. A su vez, deberían generarse simultáneamente título y subtítulo, entendiendo que la información que no está contenida en uno puede estarlo en el otro.

En segundo lugar, entendiendo que la generación del texto puede volverse repetitiva o falta de coherencia, sería necesario tener algún método de contingencia, basado en alguna lógica que analice la estructura del texto.

En tercer lugar, creo que la tarea más difícil es mejorar la veracidad de los modelos. Debido a que el modelo extrae información del texto que se le ingresa, puede ocurrir que establezca una relación errónea relacionada a una entidad del texto, por ejemplo, una fecha, una persona o un lugar puede ser intercambiado por otra mencionada en el mismo texto, generando la ilusión de que la frase es correcta cuando no lo es. Para mejorar la calidad en este aspecto es probable que sea necesario emplear una métrica distinta al rouge.

Por lo expuesto anteriormente, creo que los primeros dos puntos son realizables en el corto plazo, sin tanta nueva investigación pero el tercero puede aún ser difícil de alcanzar, al menos en el idioma español. Pegasus obtiene resultados mucho mejores en este aspecto, una hipótesis es que fue entrenado en un corpus con mucha más información en el objetivo a predecir pero su análisis excede a este trabajo.

Finalmente, algunas consideraciones adicionales sobre los humanos como generadores de resúmenes y como evaluadores de los mismos.

Como se mencionó previamente qué es un buen resumen no es algo sencillo de evaluar, sobre todo ya que ésto depende de la línea editorial. En algunos casos hay información que ya se supone conocida, por ejemplo, véase la siguiente noticia:

- Título: Messi: “Contra el Liverpool cometimos errores boludos”
- Subtítulo: El delantero argentino asegura que La Liga echa de menos a Cristiano y cuenta que todavía sigue

en contacto con Neymar

- Enlace a la nota original

La noticia comienza hablando sobre la derrota que sufrió el Barcelona ante Liverpool, la cual es de gran envergadura porque se presuponía que las condiciones estaban dadas para que el primero avanzase a la siguiente ronda. Este contexto, que es clave, se desconoce en el título, y el subtítulo se centra en las declaraciones sobre Cristiano y Neymar.

Ahora bien, ¿cómo evaluar este caso? Desde el punto de vista de la información, cierta parte importante está ausente. Por otro lado, los lectores de la sección de deportes de este diario tal vez ya cuentan con esa información porque, por ejemplo, ya aparece en otra noticia ubicada más arriba en el portal.

Entonces qué es lo relevante a incluir no siempre es fácil de evaluar, esto lo vimos en el nivel de acuerdo, ya que el Kappa para relevancia es mucho más bajo que el promedio, 0.389 contra 0.469. Desde este punto de vista podemos decir que es más fácil evaluar la calidad de los resúmenes en las demás dimensiones: veracidad, y coherencia y cohesión.

Por otro lado, debido a que Pegasus está especializado en resúmenes (mucho más informativos que los título o subtítulos), éste obtiene mejores puntajes que los propios humanos en relevancia.

Para concluir, podemos decir que las anotaciones humanas son útiles para evaluar los modelos principalmente en cuanto a la veracidad, y la coherencia y cohesión pero en cuanto a la relevancia es posible que sea necesario reducir el dominio de lo que es relevante y dar una indicación o

instrucción más precisa de qué se espera. Esta indicación o instrucción podría permitir los sesgos propios de la línea editorial que se quiere generar, como habilitar a que cierta información se presuponga, se deje solamente en el título, o que se espere una imagen que complemente el texto. En este sentido, concluimos del mismo modo a [26], allí muestran que la detección de contenido relevante varía fuertemente según si se restringe o no la definición de relevancia. Para ello realizan un experimento en donde distintos anotadores tienen que escribir un resumen, en un caso sin restricciones, y en otro caso, respondiendo tres preguntas.

En otras palabras, debido a que la pregunta sobre qué es un buen resumen es altamente subjetiva, es esperable encontrar un nivel de acuerdo más bajo que en otras tareas. Una forma de mitigar ésto es reduciendo fuertemente el dominio del problema mediante una definición clara de qué se espera que incluya el resumen, por ejemplo, exigiendo que responda ciertas preguntas. Aún así, no está claro qué tanto se puede reducir esta subjetividad, por la naturaleza misma del problema.

Una posible extensión del modelo MT5 sería entrenar distintos prefijos con distintos estilos de títulos y/o subtítulo. Así también, sería posible dar distintos estílos según la sección del diario: deportes, política, economía, etc.

# 7 Conclusiones

En el presente trabajo se aplican métodos para entrenar modelos de resumen abstracto en español. Estos métodos están basados en la arquitectura conocida como Transformer. La base teórica detrás de esta arquitectura y otras posteriores están desarrolladas en las secciones Marco teórico y Modelos preentrenados y ajuste fino. Allí se detalla la diferencia entre modelos preentrenados y la especialización de esos modelos en tareas puntuales, este último proceso conocido como ajuste fino.

A partir de lo visto allí, en la sección Metodología se plantean dos técnicas para entrenar modelos de resumen en español. La primera de ellas es inicializar un modelo encoder-decoder basado en Beto (un modelo Bert entrenado en español), continuar el preentrenamiento en datos similares (usando el conjunto CC-NEWS-ES) y, finalmente, ajustar en las tareas definidas. El segundo enfoque es emplear MT5 haciendo un ajuste fino del modelo a las tareas. Además, las tareas definidas son dos: la predicción de títulos a partir del conjunto CC-NEWS-ES-titles y la predicción de los resúmenes de la sección en español de MLSUM, que surgen de el diario El País de España.

Los resultados obtenidos son medidos con la métrica rouge y sus variantes. Según esta métrica podemos ver que Beto2Beto obtiene un mejor resultado que MT5 pero ambos tienen un mejor resultado que mBert que es el modelo usado en MLSUM como benchmark. Sin embargo, como se ve en la sección Evaluación de modelos, las métricas automáticas no reflejan fielmente la calidad de los resúme-

nes. Por este motivo, complementamos el análisis con una evaluación basada en anotaciones humanas y comparamos Beto2Beto, MT5 y el resumen humano usando datos de MLSUM. Como benchmark se usa el modelo Pegasus, el cual está entrenado en un conjunto de datos en inglés, con lo cual es necesario primero traducir de español a inglés y luego de vuelta de inglés a español. Además, está entrenado con un resúmenes bastante más largos lo que afecta en parte algunos resultados.

Del análisis basado en anotaciones humanas podemos decir que MT5 sólo obtiene resultados comparables a los humanos en veracidad. Tanto en coherencia y cohesión como en relevancia queda relegado contra los humanos. Pegasus obtiene un mejor resultado aunque no totalmente comparable en relevancia y un mejor resultado en veracidad. Beto2Beto queda muy detrás en los tres ejes de análisis.

Evaluar la relevancia de un resumen es una tarea altamente subjetiva y, sin una adecuada delimitación de la misma, el grado de acuerdo entre anotadores se puede ver afectado. En este trabajo, vemos que el nivel de acuerdo entre anotadores con respecto a la relevancia es menor que con respecto a la veracidad y a la coherencia y cohesión. Para una futura línea de investigación, se recomienda acotar la evaluación de los resúmenes mediante preguntas claras que deban ser respondidas.

La diferencia en los resultados entre MT5 y Beto2Beto es mayor en veracidad que en coherencia y cohesión. Una posible explicación sobre esta diferencia es que MT5 usa un entrenamiento multitarea con objetivos supervisados y semi-supervisados durante el preentrenamiento. Además, el

conjunto de datos usado es mucho mayor al de Beto2Beto (unos 2 órdenes de magnitud de diferencia).

Como parte de este trabajo y en línea con el tercer objetivo que nos propusimos, se entrenan 5 modelos y se construyen dos conjuntos de datos, todos disponibles en la plataforma de HuggingFace. El entrenamiento de estos modelos implica un nivel de esfuerzo y costos razonable, gracias al empleo de los paquetes disponibilizados por HuggingFace, así como también gracias al empleo de cómputo en AWS.

Como líneas de investigación futura se puede mencionar el desarrollo de modelos basados en aprendizaje semi-supervisado. En este enfoque se construye un problema supervisado a partir de información que no está etiquetada, es decir, que no está manualmente curada. Por ejemplo, en [17] se aprende a generar un resumen abstracto de opiniones sobre Yelp y sobre Rotten Tomatoes tomando como target un comentario (de los más valorados) y como entrada los 3 comentarios más parecidos al elegido en el objetivo. De este modo se obtienen excelentes resultados sin información etiquetada. Pueden verse otras aproximaciones semi-supervisadas a PLN en Marge [30] o en [6] y en [62]. Por otro lado, los conjuntos de datos disponibilizados permiten otras líneas de trabajo posibles. Por ejemplo, con CC-NEWS-ES es posible realizar un análisis de tópicos por país, comparando los temas que en cada país se tratan. Siguiendo con esta idea, es posible realizar un análisis de sesgos y comparar los sesgos entre los distintos países, a partir de entrenar un embedding por país.

### 7.1 Consideraciones éticas

En esta sección describimos los riesgos éticos implicados por los conjuntos de datos creados durante este trabajo y los modelos, siguiendo los lineamiento establecidos por ACL. En primer lugar, los dos conjuntos de datos abiertos se basan en el mismo origen: el índice de noticias de Common Crawl. Como se mencionó previamente, este índice no es el mismo que se usa habitualmente, por ejemplo, como parte de C4. Dicho ésto, vale aclarar que no se usaron anotaciones manuales como parte de la construcción de estos conjuntos de datos.

Este índice se basa en la extracción de noticias a partir de las suscripciones a los RSS de las webs y también basándose en los mapas de los sitios de noticias. En el primer caso, está claro que el portal de noticias está intentado facilitar el acceso programático mediante el establecimiento de un protocolo RSS, sin embargo, en el segundo caso no es obvio que el acceso programático de terceros sea algo buscado. Si bien han habido algunas discusiones sobre la legalidad o ilegalidad de extraer las páginas webs de manera automática, el caso de Common Crawl no parece ser sustancialmente distinto al índice que construye Google, solamente que en un caso la información se disponibiliza publicamente y en otro no. Dicho ésto, no se han podido encontrar antecedentes de acciones legales contra Common Crawl que argumenten algún accionar indebido. Por otro lado, Common Crawl tiene publicados los términos de uso y, a su vez, describen algunos aspectos técnicos relevantes como, por ejemplo, que honran los archivos **robotos.txt**,

en los cuales se puede establecer la velocidad de consumo de la información (para evitar enlentecer la página en cuestión), así como también los meta-tag **NOFOLLOW** del mapa del sitio.

Como fue mencionado anteriormente, el índice de noticias de Common Crawl no consta con un atributo con el idioma. Por este motivo, se usó un proceso que selecciona los párrafos en base un modelo que predice el idioma. Es probable que esto haya introducido algún sesgo contra los textos bilingües o en los que se emplean términos en inglés dentro de un texto mayormente en español.

Como es esperable, siendo un corpus de noticias es posible encontrar distintos tipos de sesgos, como sesgos de género y de raza, y también es posible hallar expresiones discriminatorias o potencialmente dañinas. Además, este tipo de expresiones es probable que varíen según la línea editorial según la cual se redactó el texto en sí.

Por este motivo, es recomendable que si se usaran estos datos para entrenar un modelo con una aplicación tal que los sesgos de éstos pudieran reproducirse o exacerbarse, se considere aplicar una serie de preprocesamientos a fin de reducir los mismos. Por ejemplo, se podrían definir reglas para eliminar noticias enteras o párrafos. Estas reglas tendrían por fin identificar expresiones potencialmente maliciosas para eliminarlas.

Otra observación relacionada a las noticias es que los títulos y el texto en sí puede tener una organización que imprime su propio sesgo a los modelos de lenguaje. Por un lado, los títulos están muchas veces escritos de modo que incitan al lector a hacer clic, con lo cual no siempre fun-

cionan como un resumen efectivo. En segundo lugar, los cuerpos de las noticias pueden estar organizados según lo que se llama una **pirámide invertida**, es decir, la mayor parte de la información se encuentra al comienzo y luego se completa la noticia con detalles. Ambas características pueden o no afectar el rendimiento del modelo que se entrene con estos datos pero vale la pena analizar las posibles implicancias.

Vale la pena agregar también una observación técnica. La extracción del cuerpo y de los títulos se basó en tags html. Para el cuerpo se usó el tag **body** y para el título el tag **h1**. Esta elección parece ser en términos generales correcta pero, por supuesto, podría fallar.

En cuanto a los modelos entrenados como parte de este trabajo, es preciso mencionar que sus sesgos no fueron analizados. Con lo cual, antes de emplearlos en una aplicación real es altamente recomendable evaluar los mismos atendiendo con especial cuidado los grupos menos representados o con sesgos potencialmente perjudiciales. Para ello, se sugiere establecer un conjunto de casos de evaluación, donde se pueda analizar el comportamiento del modelo en torno a distintos ejes, como valores, profesiones, oficios, etc.

Por último, en este trabajo se analizan otras características de los modelos, entre las cuales se destaca como potencialmente perjudicial la veracidad, ya que en muchas ocasiones los modelos generan resúmenes verosímiles pero no verídicos.

## Referencias


[1] Word2vec tutorial part 2 - negative sampling, Jan 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2017.

[5] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab,

Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2021.

[6] Arthur Bražinskas, Mirella Lapata, and Ivan Titov. Unsupervised opinion summarization as copycat-review generation, 2020.

[7] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based $n$-gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992.

[8] Ronald Cardenas, Matthias Galle, and Shay B. Cohen. Unsupervised extractive summarization by human memory simulation, 2021.

[9] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

[10] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[12] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *27th Annual Meeting of the Association for Computational Linguistics*, pages 76–83, Vancouver, British Columbia, Canada, June 1989. Association for Computational Linguistics.

[13] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention, 2019.

[14] Jacob Devlin. Multilingualbert, 2018.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[16] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation, 2019.

[17] Hady Elsahar, Maximin Coavoux, Matthias Gallé, and Jos Rozen. Self-supervised and controlled multi-document opinion summarization, 2020.

[18] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA, 1998.

[19] J.L. Fleiss et al. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[20] Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15, San Diego, California, June 2016. Association for Computational Linguistics.

[21] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend, 2015.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[23] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.

[24] Chip Huyen. Evaluation metrics for language modeling. *The Gradient*, 2019.

[25] Kåre Jensen and Søren Riis. Self-organizing letter code-book for text-to-phoneme neural network model. pages 318–321, 01 2000.

[26] Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Neural text summarization: A critical evaluation, 2019.

[27] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining, 2019.

[28] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

[29] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[30] Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. Pre-training via paraphrasing, 2020.

[31] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[32] Xuan Liu, Di Cao, and Kai Yu. Binarized lstm language model. pages 2113–2121, 01 2018.

[33] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation, 2020.

[34] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[35] Alexandra Sasha Luccioni and Joseph D. Viviano. What's in the box? a preliminary analysis of undesirable content in the common crawl corpus, 2021.

[36] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online, July 2020. Association for Computational Linguistics.

[37] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one?, 2019.

[38] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[40] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics, 2005.

[41] Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-Scale Knowledge Extraction*, AKBC-WEKEX '12, page 95–100, USA, 2012. Association for Computational Linguistics.

[42] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[43] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014.

[44] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[45] A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[46] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[47] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[48] Justus J. Randolph. Free-marginal multirater kappa (multirater k[free]): An alternative to fleiss' fixed-marginal multirater kappa. 2005.

[49] Søren Riis and Anders Krogh. Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *Journal of computational biology : a journal of computational molecular cell biology*, 3:163–83, 02 1996.

[50] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Leveraging pre-trained checkpoints for sequence generation tasks, 2020.

[51] Piotr Rybak, Robert Mroczkowski, Janusz Tracz, and Ireneusz Gawlik. Klej: Comprehensive benchmark for polish language understanding, 2020.

[52] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. Mlsum: The multilingual summarization corpus, 2020.

[53] Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview, 2020.

[54] Noam Shazeer. Glu variants improve transformer, 2020.

[55] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[56] Wenyi Tay, Aditya Joshi, Xiuzhen Zhang, Sarvnaz Karimi, and Stephen Wan. Red-faced ROUGE: Examining the suitability of ROUGE for opinion summary evaluation. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 52–60, Sydney, Australia, 4–6 December 2019. Australasian Language Technology Association.

[57] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline, 2019.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[59] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020.

[60] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

[61] Lilian Weng. The transformer family. *lilianweng.github.io/lil-log*, 2020.

[62] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training, 2020.

[63] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention, 2016.

[64] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer, 2021.

[65] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2020.

[66] Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. The United Nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3530–3534, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

# 8 Apéndice

## 8.1 La representación del lenguaje mediante embeddings de palabras

La idea de entrenar un *embedding* como forma de representar una palabra en un espacio vectorial 'manejable' cobra mucha fuerza especialmente con el desarrollo de modelos entrenados sobre corpus masivos. El primero de éstos modelos, Word2Vec [38], es entrenado sobre Google News. A este modelo le siguen Glove, entrenado sobre Wikipedia, Gigaword, Twitter y CommonCrawl [43] y FastText, entrenado en Wikipedia y CommonCrawl [4].

### 8.1.1 Word2Vec

En 2013 la disciplina del procesamiento del lenguaje natural da un salto con el algoritmo conocido como *Word2Vec* [38]. Con la publicación de este modelo se disponibilizan los parámetros entrenados de varios modelos en un corpus muy grande (al menos para ese entonces) de Google News.

En este trabajo se plantean dos arquitecturas nuevas que permiten un entrenamiento más rápido y con mejores resultados que las arquitecturas previas.

Una de estas arquitecturas es la llamada *Continuous Bag of Words* o *CBox* y es similar al modelo en [3] pero se elimina la capa oculta. Se toman como entrada los vectores one-hot de cada palabra, luego se los agrega y se los pasa a la capa que los proyecta. Esta capa es el *embedding* y equivale a la $C$ en el modelo anterior. Es de dimensión $|V|.m$ donde $|V|$ es la cantidad de palabras distintas y $m$ la dimensión del embedding. En el trabajo original se prueba

el entrenamiento con embeddings de un $m$ de 100, 300 y 600.

Vale la penar notar que el embedding es compartido entre todas las palabras, haciendo que los vectores de éstas sean promediados sin importar la posición de cada término en la frase. En este caso se usan palabras anteriores y posteriores indistintamente, y el objetivo en CBow es predecir la palabra del medio de una ventana de ancho fijo. A las palabras previas y posteriores se las conoce como el contexto.

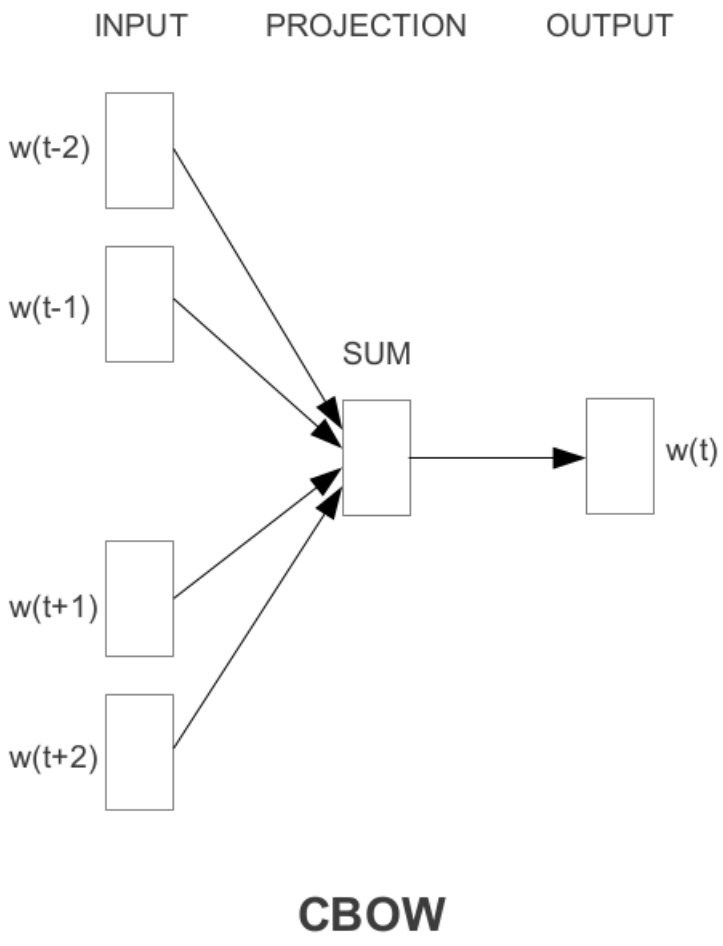


Figura 17: CBow architecture

La segunda arquitectura propuesta es el modelo *skip-gram*. En este caso, dada una oración se toma una palabra y se intenta predecir otra palabra tomadas de esa misma frase, empleando una arquitectura muy similar, donde el vector de entrada pasa a la capa de proyección (o embedding) y luego se pasa a la capa de salida.

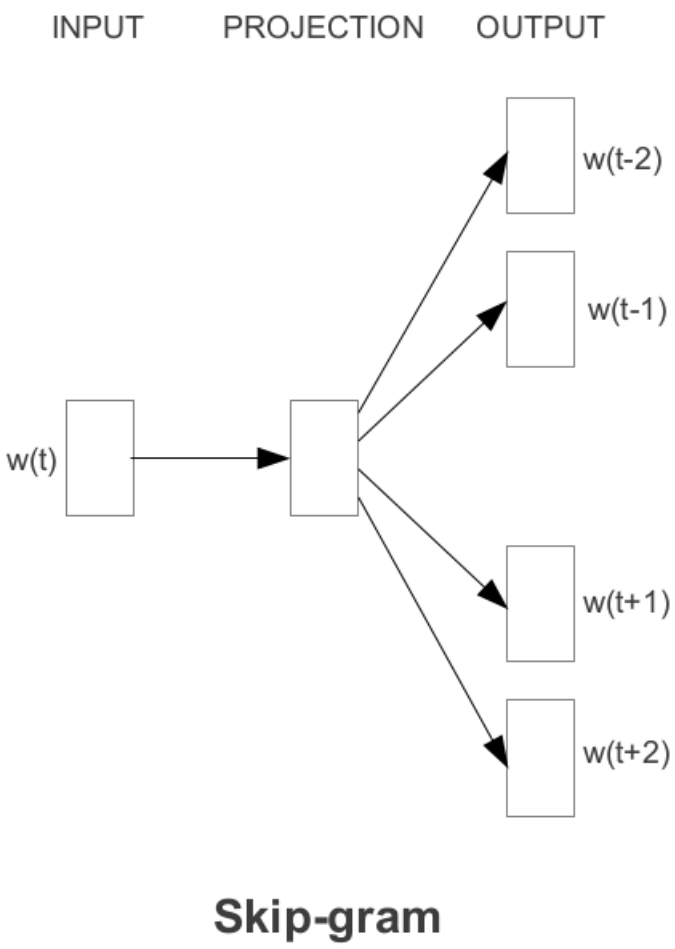


Figura 18: Skip-gram architecture

En la capa de salida vez de usar softmax se usa hierarchical softmax [40].

Por otra parte, en este trabajo se plantea una nueva forma de presentar el resultado de los embeddings. En vez de presentar las palabras más similares a una determinada palabra, se propone una nueva tarea. Ésta consiste en tratar de adivinar la mejor analogía para distintos tipos de rela-

ción bajo la pregunta "¿qué palabra es similar a C, en el mismo sentido (o en la misma proporción) en que lo son A y B?". Por ejemplo, "¿qué palabra es similar a 'Berlin' en el mismo sentido en que 'Francia' y 'Paris' son similares? La respuesta esperada en este caso es Alemania.

Para responder esta pregunta se pueden emplear los vectores del embedding haciendo la siguiente operación:

$$X = vector("biggest") - vector("big") + vector("small") \tag{8.1}$$

Luego, resta encontrar la palabra correspondiente al vector más similar a $X$, y con eso tendremos la respuesta al problema.

Una manera gráfica de entender esta tarea es realizando una proyección de los embeddings sobre un espacio de 2 dimensiones para graficarlo. En [39] se realiza eso, empleando PCA y tomando los primeros dos componentes para generar la Figura 19.

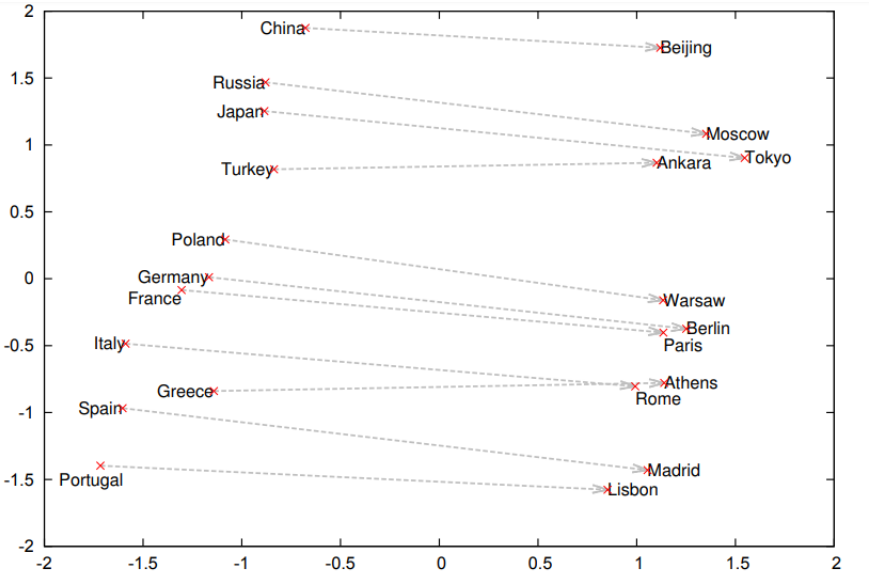


Figura 19: Vectores proyectados con PCA

Además, se presenta un metodo de evaluación basado en relaciones semánticas y sintácticas llamado Semantic-Syntactic Word Relationship test set. Presenta las siguientes subtareas de analogías:

- Ciudades capitales comunes
- Todas las ciudades capitales
- Monedas
- Ciudad-en-estado
- Hombre-Mujer
- Adjetivo a adverbio
- Opuestos
- Comparaciones

- Superlativos
- Gerundios
- Gentilicios de nacionalidades
- Pretéritos
- Sustantivos plurales
- Adjetivos plurales

Las tareas para cada uno de esos grupos consiste en encontrar la palabra que cumple con una determinada analogía. Por ejemplo, en la tarea de pretéritos se desea conocer qué palabra es similar a 'nadando' en la misma manera en que lo son 'caminando' y 'caminó'. La respuesta a la analogía se considera correcta sólo si corresponde exactamente a la palabra esperada, los sinónimos se consideran incorrectos.

En [20] se propone un nuevo conjunto de testeo de analogías en el cual se excluyeron (en parte) homónimos (ya que presentan dificultades para ser evaluados), se trató de excluir la mayor cantidad posibles de palabras evidentemente ambiguas y se acepta más de una respuesta correcta posible, basándose en WordNet [18].

#### 8.1.2 Extensiones a Word2Vec

En [39] los autores realizan varios avances que se comentan en esta sección.

Hay determinadas palabras que aparecen más frecuentemente en los corpus que otras. Por ejemplo, en un modelo

skip-gram la coocurrencia de *Argentina* y *Mendoza* probablemente sea de más ayuda que la coocurrencia de *Argentina* y *la*, ya que *la* aparece muchísimas más veces y con muchas más palabras. De la misma forma, la representación de *la* no debería cambiar mucho por estar presente un poco menos en el conjunto de entrenamiento. Por este motivo, para balancear el conjunto de entrenamiento se realiza un proceso llamada **subsampling de palabras**, en el cual cada palabra tiene una probabilidad de ser eliminada de:

$$P\left(w_i\right) = 1 - \sqrt{\frac{t}{f\left(w_i\right)}} \tag{8.2}$$

Donde $f(w_i)$ es la frecuencia de la palabra $w_i$ y $t$ es una constante, generalmente cerca de $10^{-5}$. Esta fórmula tiene la ventaja de disminuir fuertemente las palabras más frecuente que el punto de corte pero mantiene el ranking de la frecuencia de las palabras.

Una segunda extensión al algoritmo original de Word2vec es **Negative sampling**. En cada paso del entrenamiento del modelo se compara el vector predicho con el vector esperado. El vector predicho consta de un valor de probabilidad para cada una de las palabras de $V$ mientras que el vector esperado consta de un 1 para la palabra observada y 0 para cada otra palabra. Ahora bien, en cada paso se actualizan los pesos de **todas** las palabras. Con negative sampling, en cambio, se toma otra estrategia: se define un número $k$ (de 5 a 20 en conjuntos de datos chicos y de 2 a 5 en conjuntos de datos más grandes) de palabras negativas que se usarán para evaluar. Es decir, sólo se evalua el valor positivo y $k$ valores negativos. La ventaja de ésto es que sólo una

pequeña parte del total de parámetros serán actualizados, ahorrando muchísimo costo computacional.

Para elegir las muestras negativas se toman al azar $k$ elementos. Si ésto se hiciera en base a la distribución de los datos la distribución sería:

$$P\left(w_i\right) = \frac{f\left(w_i\right)}{\sum_{j=0}^{n}\left(f\left(w_j\right)\right)} \tag{8.3}$$

Se realiza un cambio que disminuye un poco a las palabras más frecuentes, quedando definida la distribución de la siguiente manera [1]:

$$P\left(w_i\right) = \frac{f\left(w_i\right)^{3/4}}{\sum_{j=0}^{n}\left(f\left(w_j\right)^{3/4}\right)} \tag{8.4}$$

El valor de 3/4 es definido empíricamente.

De esta forma, si en el modelo skip-gram la función objetivo original es:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p\left(w_{t+j}|w_t\right) \tag{8.5}$$

Siendo:

$$p\left(w_O|w_I\right) = \frac{\exp\left(v'^{\top}_{w_O} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left(v'^{\top}_{w} v_{w_I}\right)} \tag{8.6}$$

Con negative sampling:

$$\log\sigma\left(v'^{\top}_{w_O} v_{w_I}\right) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\left[\log\sigma\left(-v'^{\top}_{w_i} v_{w_I}\right)\right] \tag{8.7}$$

en donde $P$ es la distribución de la que se extraen los casos negativos.

Una tercera extensión a mencionar es la **composición de frases**. Sin necesidad de ahondar en detalles, un problema con la implementación original que se subsana en este trabajo es el uso de frases como tokens. Por ejemplo, es necesario aprender un embedding específico para *Buenos Aires*, ya que se usa en contextos diferentes a *Buenos* y a *Buenos* y tiene un significado propio. Para ello, los autores generan una técnica para encontrar frases que son composición de términos y agrandar $V$.

En este trabajo se introduce una nueva propiedad algebraica de interés, veamos los siguiente ejemplos:

$$vec(Russia) + vec(river) \text{ es cercano a } vec(VolgaRiver) \tag{8.8}$$

$$vec(Germany) + vec(capital) \text{ es cercano a } vec(Berlin) \tag{8.9}$$

Los vectores de palabras del embedding son entrenados para predecir las palabras cercanas en una frase. Además, se encuentran relacionados logarítmicamente con las probabilidades computadas en la salida del modelo. Por este motivo, la suma de dos vectores de palabras está relacionada con el producto de dos distribuciones de contexto. El producto opera como una función AND. Esto es lo que permite que al sumar dos vectores tengamos un nuevo vector que es similar a alguna palabra que tiene sentido en el contexto de esas dos palabras.

### 8.1.3 Global Vectors (GloVe)

En [43] se plantea que existen dos familias de técnicas que permiten generar vectores de palabras:

1. Métodos de factorización de matrices como Latent Semantic Analysis (LSA). En este tipo de modelos lo que se intenta es generar una matriz que contenga información estadística relevante y luego reducir su dimensionalidad. En LSA se emplea una matriz de términos y documentos, donde en cada fila hay un término y en cada columna un documento, y en las celdas la cantidad de ocurrencias. Luego, se reduce la dimensionalidad de la matriz aplicando una descomposición en valores singulares de la misma, esto permite conservar en cierta medida las propiedades estadísticas reduciendo el tamaño de la matriz.

2. Modelos basados en ventana, donde básicamente se trata de construir un problema predictivo en el contexto de una palabra, como los ya presentados [3] y [38]. Estos modelos tienen la ventaja de permitir relaciones algebraicas interesantes pero, por otra parte, no aprovechan los estadísticos globales del corpus sino que aprenden sólo por el contexto.

GloVe pretende acercar ambas familias de técnicas con un enfoque conjunto.

Este trabajo arranca con la siguiente observación: dadas dos palabras, como hielo (ice) y vapor (steam), la palabra sólido debería aparecer más frecuentemente con hielo que con vapor. Con lo cual, la probabilidad de sólido dado que

| Probabilidad y ratio | $k$ = solid | $k$ = gas | $k$ = water | $k$ = fashion |
|---|---|---|---|---|
| $P(k\|\text{ice})$ | $1{,}9 \times 10^{-4}$ | $6{,}6 \times 10^{-5}$ | $3{,}0 \times 10^{-3}$ | $1{,}7 \times 10^{-5}$ |
| $P(k\|\text{steam})$ | $2{,}2 \times 10^{-5}$ | $7{,}8 \times 10^{-4}$ | $2{,}2 \times 10^{-3}$ | $1{,}8 \times 10^{-5}$ |
| $P(k\|\text{ice})/P(k\|\text{steam})$ | 8,9 | $8{,}5 \times 10^{-2}$ | 1,36 | 0,96 |

Cuadro 7: Probabilidad y ratios, tomado de [43]

está la palabra hielo debería ser mayor que la probabilidad de sólido dado que está presente la palabra vapor. El ratio entre ambas probabilidades debería ser mucho mayor a 1. En cambio, si en vez de sólido tuviésemos una palabra no relacionada con hielo ni con vapor el ratio debería ser cercano a 1. Puede observarse lo dicho en la tabla 7.

De esta manera, se plantea que puede ser interesante modelar como objetivo el ratio de las probabilidades, ya que este valor es altamente informativo de la relación entre tres palabras.

Antes de seguir con la derivación del modelo se define lo siguiente. Siendo $X$ la matriz de coocurrencias entre pares de palabras, donde $X_{ij}$ representa la cantidad de veces que la palabra $j$ ocurre en el contexto de $i$. Entonces $X_i = \sum_k X_{ik}$ es la cantidad de veces que cualquier palabra aparece en el contexto de $i$. De esta forma tenemos que la probabilidad de ver la palabra $j$ en el contexto de $i$ es:

$$P_{ij} = P(j|i) = X_{ij}/X_i \tag{8.10}$$

[5]Los autores no reportan el número de tokens, como aproximación se toman la cantidad de palabras en Wikipedia para el idioma español (accedidas el 29/04/2021 en https://es.wikipedia.org/w/index.php?title=Especial:Estad%C3%ADsticas&action=raw)

[6]Los autores no reportan el número de tokens sino el número de páginas, un total de 19.102.397 en español. Además, indican que cortan los documentos a un máximo de 512 tokens. De allí que inferimos que el total de tokens es, como máximo, 9.780.427.264

Ahora bien, dadas tres palabras $i$, $j$ y $k$, la forma más general para modelar el ratio de probabilidad es:

$$F\left(w_i, w_j, \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{8.11}$$

donde $w \in \mathbb{R}^d$ son vectores de palabras y $\tilde{w} \in \mathbb{R}^d$ son los vectores de palabras de contexto, es decir, hay dos tipos de embeddings. En nuestro ejemplo $i$ era hielo, $j$ era vapor y $k$ era sólido.

Como se pretende obtener vectores de palabras que codifiquen el ratio entre ambas probabilidades pero en el espacio vectorial, podemos restringir $F$ a que dependa de la diferencia entre las palabras objetivo:

$$F\left(w_i - w_j, \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{8.12}$$

Siendo que se desean obtener embeddings que tengan las propiedades algebraicas que vimos en Word2Vec $F$ no debería ser una red que oculte la relación lineal entre los vectores. En definitiva, se intenta conservar la relación lineal entre $w_i - w_j$ y $\tilde{w}_k$. Por este motivo se plantea a $F$ como:

$$F\left(\left(w_i - w_j\right)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{8.13}$$

Debe tenerse en cuenta que la distinción entre un término como objetivo y como contexto es arbitraria, con lo cual el modelo debe ser invariante al cambio de rol para una palabra. Para mantener la simetría se comienza pidiendo homomorfismo, con lo cual se puede mostrar que:

$$F\left(\left(w_i - w_j\right)^T \tilde{w}_k\right) = \frac{F\left(w_i^T \tilde{w}_k\right)}{F\left(w_j^T \tilde{w}_k\right)} \tag{8.14}$$

De esta forma:

$$P_{ik} = X_{ik}/X_i = F\left(w_i^T \tilde{w}_k\right) \tag{8.15}$$

La solución a 8.14 es $F = exp$ o, lo que es igual:

$$w_i^T \tilde{w}_k = \log\left(P_{ik}\right) = \log\left(X_{ik}\right) - \log\left(X_i\right) \tag{8.16}$$

Así, podemos convertir el ratio de probabilidad en un diferencia de logaritmos. Por otra parte, el término $log(X_i)$ no depende de $k$ con lo que puede ser absorbido en un término de sesgo $b_i$. Además, para mantener la simetría con $k$ es necesario agregar un término $\tilde{b}_k$.

Así tenemos que:

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log\left(X_{ik}\right) \tag{8.17}$$

Una última cuestión a considerar es que un problema de esta especificación del modelo es que le da la misma importancia a todos los pares de coocurrencias, sin importar si son frecuentes o no. Para ello se agrega una función que pesa la importancia del par. De esta manera, se tiene la siguiente función de costo del modelo:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \tag{8.18}$$

Como $f$ toman:

$$f(x) = \begin{cases} (x/x_{\text{máx}})^{\alpha} & \text{if } x < x_{\text{máx}} \\ 1 & \text{otherwise} \end{cases} \tag{8.19}$$

En este caso los autores argumentan que el modelo no depende demasiado de $x_{\text{máx}}$, que lo fijan en 100 y que encontraron los mejores resultados con $\alpha = 3/4$.

En este trabajo, además, los autores hacen una comparación entre las funciones objetivos de Word2Vec y GloVe. Comienzan planteando que la función objetivo global de un modelo skip-gram se puede pensar como:

$$J = - \sum_{\substack{i \in \text{corpus} \\ j \in \text{context(i)}}} \log Q_{ij} \tag{8.20}$$

Donde $Q_{ij}$ es una función softmax que modela la probabilidad de ver a $j$ en el contexto de $i$. Si en vez de evaluar el modelo corriendo una ventana por el contexto primero agrupamos aquellas palabras que ocurren juntas, haciendo uso de la matriz de coocurrencia $X$:

$$J = -\sum_{i=1}^{V} \sum_{j=1}^{V} X_{ij} \log Q_{ij} \tag{8.21}$$

Luego, recordando que $X_i = \sum_k X_{ik}$ y que $P_{ij} = X_{ij}/X_i$ tenemos que:

$$J = -\sum_{i=1}^{V} X_i \sum_{j=1}^{V} P_{ij} \log Q_{ij} = \sum_{i=1}^{V} X_i H\left(P_i, Q_i\right) \tag{8.22}$$

Donde vemos que $H$ es la función de entropía cruzada entre $P_i$ y $Q_i$, pesada por $X_i$. Los autores argumentan

que esta función para medir distancia entre probabilidades tiene el defecto de darle demasiado peso a los términos poco frecuentes. Además, requiere que $Q$ sea normalizada, lo cual genera un costo computacional insoslayable. Aunque, como vimos antes, existen aproximaciones que mejoran la eficiencia en el cómputo.

Además, los autores argumentan que una alternativa sería tomar la diferencia de cuadrados de los $P$ y $Q$ sin normalizar pesados por la cantidad de ocurrencias, obteniendo:

$$\hat{J} = \sum_{i,j} X_i \left( \hat{P}_{ij} - \hat{Q}_{ij} \right)^2 \tag{8.23}$$

donde $\hat{P}_{ij} = X_{ij}$ y $\hat{Q}_{ij} = \exp\left(w_i^T \tilde{w}_j\right)$. Además, argumentan que $X_{ij}$ puede crecer mucho en la práctica, lo cual puede generar problemas numéricos. Para resolver eso se puede aplicar logaritmos a $\hat{P}_{ij}$ y a $\hat{Q}_{ij}$ teniendo:

$$\hat{J} = \sum_{i,j} X_i \left( \log \hat{P}_{ij} - \log \hat{Q}_{ij} \right)^2 = \sum_{i,j} X_i \left( w_i^T \tilde{w}_j - \log X_{ij} \right)^2 \tag{8.24}$$

Por último, es evidente que $X_i$ como factor de peso no tiene por qué ser óptimo, con lo cual podemos generalizar ésto reemplazando por una función $f(X_{ij})$ lo cual nos deja en la ecuación 8.18.

### 8.2 Guía para anotadores

**Relevancia**: ¿se selecciona contenido importante? Qué es importante o relevante es una cuestión subjetiva, sin embargo, en términos generales un resumen no debería incluir detalles o comentarios marginales o "al pasar"de la noticia original. Puntajes:

- 2: el resumen refleja lo más importante de la nota y no se incluye información irrelevante
- 1: O bien el resumen refleja algo importante pero tal vez no lo más importante, o bien incluye lo más importante de la nota pero también incluye información irrelevante.
- 0: El resumen no refleja lo más importante de la nota, sólo incluye información irrelevante

**Veracidad**: ¿los hechos que marca el resumen están alineados con la fuente? En este caso nos interesa saber si los hechos mencionados en la nota se reflejan fielmente en el resumen. En ocasiones un resumen puede contener información que parece real pero no lo es. Por ejemplo, fecha, lugares, personas, etc. Por ejemplo, a veces se menciona un sustantivo que sí aparece en la nota pero asociado a una acción que corresponde a otro sustantivo.

- 2: Todos los hechos mencionados son ciertos.
- 1: Algunos hechos mencionados son ciertos pero otros no.
- 0: Ningún hecho mencionado es cierto.

**Coherencia y cohesión**: las frases, ¿tienen coherencia y cohesión?

- 2: Todas las frases están bien formadas (son coherentes y muestran cohesión).
- 1: Algunas frases están bien formadas (son coherentes y muestran cohesión) pero otras no.
- 0: Ninguna frase está bien formada.

Nota: los errores de tipeo no deben ser considerados salvo que sean tan graves como para realmente afectar el significado.

### 8.3 Modelo de lenguaje - Beto2Beto

A continuación se presentan resultados de generación de texto empleando Beto2Beto. El único postprocesamiento realizado es la eliminación de los tokens como CLS, SEP Y UNK. Debe tenerse presente que Beto2Beto se basa en una arquitectura encoder-decoder y que no es la arquitectura más adecuada para la generación de texto libre.

| Entrada | Salida |
|---|---|
| El ministro de economia de la argentina dijo que | el gobierno de mauricio macri no tiene nada que ver con la crisis economica y social, pero si es cierto lo contrario. en este momento hay una situacion muy dificil para todos los argentinos : tenemos un deficit fiscal primario del 27 por ciento anual ( 18 puntos porcentuales ) dijo a periodistas fernandez al ser consultado sobre las medidas anunciadas ayer tras conocerse su decision final respecto hacia argentina luego se conocieran datos oficiales publicados hoy donde estimaban haber caido hasta 40 mil millones durante 2016 - 2019 debido principalmente parcialmente tambien factores externos como inflacionaria e incertidumbre politica |

| Entrada | Salida |
|---|---|
| El fiscal general de la nación dijo estar en contra de ese fallo. | no es un fallo de la corte suprema, sino que se trata del tribunal constitucional. el presidente tiene una responsabilidad muy grande en este tema y por eso le pedimos a los magistrados ( jueces ) tener claridad sobre lo actuado hasta ahora para poder actuar con total transparencia e imparcialidad dijo al ser consultado si habian pedido su destitucion o bien porque estaban dispuestos hacerlo sin ningun tipo contravencional ni penal alguno |

| Entrada | Salida |
|---|---|
| River derrotó a su clásico rival el domingo pasado | y se quedo con el titulo de la superliga. en ese contexto, los hinchas del millonario no dudaron un segundo para festejar su triunfo : es una alegria inmensa poder estar aca hoy ( por ayer ) porque nos da mucha fuerza seguir trabajando dia a partido estamos muy contentos que esten aqui todos ustedes! gracias al club xeneize siempre lo digo asi cuando me toca jugar contra boca juniors o river plate pero nunca mas voy hacer esto como entrenador? |

| Entrada | Salida |
|---|---|
| La economía mexicana volvió a caer este trimestre, debido principalmente a | la desaceleracion de las exportaciones, que se contrajo un 0. 1 en el segundo trimestre del ano y una caida interanual cercana al 3 por ciento frente a igual periodo previo ( - 2 ) debido principalmente sobre todo para los sectores manufacturero e industrial con menor dinamismo economico como son automotriz o telefonia celular entre otros rubros mas importantes durante este primer semestre fiscal 2019 |

| Entrada | Salida |
|---|---|
| Las criptomonedas son monedas que se usan cada vez mas | en todo el mundo. son un medio de pago que se utiliza para comprar billetes, monedas y otros productos financieros a traves del internet o la banca tradicional como parte importante tambien es una forma segura e independientemente si estas usando criptomonedas tradicionales ( bitcoin ) u otras divisas virtuales con las cuales cuentas actualmente sin necesidad tener acceso al sistema financiero mundial por lo general no existen bancos centrales ni reguladores internacionales autorizados especificando su uso especifico dentro los mercados emergentes donde operan estos instrumentos bancarios |