

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“ANÁLISIS Y CLASIFICACIÓN DE TEXTOS ILÍCITOS COMO APOYO PARA LA GESTIÓN DE ESPIONAJE POLICIAL”

FRANCO ENRIQUE PARRA CAMPOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Carlos Valle
Profesor Correferente: Claudio Torres

Julio - 2023

DEDICATORIA

Para mi madre, Fresia Campos, y mi hermano, Christian Parra.

AGRADECIMIENTOS

En la vida, pienso que son pocos los logros que se puedan compartir en comunidad, por eso, quiero que estas palabras queden presentes en los corazones de todas aquellas personas que fueron partícipes, directos e indirectos, de mi larga trayectoria por esta fantástica casa de estudios. Para ustedes, Pedro Campos, Fresia Quijana, Fresia Campos y Christian Parra, con quienes he tenido la fortuna de convivir y adquirir todos los valores que me han hecho ser noble ante la vida; sacrificaron su tiempo por mi tiempo y su sudor por mis sueños. A ti, Gustavo Cabrera, que sin pedir nada a cambio, asumiste un rol y supiste ganarte mi corazón y el de mi familia. A ti, Scarlett Cortés, por el cariño, la paciencia y las palabras de aliento. A mis amigos, André Morgado y Jhoss Figueroa, por siempre estar dispuestos a escuchar. A mis compañeros de universidad, Bryan Masías y Patricio Calderón, con quienes me relacioné y apoyé durante toda mi carrera. A todos los profesores de informática quienes supieron sacar lo mejor de mí. A mi profesor guía, Carlos Valle, por creer en este proyecto y aconsejarme oportunamente. A todos, mis más sinceros agradecimientos.

RESUMEN

Resumen— Día a día, nuevas generaciones encuentran en Internet un lugar más por donde coexistir y donde crear una imagen de sí mismos distinta a la que reciben de sus padres o profesores. Muchas veces, ni ellos, ni sus tutores, cuentan con la madurez y/o las herramientas suficientes para concebir los riesgos inherentes en Internet. Uno de los más frecuentes suele ser la captación de menores (*grooming*) que, tan solo en el año 2020, tuvo un incremento del 97,5 % debido a la COVID-19. Parte de las soluciones propuestas por los investigadores, se enfocaron en distinguir líneas escritas por pedófilos o pseudovíctimas, y en crear perfiles con base en características psicolingüísticas, empleando clasificadores tradicionales como KNN o SVM, combinados con n-gramas, LIWC o TF-IDF. El problema, sin embargo, es que al no ser alternativas a nivel de conversaciones, resultan poco prácticas. La competencia “PAN 2012” buscó precisamente este objetivo, consiguiendo en el mejor de los casos (BOW con TF-IDF y SVM con NN) un Valor F del 87,34 %. Con la llegada de los modelos atencionales, muchas tareas NLP se vieron beneficiadas. Producto de lo anterior, en este trabajo se propone un modelo de inteligencia artificial BERT capaz de discriminar conversaciones con fines perversos (manipulación, explotación o abuso sexual) hacia menores. Los resultados muestran una mejora del 7,00 % en el Valor F con respecto al mejor candidato en “PAN 2012”. La aplicación de la solución en diversos diálogos protagonizados por acechadores sexuales, revelan la eficacia, robustez y certinidad de la arquitectura, consolidándola como un primer acercamiento hacia un instrumento preventivo que apoye y facilite el trabajo de los agentes policiales durante sus investigaciones.

Palabras Clave— Captación de Menores, Atención, BERT, NLP

ABSTRACT

Abstract— Day by day, the new generations find on the Internet an extra place where they can coexist and create an image of themselves different from the one they receive from their parents or teachers. Very often, neither they nor their tutors have the maturity and/or tools to comprehend the existing risks on the Internet. One of the most frequent tends to be grooming, which just in the 2020 year grew by 97.5 % because of COVID-19. Part of the proposals by the researchers focused on distinguishing lines written by pedophiles or pseudovictims and creating profiles based on psycholinguistic features by using traditional classifiers like KNN or SVM combined with n-grams, LIWC or TF-IDF. However, the problem is that they are not practical because of their non-conversational focus. The “PAN 2012” competition tried to achieve this objective, getting in the best scenario (BOW with TF-IDF and SVM with NN) an F-Score of 87.34 %. With the arrival of attention models, many NLP tasks benefited. Because of this, in this work we propose a BERT artificial intelligence model able to distinguish conversations with perverted intentions (manipulation, exploitation, or sexual abuse) among minors. The results show an improvement of 7.00 % in the F-Score with respect to the best candidate in “PAN 2012”. The application of the solution in different dialogues carried out by sexual predators reveals the effectiveness, sturdiness, and certainty of the architecture, making it a first approach to a preventive instrument that supports and eases police officers’ work during their research.

Keywords— Grooming, Attention, BERT, NLP

GLOSARIO

ANN: *Artificial Neural Network* (red neuronal artificial).

BERT: *Bidirectional Encoder Representations from Transformers* (representaciones de codificador bidireccional de transformadores).

BLEU: *Bilingual Evaluation Understudy* (Estudio de evaluación bilingüe).

BOW: *Bag of Words* (bolsa de palabras).

CBOW: *Continuous bag-of-words model* (Modelo de “bolsa de palabras” continuo).

CL: *Controlled Language* (lenguaje controlado).

CNN: *Convolutional Neural Network* (redes neuronales convolucionales).

DT: *Decision trees* (árboles de decisión).

ELMo: *Embeddings from Language Model* (embedidos para modelamiento del lenguaje). Es un método para representar una secuencia de palabras en una secuencia de vectores.

EOS: *End of Sequence* (término de secuencia).

FNN: *Feedforward Neural Network* (red neuronal pre-alimentada).

GD: *Gradient Descent* (gradiente descendente).

GPU: *Graphics Processing Unit* (unidad de procesamiento gráfica).

GRU: *Gated Recurrent Unit* (unidad recurrente cerrada).

KNN: *K-Nearest Neighbors* (k-vecinos más cercanos).

LIWC: *Linguistic Inquiry and Word Count* (investigación lingüística y recuento de palabras). Es un programa de análisis de textos.

LR: *Logistic Regression* (regresión logística).

LSTM: *Long Short-Term Memory* (memoria a “corto plazo” larga).

MLM: *Masked Language Modeling* (modelado de lenguaje enmascarado). Corresponde a una de las dos tareas no supervisadas utilizadas para entrenar BERT.

NLP: *Natural language processing* (procesamiento del lenguaje natural).

NN: *Neural Network* (red neuronal).

NSP: *Next Sentence Prediction* (predicción de la siguiente oración). Corresponde a una de las dos tareas no supervisadas utilizadas para entrenar BERT.

OOM: *Out Of Memory* (sin memoria). Es un error generado cuando no existe más memoria disponible a asignar por parte del sistema operativo a un programa.

P2P: *Peer to Peer* (de igual a igual).

PJ: *Perverved Justice*. Es el nombre de la fundación y sitio web encargado capturar depredadores sexuales en Estados Unidos.

QA: *Question Answering* (pregunta y respuesta).

RAM: *Random Access Memory* (memoria de acceso aleatoria).

RNN: *Recurrent Neural Network* (red neuronal recurrente).

SVM: *Support Vector Machine* (máquinas de soporte vectorial).

Seq2Seq: *Sequence to Sequence* (secuencia a secuencia).

TF-IDF: *Term Frequency-Inverse Document Frequency* (frecuencia de término – frecuencia inversa de documento).

TPU: *Tensor Processing Unit* (unidad de procesamiento tensorial).

XML: *Extensible Markup Language* (lenguaje de marcado extensible).

NOTACIÓN

Los vectores serán denotados en negritas, por ejemplo, $\mathbf{x}$. Los conjuntos serán denotados con letras caligráficas y en mayúsculas, por ejemplo, $\mathcal{A}$. El largo de los elementos que se describirán a continuación se denotarán por $\mathfrak{L}(\cdot)$:

1. Vectores unidimensionales. Por ejemplo, si $\mathbf{x} \in \mathbb{R}^n$, entonces, $\mathfrak{L}(\mathbf{x}) = n$.
2. Secuencias de escalares. Por ejemplo, si $x = (x_i)_{i=1}^n = (x_1, \dots, x_n)$ tal que $x_i \in \mathbb{R}$, entonces, $\mathfrak{L}(x) = n$.
3. Secuencias de vectores. Por ejemplo, si $\mathbf{x} = (\mathbf{x}_i)_{i=1}^n = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ tal que $\mathbf{x}_i \in \mathbb{R}^m$, entonces, $\mathfrak{L}(\mathbf{x}) = n$ y $\mathfrak{L}(\mathbf{x}_i) = m$. La ambigüedad entre un vector y una secuencia de vectores a causa de la notación, quedará resuelta en la definición misma de la variable.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	V
GLOSARIO	VI
NOTACIÓN	VII
ÍNDICE DE FIGURAS	X
ÍNDICE DE TABLAS	X
ÍNDICE DE MÓDULOS	X
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	4
1.1 Niños y su interacción con Internet	5
1.2 Tipos de relaciones e intereses	5
1.3 Grooming en Internet	6
1.4 Detección y prevención de grooming en Internet	7
1.5 Objetivos	10
1.5.1 Objetivo general	10
1.5.2 Objetivos específicos	10
CAPÍTULO 2: MARCO CONCEPTUAL	11
2.1 Unidades de procesamiento	11
2.2 Funciones de activación	12
2.3 Funciones de pérdida	13
2.4 Optimización de los modelos	14
2.5 Propagación hacia adelante	14
2.6 Propagación hacia atrás	14
2.7 Gradiente descendente	16
2.8 Procesamiento del lenguaje natural	16
2.9 Vectorización de textos	18
2.9.1 Incrustación de palabras	19
2.9.2 Word2vec	19
2.10 Procesamiento de textos	20
2.10.1 Tokenización basada en espacios	21
2.10.2 Tokenización basada en caracteres	22
2.10.3 Tokenización basada en subpalabras	23

2.10.4 WordPiece	23
2.10.4.1 Generación del vocabulario	23
2.10.4.2 Tokenización de las entradas	25
2.11 Redes neuronales recurrentes	27
2.12 LSTM	29
2.13 Atención	31
2.13.1 La propuesta de Bahdanau et al.	31
2.13.2 La propuesta de Luong et al.	33
2.13.3 La propuesta de Vaswani et al.	35
2.13.3.1 Codificación posicional	35
2.13.3.2 Atención producto-punto escalado	36
2.13.3.3 Atención producto-punto escalada con enmascaramiento	36
2.13.3.4 Cabezal	37
2.13.3.5 Atención multicabezal	37
2.13.3.6 Conexiones residuales	38
2.13.3.7 Codificador	38
2.13.3.8 Decodificador	38
2.14 BERT	38
2.14.1 Arquitectura	39
2.14.2 Preentrenamiento	40
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	42
3.1 Primera fase	42
3.2 Segunda fase	43
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	44
4.1 Primera fase	44
4.2 Segunda fase	49
4.3 Aplicación de principio a fin	53
CAPÍTULO 5: CONCLUSIONES	56
ANEXOS	59
5.1 Keras Tuner	59
5.2 AdamW	59
5.3 Programación de las tasas de aprendizaje	60
5.4 Métricas	60
REFERENCIAS BIBLIOGRÁFICAS	62

ÍNDICE DE FIGURAS

1 Modelo de “bolsa de palabras” continuo 20

2 Modelo “salta gramas” (*skip-gram model*) 21

3 RNN con y sin desplegado. 29

4 Arquitectura LSTM vista como nodos. 30

5 Flujo de procesamiento de conversaciones desde la primera fase hacia la
segunda fase 43

ÍNDICE DE TABLAS

1 Tiempos de ejecución y métricas para cada repetición en la primera fase. 45

2 Rango de valores de los hiperparámetros para la segunda fase 50

3 Valores de los hiperparámetros para la segunda fase 51

4 Métricas para cada repetición utilizando distintos números de *epochs* en la
segunda fase 52

5 Resultados de la aplicación del modelo sobre los diez perfiles más “babosos”
según el sitio Perverted Justice 55

ÍNDICE DE MÓDULOS

1 El resultado de aplicar la tokenización basada en espacios para la oración
“Don’t you love transformers? We sure do.” como una lista de *strings*. 21

2 El resultado de aplicar la tokenización basada en espacios para la oración
“Don’t you love transformers? We sure do.” considerando signos de
puntuación como una lista de *strings*. 22

3 El resultado de aplicar la tokenización basada en caracteres para la oración
“Don’t you love transformers? We sure do.” como una lista de *strings*. 22

4 Distribución de las frecuencias absolutas de cada palabra presente en un
hipotético conjunto de datos 23

5	Distribución de las frecuencias absolutas de cada carácter de una palabra presente en un hipotético conjunto de datos	24
6	El vocabulario inicial como una lista de <i>strings</i>	24
7	Todas las combinaciones posibles de unión entre elementos de la misma palabra partiendo de izquierda a derecha	25
8	Distribución de las frecuencias absolutas de cada elemento en el <i>split</i> luego de combinar los caracteres “h” y “u” en un hipotético conjunto de datos	25
9	El resultado de aplicar una iteración del algoritmo WordPiece a un hipotético conjunto de datos	26
10	Un hipotético vocabulario obtenido luego de aplicar el algoritmo de la sección (2.10.4.1) expresado como una lista de <i>strings</i>	26
11	Tokenización basada en espacios considerando signos de puntuación para la oración “Hong Tong!” en un formato de lista de <i>strings</i>	27
12	Tokenización basada en WordPiece para la palabra “Hong” en un formato de lista de <i>strings</i>	27
13	Tokenización basada en WordPiece para la oración “Hong Tong!” en un formato de lista de <i>strings</i>	27

INTRODUCCIÓN

Desde la creación de Internet por medio del Departamento de Defensa de Estados Unidos en 1969 [Abbate, 1994], el mundo se ha visto sometido a un periodo de globalización jamás antes visto: con casi dos tercios de la población mundial conectada a través de distintos aparatos tecnológicos [Fernández, 2022], resulta imposible negar la transversalidad de su impacto en las distintas actividades humanas, en específico, aquellas donde se establecen relaciones interpersonales. A raíz de lo anterior, la preocupación por regular sus formatos y consumos se ha hecho evidente a través de distintos estudios psicológicos [Arab y Díaz, 2015, V.K et al., 2021, Berger et al., 2019], los cuales han demostrado que las jerarquías familiares se han visto alteradas, pues muchas veces son los “nativos digitales” quienes enseñan y educan a sus padres [Arab y Díaz, 2015] dejándolos expuestos a mayores riesgos tales como *cyberbullying*, contacto con extraños, *sexting* y pornografía [Drotner y Livingstone, 2008]. Nuevos campos de investigación surgen y revistas especializadas como la Journal of Computer-Mediated Communication, Cyberpsychology, Behavior, and Social Networking o New Media & Society, informan periódicamente distintas métricas que reflejan evidencias sobre los distintos daños ocasionados a menores de 18 años, entre ellas, agresiones en línea y *cyberbullying*, contenido dañino, mal uso de datos personales, uso excesivo de internet, *sexting*, visualización de imágenes sexuales y encuentros con nuevas personas [Smahel et al., 2020]. Según resultados entregados por EU Kids Online en el año 2020, en promedio, 20 % de los niños encuestados entre 9 a 16 años reportan haber sido víctimas de agresiones en línea, mientras que de un 10 % a 20 % confiesan haber sido perpetradores [Smahel et al., 2020]. Otras menciones indican que entre un 11 % y 31 % de los niños entre 12 a 16 años reportan haber visto mensajes de odio, como mínimo, una vez al mes.

Si bien la exposición en los medios digitales propicia múltiples contextos para expresar y explorar aspectos de la identidad [Valkenburg y Peter, 2011], en adolescentes (cuya construcción de su persona se modula a través de la interacción y retroalimentación de sus pares) puede ser perjudicial, principalmente, porque se pueden manifestar desde el anonimato, con un distanciamiento afectivo importante, un nivel bajo de empatía y un “pensamiento hablado” [Arab y Díaz, 2015]. Otras consecuencias involucran la dificultad que tienen para diferenciar el contenido público del privado, por lo que son más propensos a verse involucrados en situaciones que pueden generar altos niveles de estrés y daños importantes en la visión de sí mismos con serios riesgos para su salud mental, sobre todo, si consideramos la rápida difusión de los contenidos negativos y descalificativos.

Actualmente, las redes sociales en línea son uno de los medios por donde más gente interactúa y se estima que cerca de un 59 % de los habitantes de la tierra las utiliza [DataReportal, 2022]. Un usuario promedio invierte diariamente cerca de 2 horas con 27 minutos, lo que se traduce a más de 3 días completos por mes dedicados a su consumo (equivalente al 15 % del tiempo que pasamos despiertos en una vida) [Dixon, 2022a]. Hasta julio de 2022, un usuario de TikTok pasaba 23,6 horas mensuales; en YouTube, 23,2 horas

mensuales; en Facebook, 19,4 horas mensuales; en WhatsApp, 17,5 horas mensuales; en Instagram, 11,8 horas mensuales; en Twitter, 5,4 horas mensuales. Un 59,4 % de las mujeres que usan redes sociales entre 16 a 24 años, identifican como favorita a alguna de las grandes aplicaciones líderes en redes sociales (WhatsApp, Instagram, Facebook, TikTok y Twitter), mientras que en los hombres, la proporción se estima en un 59,3 % [Kemp, 2022]. La mención de estas estadísticas no es arbitraria, primero, porque durante los últimos años, hemos visto que distintos episodios de violencia se han originado desde variadas redes sociales tales como Facebook al lograr fomentar pensamientos antiinmigración, e incluso, favoritistas hacia ciertos candidatos presidenciales [Naher y Minar, 2018, Orrù, 2014, Vendemia *et al.*, 2019]; segundo, porque existe una considerable proporción de jóvenes en vías de desarrollo que las utiliza a diario [United Nations, 2021]. Tan solo en Facebook, conforman el 28,2 %, mientras que en Instagram y TikTok, el 39,1 % y 25 % [Dixon, 2022b, Dixon, 2022c, Ceci, 2022].

Distintas redes sociales han tomado cartas en el asunto y han propuesto medidas para regular el contenido desplegado a los usuarios con el fin de mejorar la convivencia. En Instagram, cuyos principales elementos de interacción son el *Feed* y las *Stories*, se han reducido considerablemente las probabilidades de mostrar contenido que incluya personas desnudas, temas sexuales afines, desinformación, *bullying*, discursos de odio e incitación a la violencia [Instagram, 2022a]; si bien no se menciona el modelo, sí se indican algunas pautas que se siguen a la hora de filtrar basándose en las Normas Comunitarias [Instagram, 2022b]. YouTube, por su parte, propuso YouTube Kids en 2015 como una iniciativa para crear una aplicación segura con los niños [Ben-Yair, 2015]: decide si un video es apropiado o no basándose en los cuadros (*frames*) y metadata del video tales como el título, descripción, etiquetas, número de visitas, calificaciones y apreciaciones por parte de la comunidad en los comentarios. De igual forma, Facebook ha estado continuamente desarrollando modelos de inteligencia artificial tales como RoBERTa, XLM-R y Linformer para detectar contenido dañino usando como materia prima el procesamiento de lenguajes naturales (NLP) [Facebook, 2020, Liu *et al.*, 2019, Conneau *et al.*, 2019, Wang *et al.*, 2020]. Twitter, por su parte, se encuentra en la construcción de una plataforma libre de abusos y *spam* usando una métrica desarrollada por Cortico [Cortico, 2018], la cual busca medir el “estado de salud” de una conversación usando cuatro indicadores: atención compartida (*shared attention*), realidad compartida (*shared reality*), variedad de opinión (*variety of opinion*) y recepción (*receptivity*). Desde su aplicación, 38 % del contenido abusivo ha sido manejado por el equipo de Twitter sin depender de reportes externos de usuarios de la plataforma, se han triplicado los números de cuentas abusivas suspendidas dentro de 24 horas y los contenidos privados y sensibles han sido eliminados 2,5 veces más rápido [Hicks y Gasca, 2019].

En la mayoría de soluciones dadas, el enfoque está puesto en el contenido que consumen los usuarios y no así en las interacciones que se generan entre ellos a través de la mensajería instantánea, por ello, en esta memoria se buscará implementar un sistema de reconocimiento de conversaciones ilícitas usando un modelo *transformer* llamado BERT [Devlin *et al.*, 2018]. Dado que lo “ilícito” tiene muchas manifestaciones, nos abstendremos a trabajar aquellas que se relacionen con la pederastía y pedofilia.

Para cumplir el fin de este trabajo, se investigarán y describirán los fundamentos matemáticos que definen a los modelos atencionales y por qué resultan ser efectivos en el procesamiento de lenguajes naturales a diferencia de sus antecesores arquitecturas, además, se realizará un preprocesamiento de las fuentes de datos con el fin de disminuir el ruido en las detecciones. Tanto la implementación, entrenamiento y visualización del modelo BERT y sus salidas, se efectuarán en entornos de Kaggle dado que ofrecen máquinas con mayor capacidad de cómputo tanto a nivel CPU, GPU como TPU. Para medir la efectividad, se generarán variados estadísticos basados en conversaciones de pedófilos y pederastas usadas anteriormente como evidencia en investigaciones llevadas a cabo por la Justicia de Estados Unidos.

Finalmente, el documento quedará estructurado de la siguiente forma: en el capítulo 1, se entregará una definición del problema; en el capítulo 2, se ahondará en el marco conceptual; en el capítulo 3, se dará a conocer la propuesta de solución; en el capítulo 4, se validará la solución; en el capítulo 5, se detallarán las conclusiones y trabajos futuros.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

Desde el momento en que una persona decide registrarse en una plataforma en línea, se genera una ruptura de la identidad que da lugar a una **identidad real** y a una **identidad en línea**. La primera se puede entender como un conjunto articulado de rasgos específicos de un individuo o de un grupo que, pueden o no, cambiar durante el tiempo [Dronda, 2005]. La identidad en línea, por su lado, es aquella que se da únicamente en Internet y dado que se interactúa de manera diferente por cada sitio, se genera una imagen puntual de qué somos y qué hacemos, por ello, es que se les suele catalogar como **identidades parciales**, ya que no son una representación completa de lo que somos [Internet Society, 2011], sino más bien un subconjunto, básicamente porque no todos los sitios esperan recibir la misma información. A medida que estos espacios comienzan a recolectar más datos sobre nosotros, se generan procesos que dan orígenes a **perfiles** capaces de saber qué tipo de música consumimos o qué productos harán que, probablemente, desembolsemos dinero. Si alguno de los datos que proporcionamos a dichas plataformas son reales, entonces, estamos creando a una **persona**.

Inevitablemente, cuando nos referimos a “personas” surgen cuestiones relacionadas con la veracidad y honestidad de quienes están detrás de la pantalla, en principio, porque Internet es un medio libre en expresión y acción, y bajo esa premisa, las obligaciones en el ser y hacer quedan como criterio meramente personal, por ende, no es ajeno intuir que muchos internautas sientan que esta red es el medio predilecto para el engaño [Hancock y Woodworth, 2013]. En general, las manifestaciones de ello se pueden clasificar por burla (*trolling*), engaño en categoría, interpretación y ocultamiento de identidad [Donath, 1996]. En comunidades virtuales, vemos que el problema de la “interpretación” sucede con regularidad. Por ejemplo, una muestra encontró que un 16 % de quienes tuvieron citas en línea, resultaron ser engañados por un estafador (*scammer*) [Buchanan y Whitty, 2013]. También, es posible ver *engaños por categoría* cuando se asumen géneros o edades diferentes, en particular, entre un 18 % a 62 % de participantes de distintas “salas de conversación” (*chat rooms*) reportan haber mentido sobre su edad, género, ocupación, educación e ingresos (dicha variación se debe principalmente al tipo de mentira y género de los encuestados) [Whitty, 2002]. Otras fuentes indican que un 45 % ha mentido sobre su edad y un 27 % sobre su género [Caspi y Gorsky, 2006]. La percepción sobre el engaño, después de todo, es compartida, en especial si consideramos que existe un 54 % de probabilidades de que la persona con quien estemos interactuando mienta, omita o atenúe, algún aspecto de su vida [Whitty, 2008], y esto hace detenernos para reflexionar sobre las relaciones que generan los niños¹ en las redes sociales.

¹Usaremos la definición de las Naciones Unidas para referirnos indistintamente a los niños como todas aquellas personas menores de 18 años [United Nations, 2021].

1.1. Niños y su interacción con Internet

Desde estar en contacto con familiares hasta expresar sus más íntimos pensamientos [DataReportal, 2022], los niños se han acostumbrado a convivir en un entorno digital que ofrece una característica invaluable: la visibilidad. Cuando estos niños alcanzan los periodos de la adolescencia y madurez, experimentan la necesidad de compartir entre pares porque les permite, por un lado, establecer sus propias relaciones sociales, y por otro, el poder formar una imagen de sí mismos distinta a la que reciben de sus padres o profesores [Martín *et al.*, 1998]. El punto, después de todo, es conseguir aceptación. Naturalmente, los niños comenzarán a generar lazos con quienes compartan actividades, intereses o un mismo estatus socioeconómico [Simkin, 2013]. Como el formato de presentación cambia, la manera de llegar se basa a través de la exposición [Hawk *et al.*, 2019]. Son de amplio conocimiento los problemas psicológicos que conlleva esta práctica cuando no es regulada [Shabahang *et al.*, 2022], sin embargo, es importante añadir a la discusión el tipo de amistades, tratos e intereses que se forjan. Lo anterior se vuelve especialmente sensible si la persona con quien el niño interactúa es adulta.

1.2. Tipos de relaciones e intereses

Comúnmente, las relaciones interpersonales que se dan entre seres humanos son de tipo **íntimas** o **afectivas**, en donde hay una conexión profunda y de intimidad con la otra parte y en las que el afecto está presente aunque sea a diferentes niveles; **familiares**, cuya manifestación se da entre personas de nuestra familia y de quienes nos acompañan desde que nacemos; **superficiales**, las cuales son breves, pasajeras y cortas en el tiempo y en las que el afecto y la intimidad no están presentes y tampoco hay ninguna implicancia emocional; **circunstanciales**, que son un híbrido entre las relaciones superficiales e íntimas; **de rivalidad**, donde parten de la enemistad, de la competencia o incluso del odio y del rencor [Castro, 2022]. Una pregunta que surge de forma natural e inmediata es por qué un adulto se vería motivado a generar una relación íntima, familiar, circunstancial o de rivalidad con un menor. La respuesta a cada una de esas interrogantes genera distintas sensibilidades, pero aquellas que en su definición incluyen el intimar resultan, por lo menos, preocupantes. Lo anterior se debe a que las relaciones afectivas persiguen tres cosas: la **necesidad social** o **de pertenencia**, la **necesidad afectiva** y la **necesidad de contacto sexual** [Centro de Psicología López de Fez, 2022]. El último de todos es el más crítico porque entra dentro de los marcos legales de cada país, y en función de la edad de consentimiento sexual², se establecerá una condena siempre y cuando el niño afectado esté por debajo de dicho margen. En Chile, la edad de consentimiento sexual es a los 14 años y sus castigos incluyen presidio mayor en cualquiera de sus grados [Código Penal de Chile, 2022]. Lo que resulta notable de este perfilamiento es que sus características podrían coincidir con las de un abusador sexual y las de un pedófilo, ya

²La edad del consentimiento sexual es aquella en la que legalmente se considera que una persona puede consentir en tener relaciones sexuales.

que ambos persiguen el deleite sexual y el sometimiento aplicado a niños, niñas, adolescentes y jóvenes que carecen de desarrollo emocional y conductual, conocimientos, autoridad y poder [CESOLAA, 2017]. La mayoría de abusos sexuales reportados en los Estados Unidos [Snyder, 2000], han demostrado que cerca de un 67 % de las víctimas han sido menores de 18 años y que un 60 % de ellas no han superado los 12 años. En la mayoría de casos, los victimarios han resultado ser personas de alta confianza para la familia y/o para el niño [CEDEB, 2021].

1.3. Grooming en Internet

Centrémonos ahora en entender cómo estas personas logran acercarse a los menores. Comentamos en párrafos anteriores que Internet posee dos características invaluable: libertad y visibilidad. Dado que estas personas pueden decidir quiénes quieren ser, aprovechan la exposición desmesurada de los menores para poder imitar actitudes típicas, incluyendo el lenguaje, pensamiento y gustos personales. Al ser semejantes, están en condiciones para establecer una relación emocional y de confianza que, más tarde, se transforma en una manipulación, explotación o abuso sexual [NSPCC, 2022]. Técnicamente, a esto se le conoce como *grooming*, y sus consecuencias repercuten en la adolescencia y adultez con ansiedad o depresión, estrés postraumático, autolesiones, sentimientos de culpa, trastornos de la alimentación, alcoholismo, drogadicción y hasta pensamientos suicidas [National Center for Missing & Exploited Children, 2016]. Los medios que comúnmente utilizan son las redes sociales, correo electrónico, aplicaciones de mensajería instantánea y foros o “salas de conversación” de videojuegos y aplicaciones varias [NSPCC, 2022]. Tan solo en el año 2020, hubo un incremento del 97,5 % en casos donde adultos se comunicaron con menores en línea bajo la intención de secuestrarlos o abusar sexualmente de ellos³ [O'Donnell, 2021]. Se piensa que las causas radican en cuatro factores: la rápida introducción de internet desde edades tempranas, el mayor acceso a dispositivos móviles, la COVID-19 al forzar que millones de niños pasen más tiempo conectados y el uso de las plataformas digitales como un lugar donde estos exploran su sexualidad [WeProtect, 2021]. De lo que se conoce, los únicos países que han regulado la actividad en Internet han sido Alemania y Australia al obligar que las redes sociales remuevan rápidamente el contenido ilegal proveyendo reportes de manera transparente. Otros países como el Reino Unido e Irlanda se encuentran trabajando en nuevas medidas para hacer la navegación en línea segura y confiable. La Unión Europea actualmente está evaluando un Acta de Servicios Digitales que busca mejorar los mecanismos para remover contenido en línea ilegal (entre sus puntos se halla el combate contra el abuso sexual infantil) [WeProtect, 2021]. En Chile, no existen regulaciones a nivel de estado para garantizar la protección de menores en Internet y solo se cuentan con sugerencias para los tutores sobre cómo pueden navegar de manera segura [CSIRT, 2021b] [CSIRT, 2021a]. Algunas plataformas más comunes han adoptado muchas de las iniciativas indicadas en Safety By Design de la Australian eSafety Commissioner para reducir el riesgo

³También conocido como *online enticement*, corresponde a una categoría más amplia que incluye al *grooming*

de *grooming* [eSafety Commissioner, 2021], entre ellas destacamos:

- Roblox: Incorporó un sistema de seguridad que bloquea contenido explícito y previene que usuarios jóvenes compartan información personal de contacto [Bowles y Keller, 2019].
- TikTok: Definió políticas para menores de 18 años sobre privacidad y seguridad [TikTok, 2021].
- Instagram: Desarrolló una extensión que detecta y advierte a jóvenes mensajes directos indeseados por adultos [Instagram, 2021].
- YouTube: Añadió “experiencias supervisadas” para niños menores de 13 años que limitan sus capacidades de subir contenido, chatear o recibir comentarios, además de ayudar a padres a manejar el contenido que consumen [YouTube, 2021].

1.4. Detección y prevención de grooming en Internet

El problema de detectar pedófilos y abusadores sexuales en redes sociales se ha abordado de manera deficiente en la literatura debido a la falta de datos y a aspectos legales que involucran el uso de contenido que está a disposición de las policías locales [Bogdanova *et al.*, 2012]. Muchos investigadores han utilizado los trabajos llevados a cabo por la fundación *Perverted Justice* (PJ) como materia prima para entrenar a sus modelos. Allí, adultos de manera voluntaria entran a “salas de conversación” haciéndose pasar como jóvenes de entre 12 a 15 años esperando que personas aparentemente mayores les soliciten sexualmente, de esta forma, inician un trabajo en conjunto con la policía para perseguirlos y capturarlos [Perverted Justice, 2003].

Gran parte de las soluciones propuestas usan modelos de procesamiento de lenguaje natural (NLP), ya que uno de los focos de interacción se da justamente en el diálogo. Clasificadores KNN basados en n-gramas se han aplicado sobre dichas conversaciones y han distinguido en un 94,3 % líneas escritas por pedófilos y líneas escritas por pseudovíctimas [Pendar, 2007]. Otras aplicaciones de KNN lograron un 83 % de exactitud cuando las líneas se ordenaron bajo las siguientes categorías: intercambio de información personal, *grooming*, intenciones de acercamiento y ninguna de las anteriores [McGhee *et al.*, 2011]. Encontramos, por otro lado, trabajos que se han enfocado en detectar automáticamente la edad y género de los participantes usando SVM sobre *chats* de la red social belga Netlog [Peersman *et al.*, 2011]. La discriminación entre aquellos que eran mayores y menores que 16 años fue de un 71,3 %: dicha exactitud es aún mayor si se profundiza la edad de corte entre personas menores que 16 años y mayores a 25 años logrando un 88,2 %, además, sus experimentos mostraron que con tan solo el 50 % de los datos, la exactitud resultó ser casi la misma, y que incluso, con un 10 %

de la muestra se obtuvieron indicadores mejores que los del algoritmo base aleatorio⁴. En redes P2P tales como BitTorrent, eDonkey y Kad también se han aplicado algoritmos de clasificación SVM en conjunto con Regresión Logística Regularizada (LR) para poder determinar el origen del material pornográfico infantil analizando el nombre del archivo y su descripción [Panchenko *et al.*, 2012]. Resultados preliminares lograron obtener un 96,97 % de exactitud al distinguir títulos de origen pornográfico versus títulos extraídos de la Wikipedia (como contraparte). Otro estudio llevado a cabo en la red social Telegram usó *web scrapping*⁵ para extraer todas aquellas conversaciones que contenían la palabra “cp”, que es una abreviatura en inglés para *child pornography* (pornografía infantil) [Packeer y Kannangara, 2022]. De los 15.297 mensajes recolectados, 2.424 contenían efectivamente el término “cp”, además, se rescataron 53 invitaciones a nuevos grupos y 407 enlaces externos de material ilegal. Mediante procesamiento de imágenes, se clasificó el contenido audiovisual por edad y género, describiendo las siguientes proporciones: un 43 % contenían menores entre 10 a 15 años; un 29 % entre 5 a 10 años; un 27 % entre 15 a 17 años y un 3 % entre 1 a 5 años. De todos los casos, las mujeres conformaron el 66 %.

Muchos de estos depredadores, expresan emociones de miedo o rabia ante la frustración de ser capturados o de no poder obtener lo que quieren y esto hace que en sus conversaciones existan patrones tales como la inestabilidad e inmadurez emocional, inferioridad, soledad y baja autoestima [Bogdanova *et al.*, 2014]. Determinar la personalidad con base en los variados ánimos presentes en un texto donde hay un diálogo sexuado entre adultos versus uno donde se involucra a un menor es un desafío, porque hay muchos comportamientos depravados que, en ningún caso, pretenden acercarse a lo que se entiende por *grooming*. Lo que se ha hecho es generar un análisis semántico usando representaciones distribuidas del contexto de las diferentes conversaciones anómalas aplicando Word2Vec y redes convolucionales (CNN) sobre el conjunto de datos “PAN2012” obteniendo un 88 % de exactitud en la detección [Muñoz *et al.*, 2021]. Otras arquitecturas basadas en redes neuronales obtuvieron efectos similares: 91,57 % en exactitud y 80,87 % en el Valor F para One Hot Vector con CNN; 89,46 % en exactitud y 80,58 % en el Valor F para Word2Vector con CNN; 91,71 % en exactitud y 79,85 % en el Valor F con FNN [Ebrahimi *et al.*, 2016]. Desde una mirada psicolingüística, se ha demostrado que hay una conexión entre la personalidad de un sujeto y las palabras que utiliza [Pennebaker *et al.*, 2003]. Usando *partes de la oración*⁶ en modelos SVM combinados con LIWC y TF-IDF, se han logrado extraer varias características de estos perfiles, entre ellas, que los victimarios establecen conversaciones personales más que grupales, suelen chatear en la medianoche, usan pronombres en primera persona y que en la fase donde establecen la confianza, usan palabras relacionadas con el amor, el tiempo y lugares [Parapar *et al.*, 2014].

Otros esfuerzos se han visto reflejados en el perfilado de estos autores utilizando ritmos de

⁴Es aquel que elige si una línea pertenece a una clase (o no) de manera aleatoria, por lo que teóricamente su exactitud es de un 50 %, ya que existen solo dos posibilidades.

⁵También conocido como *web harvesting* o *web data extraction*, se refiere a la extracción de datos por medio de sitios web usando herramientas automatizadas.

⁶También, conocido en inglés como *parts of speech* (POS), describe nueve componentes: sustantivos, adjetivos, artículos, pronombres, verbos, adverbios, conjunciones y preposiciones [Editorial Etecé, 2022].

tipeado (*keystroke dynamics*) y estilometría. Con respecto al primero, encontramos principalmente detecciones de género. Uno de ellos, utiliza SVM sobre de los datos presentes en el conjunto “GREYC-KeyStroke” [Giot *et al.*, 2009] considerando 5 características: PR (el tiempo que se mantiene presionada una tecla), RP (la latencia entre la liberación de una tecla y la pulsación de otra), PP (las latencias entre cada pulsación de teclas), RR (las latencias entre cada liberación de teclas) y una combinación de las anteriores. Resultados arrojaron un 91 % de exactitud [Giot y Rosenberger, 2012]. De manera similar, se experimentó sobre una red social basándose en el conjunto de datos “GREY”, esta vez, aplicando KNN, árboles de decisión (DT), clasificadores Naive-Bayes y técnicas de fusión⁷ [Fairhurst y Da Costa-Abreu, 2011]. También, se han usado “esquemas de votaciones” tales como el **voto mayoritario** (también conocido como *majority voting* o *hard voting*) a través de un conjunto de textos breves de entre 17 a 24 caracteres logrando un 92,1 % de exactitud [Syed Idrus *et al.*, 2014].

Ahora, en el segundo caso, se ha demostrado que una combinación entre el contenido mismo del texto (*content-based features*) y los estilos o rasgos de la escritura (*style-based features*) han logrado ser una combinación eficaz en la detección de género y edad [Argamon *et al.*, 2009]. Las “características del contenido” suelen ser las palabras más frecuentes del conjunto de datos (aunque es muy común que sea un diccionario externo, ya que todo depende de los objetivos que persiga la investigación), mientras que las “características del estilo” suelen incluir palabras gramaticales⁸, partes de la oración, jergas o modismos (*slang words*) y número de caracteres [Goswami *et al.*, 2009]. Algunas aplicaciones de estas han utilizado n-gramas junto con descripciones morfológicas, léxicas y semánticas para distinguir adultos versus adolescentes [Cano *et al.*, 2014]. En la competencia “PAN 2013”, por ejemplo, se utilizaron características del estilo tales como la frecuencia de los signos de puntuación, de las letras en mayúsculas, del uso de comillas, emoticones, direcciones *web* y partes de la oración, mientras que en las características del contenido se hizo uso de BOW, TF-IDF, palabras basadas en diccionario, palabras sentimentales (*sentiment words*) y jergas [Rangel *et al.*, 2013]. En “PAN 2016” se evidenció algo similar pero con un fuerte enfoque en características del estilo tales como la frecuencia de uso de vocablos específicos, entre ellos, palabras gramaticales y jergas [Rangel *et al.*, 2016].

Mientras más nos adentramos en los trabajos realizados por distintos investigadores, notamos que las mismas técnicas van apareciendo, pero el formato de los datos con que se entrenan a estos modelos es lo que marca la diferencia. Ahora, es importante destacar que las redes neuronales no han sido la alternativa predilecta a la hora de solucionar estos problemas, salvo contadas excepciones [Ebrahimi *et al.*, 2016, Misra *et al.*, 2019, Muñoz *et al.*, 2021, Ngejane *et al.*, 2021, Zuo *et al.*, 2019]. Es comprensible, por un lado, ya que tareas como la detección de *grooming* (así también alguna de sus etapas) o la identi-

⁷Las técnicas de fusión son clasificadores de selección dinámica (*dynamic classifier selection*), es decir, se entrenan para elegir al clasificador más idóneo en función de la entrada. Basan sus reglas usando DCS-LA (*dynamic classifier selection on local accuracy*), voto mayoritario (*majority voting*) y suma basada en reglas (*sum rule-based*).

⁸Es común encontrar su traducción literal como *palabras funcionales* pues deriva del término *functional words* propuesto por Fries para referirse a todo vocablo cuyo significado es funcional o estructural, no léxico. Algunos ejemplos son los pronombres, preposiciones o conjunciones [Hallebeek, 1986].

ficación de depredadores, solo involucran dos categorías, y clasificadores como SVM están diseñados bajo estos parámetros [Evgeniou y Pontil, 2001]; otras sospechas podrían deberse a la interpretabilidad de las decisiones tomadas en las capas ocultas [Li *et al.*, 2021], sobre todo, en tareas que vayan más allá de la simple detección como puede ser el perfilado. Resulta curioso que esta ausencia exista en una de las arquitecturas claves en la consolidación de lo que es el estado del arte del procesamiento de lenguaje natural y es por ello que adherir nuevos matices tecnológicos y metodológicos resulta imperante.

1.5. Objetivos

1.5.1. Objetivo general

Considerando entonces todos los peligros asociados a la falta de herramientas y control parte de los tutores sumados a la exposición desmesurada de los menores en Internet, el propósito de esta memoria será desarrollar un sistema de procesamiento de texto capaz de discriminar sentencias con fines perversos (manipulación, explotación o abuso sexual) hacia menores utilizando un modelo de inteligencia artificial BERT.

1.5.2. Objetivos específicos

1. Determinar los riesgos existentes en Internet, y así, entender cómo pueden gatillar escenarios de manipulación, explotación o abuso sexual infantil.
2. Analizar el estado del arte en cuanto a la detección y prevención de *grooming* en Internet para establecer qué fuentes de datos son utilizadas, qué estrategias se emplean durante el procesamiento de los datos y cuáles modelos resultan tener mejores desempeños.
3. Implementar un modelo de inteligencia artificial BERT y ajustar parámetros para adecuarlo al contexto del problema y mejorar la capacidad de generalización.
4. Aplicar solución en casos reales y comentar resultados para corroborar eficacia y futuras mejoras.

CAPÍTULO 2

MARCO CONCEPTUAL

Las redes neuronales artificiales (ANN) son un modelo computacional inspirado por los procesos cognitivos llevados a cabo en el cerebro [McCulloch y Pitts, 1943]. Se caracterizan por ser capaces de modificar su “estructura interna” con relación a una función objetivo. Se aplican en problemas cuya naturaleza es de carácter (principalmente) no lineal. Los componentes elementales son los siguientes:

- Las **neuronas, nodos**, o simplemente, **unidades de procesamiento**, las cuales reciben una entrada proveniente desde otros nodos, o bien, desde el “ambiente”, y a su vez, entregan una salida hacia otros nodos o hacia el “ambiente” mediante el uso de **funciones de activación**.
- Las **capas**, que como mínimo deben ser tres: una para las entradas, otra para las transformaciones y una última para las salidas. Es usual llamar a las capas que están entre las entradas y salidas como **capas ocultas**.
- La **función de pérdida** para poder calcular los gradientes y así optimizar al modelo.

Las conexiones entre neuronas están ponderadas de tal forma que, mientras más negativas sean, más se inhiben (o repelen), y mientras más positivas sean, más se potencian. Este arreglo de pesos es lo que se aprende cuando se entrenan a los modelos.

2.1. Unidades de procesamiento

Si m es el número total de capas en la red, r_k el número de unidades de procesamiento en la k -ésima capa con $k \in \mathbb{N}$ tal que $2 \leq k \leq m$ (la primera capa no posee parámetros entrenables al ser la entrada), la salida de la i -ésima neurona simbolizada por $o_i^{(k)} \in \mathbb{R}$ luego de procesar el vector de entrada $\mathbf{x}^{(k)} \in \mathbb{R}^{r_{k-1}}$ se expresa de la siguiente manera:

$$o_i^{(k)} = f_k \left(\mathbf{w}_i^{(k)} \cdot \left(\mathbf{x}^{(k)} \right)^\top + b_i^{(k)} \right) = f_k \left(a_i^{(k)} \right), \quad (1)$$

donde:

- $\mathbf{w}_i^{(k)} \in \mathbb{R}^{r_{k-1}}$ es el vector de pesos para la i -ésima neurona en la capa k .
- $b_i^{(k)} \in \mathbb{R}$ es el sesgo de la i -ésima neurona en la capa k .

- $a_i^{(k)} \in \mathbb{R}$ es la preactivación de la i -ésima neurona en la capa k .
- $f_k : \mathbb{R} \rightarrow \mathbb{R}$ es la función de activación en la capa k . No suele ser común utilizar una función de activación distinta para cada neurona en cada capa.

En la práctica, esta definición se extiende de manera matricial de forma que la salida de la k -ésima capa genere un vector $\mathbf{o}^{(k)}$:

$$\mathbf{o}^{(k)} = \vec{f}_k (W^{(k)} \cdot \mathbf{x}^{(k)} + \mathbf{b}^{(k)}) = \vec{f}_k (\mathbf{a}^{(k)}), \quad (2)$$

donde:

- $W^{(k)} \in \mathbb{R}^{r_k \times r_{k-1}}$ es la matriz de pesos en la capa k .
- $\mathbf{b}^{(k)} \in \mathbb{R}^{r_k}$ es el vector de sesgos en la capa k .
- $\mathbf{a}^{(k)} \in \mathbb{R}^{r_k}$ es el vector de preactivaciones para la capa k .
- $\vec{f}_k : \mathbb{R}^{r_k} \rightarrow \mathbb{R}^{r_k}$ es la función de activación vectorial para la capa k y opera de forma que $(\vec{f}_k)_i = f_k(a_i^{(k)})$.

Vemos que la matriz $W^{(k)}$ se define de tal forma que hace que no sea necesario realizar una operación de transposición, ya que suelen generar un costo computacional no menor. Por otro lado, se debe comprender que $\mathbf{o}^{(k)}$ viene a ser la entrada para la capa $k + 1$, es decir, $\mathbf{x}^{(k+1)}$, de esta forma, la ecuación (2) puede ser reformulada como sigue:

$$\mathbf{o}^{(k)} = \vec{f}_k (W^{(k)} \cdot \mathbf{o}^{(k-1)} + \mathbf{b}^{(k)}), \quad (3)$$

con la particularidad de que para $k = 1$, $\mathbf{o}^{(k)}$ no es más que la entrada del modelo, y para $k = m$, la salida.

2.2. Funciones de activación

En principio, el dominio de cada una de estas se define como $f_k : \mathbb{R} \rightarrow \mathbb{R}$ (en el sentido de que entregan escalares) porque operan a nivel de una neurona. Si bien existe una gran variedad, nos reduciremos a explicar aquellas que serán utilizadas durante esta memoria.

En primer lugar, encontramos a la tangente hiperbólica. Su dominio está expresado como $\tanh : \mathbb{R} \rightarrow [-1, 1]$ y su definición es la siguiente:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} . \quad (4)$$

En segundo lugar, encontramos a la unidad lineal rectificada. Su dominio está expresado como $\text{ReLU} : \mathbb{R} \rightarrow \mathbb{R}$ y su definición es la siguiente:

$$\text{ReLU}(x) = \max(x, 0) . \quad (5)$$

En tercer y último lugar, encontramos a la sigmoide. Su dominio está expresado como $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ y su definición es la siguiente:

$$\sigma(x) = \frac{1}{1 + e^{-x}} . \quad (6)$$

La función (4) se suele situar en las capas ocultas porque puede mapear cualquier valor real a un intervalo cerrado y acotado que es capaz de preservar el espacio dentro del plano de las entradas, es decir, que si las preactivaciones son negativas, las activaciones también lo serán (y viceversa). En este mismo contexto, la función (5) también se tiende a aplicar en las capas ocultas por dos razones: es menos costosa computacionalmente y permite crear gradientes más dispersos (esto es, con un significativo número de derivadas cercanas o iguales a cero). Por otra parte, la función (6) generalmente se trabaja en la última capa porque sus salidas pueden ser interpretadas como probabilidades.

2.3. Funciones de pérdida

Permiten cuantificar el error de las respuestas generadas por los modelos. Como se expresen depende exclusivamente de la naturaleza del problema. En nuestro caso, utilizaremos la **entropía cruzada**, ya que realizaremos clasificaciones en la última capa.

Si $\mathbf{o}^{(m)} \in \mathbb{R}^{r_m}$ son las predicciones del modelo tal que $\mathbf{o}_i^{(m)} \in \mathbb{R}$ es la probabilidad de que la entrada se clasifique como i , y a su vez, $\mathbf{y} \in \{0, 1\}^{r_m}$ los valores a predecir, la entropía cruzada se puede definir como:

$$L(\mathbf{y}, \mathbf{o}^{(m)}) = - \sum_{i=1}^{r_m} \mathbf{y}_i \cdot \log(\mathbf{o}_i^{(m)}) . \quad (7)$$

Cuando $r_m = 1$, la expresión (7) se denomina **entropía cruzada binaria**, por lo que $\mathbf{o}^{(m)} \in \mathbb{R}$ tanto como $\mathbf{y} \in \mathbb{R}$:

$$L(\mathbf{y}, \mathbf{o}^{(m)}) = - \left(\mathbf{y}_1 \cdot \log(\mathbf{o}_1^{(m)}) + (1 - \mathbf{y}_1) \cdot \log(1 - \mathbf{o}_1^{(m)}) \right). \quad (8)$$

2.4. Optimización de los modelos

Existen dos procesos fundamentales para poder mejorar la capacidad de generalización:

1. La **propagación hacia adelante**, que es lo que permite generar las respuestas en función de las entradas.
2. La **propagación hacia atrás**, que es lo que permite modificar los pesos en función de los errores de las actuales predicciones. Se suele utilizar el **gradiente descendente** (o modificaciones de él).

En función del tipo de red, es cómo se implementarán cada uno de estos procesos.

2.5. Propagación hacia adelante

No es más que aplicar (de manera sistemática) la definición de la ecuación (3) para $k = m$. Desde ya, notamos que el flujo de transmisión fuerza a que se tengan precalculadas las salidas de las unidades más cercanas a la entrada.

2.6. Propagación hacia atrás

La clave de este proceso está en conocer dónde es que la función de pérdida (7) tiende a minimizarse. Lo que se suele hacer es calcular el gradiente en función de un vector $\mathbf{g} \in \mathbb{R}^r$ que contiene a todos los pesos del modelo como variables. Por su parte, r (el número de parámetros totales en la red) queda expresado como:

$$r = \sum_{k=1}^{m-1} \underbrace{r_k \cdot r_{k+1}}_{\text{pesos}} + \underbrace{r_{k+1}}_{\text{sesgo}}. \quad (9)$$

Entonces, el algoritmo de propagación hacia atrás (BP) es la solución (de manera algorítmica) a la siguiente expresión:

$$\frac{\partial L(\mathbf{y}, \mathbf{o}^{(m)})}{\partial \mathbf{g}} = \nabla L(\mathbf{y}, \mathbf{o}^{(m)}) .$$

Antes de formular matemáticamente el procedimiento, redefinamos convenientemente la preactivación de una neurona de forma que quede descrita únicamente por escalares y no vectores. Sea $w_{ij}^{(k)}$ el peso que recibe el nodo i en la capa k producto de la conexión con el nodo j en la capa $k - 1$. La preactivación de una unidad de procesamiento queda descrita ahora como:

$$a_i^{(k)} = b_i^{(k)} + \sum_{j=1}^{r_{k-1}} w_{ji}^{(k)} \cdot o_j^{(k-1)} .$$

Con todo en mano, las derivadas en las **capas ocultas** se pueden calcular de la siguiente manera:

$$\frac{\partial L}{\partial w_{ij}^{(d)}} = \sum_{k=1}^{r_m} \frac{\partial L}{\partial o_k^{(m)}} \cdot \frac{\partial o_k^{(m)}}{\partial a_k^{(m)}} \cdot \delta_k^{(m)} ,$$

donde $d \in \mathbb{N}$ tal que $2 \leq d \leq m - 1$ y $\delta_k^{(m)}$ (también llamado “error”) se define como:

$$\delta_k^{(m)} = \begin{cases} \sum_{l=1}^{r_{m-1}} \frac{\partial a_k^{(m)}}{\partial o_l^{(m-1)}} \cdot \frac{\partial o_l^{(m-1)}}{\partial a_l^{(m-1)}} \cdot \delta_l^{(m-1)} & m \geq d + 2 , \\ \frac{\partial a_k^{(m)}}{\partial o_i^{(m-1)}} \cdot \frac{\partial o_i^{(m-1)}}{\partial a_i^{(m-1)}} \cdot \frac{\partial a_i^{(m-1)}}{\partial w_{ij}^{(m-1)}} & m = d + 1 . \end{cases}$$

En cambio, las derivadas en la **capa de salida** se expresan como:

$$\frac{\partial L}{\partial w_{ij}^{(m)}} = \frac{\partial L}{\partial o_i^{(m)}} \cdot \frac{\partial o_i^{(m)}}{\partial a_i^{(m)}} \cdot \frac{\partial a_i^{(m)}}{\partial w_{ij}^{(m)}} .$$

La única diferencia práctica entre el caso (7) y (8) radica en el término $\partial L / \partial o_k^{(m)}$.

2.7. Gradiente descendente

Debido a la alta complejidad de las redes neuronales, ya sea en términos del número de variables y de las funciones de activación como transformaciones no lineales, utilizar métodos exactos para poder encontrar soluciones factibles a problemas se torna computacionalmente demandante, e incluso, se puede no alcanzar una respuesta en tiempos razonables, motivo por el cual se prefieren utilizar algoritmos iterativos como el gradiente descendente (GD). Dado que la función a optimizar se expresa por (7), el algoritmo queda descrito de la siguiente forma:

$$\mathbf{g}_{i+1} = \mathbf{g}_i - \eta \cdot \nabla L(\mathbf{y}, \mathbf{o}^{(m)}), \quad (10)$$

donde $\eta \in (0, 1)$ es la tasa de aprendizaje.

2.8. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (NLP) es una rama de la inteligencia artificial que busca comprender enunciados de palabras pertenecientes a un lenguaje mediante sistemas computacionales, cuyos campos de estudio pueden clasificarse en dos partes: entendimiento del lenguaje natural (*natural language understanding*) y generación del lenguaje natural (*natural language generation*). Las tareas que actualmente se desarrollan en dichos campos incluyen:

- Resumen automático (*automatic summarization*): Produce un resumen entendible de un texto [Retkowski, 2023].
- Resolución de correferencia (*co-reference resolution*): Permite determinar cuáles palabras se refieren a un mismo objeto [Ponzetto y Poesio, 2009].
- Análisis de discurso (*discourse analysis*): Analiza cómo el significado de las ideas se ve influenciado por la fonética, fonología, morfología, semántica y sintaxis en discursos escritos y hablados [Joty et al., 2019].
- Traducción automática (*machine translation*): Traduce automáticamente textos desde un lenguaje humano a otro [Lample et al., 2018, Gu et al., 2018].
- Segmentación morfológica (*morphological segmentation*): Separa palabras en morfemas individuales e identifica sus respectivos tipos [Cotterell et al., 2019].
- Reconocimiento de entidad nombrada (*named entity recognition*): Determina cuáles palabras en un texto se refieren a entidades y las categoriza (como personas, organizaciones o lugares) [Baevski et al., 2019, Primer, 2022].

- Reconocimiento óptico de caracteres (*optical character recognition*): Dada una imagen, produce una interpretación o transcripción de lo que se ve en texto [Rodríguez *et al.*, 2022, Olejniczak y Šulc, 2023].
- Etiquetado de partes de un discurso (*part of speech tagging*): Describe oraciones en función de sustantivos, pronombres, verbos, adjetivos, adverbios, preposiciones, conjunciones e interjecciones [Demilie, 2020, Sánchez-Martínez *et al.*, 2008, Singh *et al.*, 2013].

Uno de los desafíos más grandes dentro de estos estudios es la ambigüedad, no solo a nivel de palabras, sino a nivel de sentencias: en cierto sentido, entender un texto no es solo generar el análisis sintáctico y morfológico, es también, interpretar las ideas de un emisor bajo un contexto y un medio que altera el fin mismo del mensaje. Existen técnicas que permiten acaparar esta dificultad, entre ellas:

- Minimizar ambigüedad (*minimising ambiguity*): Propone utilizar palabras y construcciones lingüísticas que eviten inducir ambigüedades en un texto. El resultado da origen a un lenguaje controlado (CL) [Umber y Sarwar, 2011].
- Preservar ambigüedad (*preserving ambiguity*): En vez de eliminar la ambigüedad, la preserva, con el objeto de crear un árbol de significados y usar un cuerpo estadístico en su análisis [Shemtov, 1997, Gao *et al.*, 2015, Umber y Sarwar, 2011].
- Desambiguación interactiva (*interactive disambiguity*): Se analiza la causa de la ambigüedad (ya sea léxica o estructural) y se proponen distintas alternativas a un usuario para solucionarla [Shemtov, 1997].

Si tratáramos de aventurarnos en alguna de las ramas que se presentaron más arriba utilizando el modelo propuesto en la sección (2.1), rápidamente nos enfrentaríamos a los siguientes problemas:

1. El formato de las entradas: Sabemos que los caracteres internamente son tratados como números, sin embargo, si quisiéramos emplear dicha representación en nuestros modelos, veríamos que se presentarían precedencias cuando en la realidad, estos símbolos no siempre siguen un orden como tal, además, notaríamos que palabras con significados similares tendrían valores diferentes.
2. Los textos presentan un largo variable: De acuerdo a como planteamos la arquitectura, las unidades de entrada siempre tienen un tamaño fijo.
3. Capturar la relación entre palabras: Las entradas en nuestra red son independientes entre sí y no tienen cómo entender alguna dependencia temporal.

Para dar solución a (1), se deben emplear técnicas relacionadas con la **vectorización de textos**, mientras que para (2) se deben emplear rellenos (*padding*s) que irán en función del formato definido en (1) y para (3) resulta necesario proponer una nueva familia de modelos llamados **redes neuronales recurrentes** (RNN).

2.9. Vectorización de textos

La necesidad de transformar los textos en vectores reales surge porque redes como la definida en la sección (2.1) aplican transformaciones lineales y no lineales a las entradas que son únicamente numéricas. En función del problema, tendremos un dominio de palabras que llamaremos **vocabulario** y que podremos ajustar de acuerdo a la frecuencia de estas o por algún otro criterio.

Una de las primeras ideas que pueden surgir es enumerar todos los elementos de un vocabulario $\mathcal{V}$ y asignarle un vector $\mathbf{v}_i \in \{0, 1\}^{|\mathcal{V}|}$ donde todos sus elementos son cero excepto en la i -ésima posición cuyo valor es 1. Esto es conocido como *one-hot encoding* y es un enfoque que resulta ser ineficiente por dos cosas:

- Genera muchos vectores *sparse*, ya que por cada uno tendremos $|\mathcal{V}| - 1$ elementos cero.
- Crece de manera proporcional al vocabulario, lo que puede ocasionar problemas de memoria.

Si ahora quisiéramos utilizar el mismo enfoque, pero rellenando dichos vectores con elementos distintos a cero, también presentaríamos dos desventajas:

- La codificación sería arbitraria y no capturaría ninguna relación entre palabras.
- Sería difícil de interpretar para una red en el sentido semántico porque dos palabras con significados similares tendrían codificaciones distintas, y a nivel de procesamiento, dos resultados difícilmente parecidos.

Lo que se suele hacer es utilizar **incrustación de palabras** (*word embedding*), ya que presenta dos ventajas:

- Genera vectores densos cuyas dimensiones son especificadas manualmente.
- Produce codificaciones parecidas entre palabras que guardan un significado similar.

2.9.1. Incrustación de palabras

El planteamiento consta en incluir inmediatamente después de la capa de entrada una capa de “incrustado” (*embedding*) que tendrá r_e unidades de procesamiento con una función de activación $f_e(x) = x$. Las entradas en la red serán enteros que representen el índice de la palabra dentro del vocabulario $\mathcal{V}$, las que, luego, serán transformadas a una matriz $H_e \in \mathbb{R}^{r_1 \times |\mathcal{V}|}$ de vectores *one-hot*. Entonces, dada una matriz de pesos $W_e \in \mathbb{R}^{|\mathcal{V}| \times r_e}$, el vector de “incrustado” puede ser expresado por:

$$\mathbf{o}^{(e)} = f_e(H_e \cdot W_e). \quad (11)$$

Cuando simplificamos (11), obtenemos que el i -ésimo elemento del vector de “incrustado” no es más que la i -ésima fila de la matriz W_e que simbolizaremos como $(W_e)_{i*}$, es decir:

$$\mathbf{o}_i^{(e)} = (W_e)_{i*} \in \mathbb{R}^{r_e}. \quad (12)$$

Gracias a la definición (11), se puede aplicar el algoritmo de propagación hacia atrás para poder entrenar a los parámetros W_e , sin embargo, durante el algoritmo de propagación hacia adelante, se opta por algo parecido a lo que (12) enuncia por dos motivos:

- Optimiza el uso de recursos al evitar crear r_1 vectores *one-hot* de dimensiones $|\mathcal{V}|$.
- Evita el cálculo matricial y retorna las operaciones en un tiempo constante $O(r_1)$.

Las palabras que los elementos del vector de entrada $\mathbf{o}^{(1)}$ identifica son llamadas **tókenes** (*tokens*) porque son una descomposición atómica de una oración, e incluso, de una palabra misma.

2.9.2. Word2vec

Esta es otra familia de algoritmos igualmente popular que se caracteriza por aprender el cuerpo de un lenguaje. Esto significa que no solo es capaz de entender la similitud entre palabras, sino que aspectos como la precedencia y ciertas reglas (de manera abstracta). Se puede materializar de dos formas [Mikolov *et al.*, 2013a, Mikolov *et al.*, 2013b]:

- Como modelo de “bolsa de palabras” continuo (CBOW): Predice **una palabra** intermedia en función de otras que le rodeen más adelante y más atrás (entenderemos esto como el contexto). Su salida es lo más parecido a un autocompletado. La arquitectura de esta propuesta queda reflejada a través de la figura (1). En la entrada, se

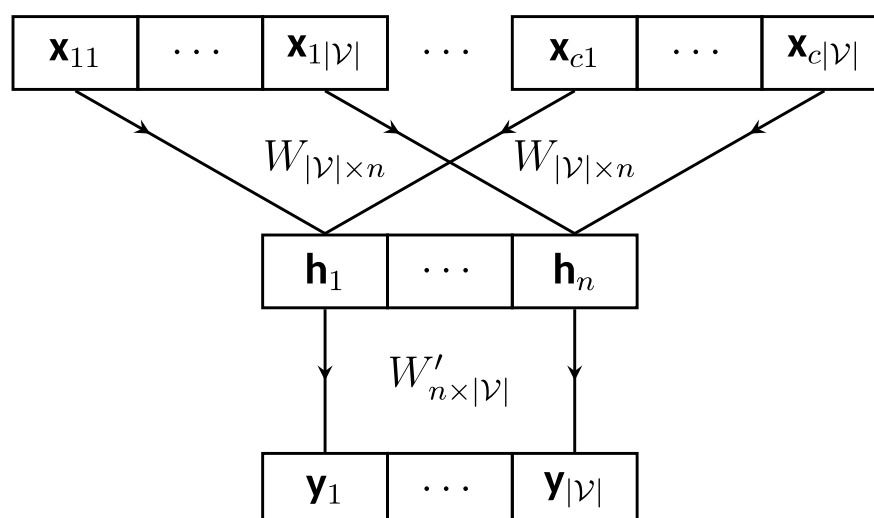


Figura 1: Modelo de “bolsa de palabras” continuo (CBOW). No se graficaron todas las conexiones entre nodos bajo el fin de simplificar el diagrama.

reciben c vectores $\mathbf{x}_i \in \{0, 1\}^{|\mathcal{V}|}$ de tipo “one-hot”, donde $|\mathcal{V}| \in \mathbb{N}$ es el tamaño del vocabulario $\mathcal{V}$. Mediante la matriz $W \in \mathbb{R}^{|\mathcal{V}| \times n}$, se genera un vector de “incrustado” $\mathbf{h} \in \mathbb{R}^n$. Finalmente, se utiliza otra matriz $W' \in \mathbb{R}^{n \times |\mathcal{V}|}$ para realizar la predicción de la palabra intermediaria $\mathbf{y} \in \mathbb{R}^{|\mathcal{V}|}$ (también codificada como “one-hot”). Para obtener los “incrustados”, basta con extraer la matriz W y multiplicarla con algún otro vector $\mathbf{w} \in \mathbb{R}^{|\mathcal{V}|}$, es decir:

$$\text{embedding}(\mathbf{w}) = W^T \cdot \mathbf{w}.$$

- Como modelo “salta gramas”⁹ (*skip-gram model*): Predice un **conjunto de palabras** en función de otras que le rodeen más adelante y más atrás. La arquitectura de esta propuesta queda reflejada a través de la figura (2). En la entrada, se recibe un vector $\mathbf{x} \in \{0, 1\}^{|\mathcal{V}|}$ de tipo “one-hot”, donde $|\mathcal{V}| \in \mathbb{N}$ es el tamaño del vocabulario $\mathcal{V}$. Mediante la matriz $W \in \mathbb{R}^{|\mathcal{V}| \times n}$, se genera un vector de “incrustado” $\mathbf{h} \in \mathbb{R}^n$. Finalmente, se utiliza otra matriz $W' \in \mathbb{R}^{n \times |\mathcal{V}|}$ para realizar la predicción de las c palabras de contexto $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{V}|}$ de tipo “one-hot”. Para obtener los “incrustados”, basta con extraer la matriz W y multiplicarla con algún otro vector $\mathbf{w} \in \mathbb{R}^{|\mathcal{V}|}$, es decir:

$$\text{embedding}(\mathbf{w}) = W^T \cdot \mathbf{w}.$$

Como se puede apreciar, esta metodología es la contraparte de la otra.

2.10. Procesamiento de textos

En una oración, es probable que existan palabras redundantes o que carezcan de un sentido por sí mismas con relación a lo que se desea expresar. Si bien nos ayudan a cohesionar

⁹Los “gramas” son cada uno de los trazos que forman una letra y en los cuales se descomponen para su descripción.

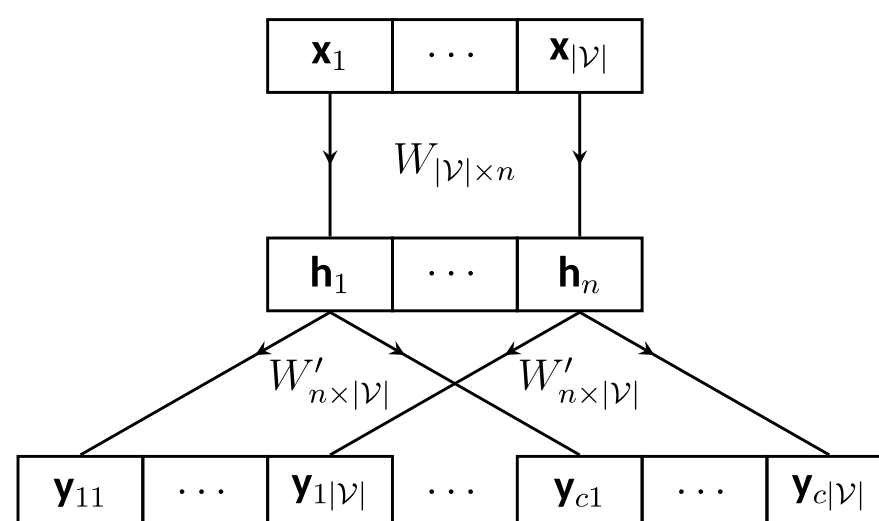


Figura 2: Modelo “salta gramas” (*skip-gram model*). No se graficaron todas las conexiones entre nodos bajo el fin de simplificar el diagrama.

conceptos, en lo que a las redes neuronales respecta, terminan a veces transformándose en ruido. Esta etapa previa a la descrita en la sección (2.9) nos permite establecer dos aspectos críticos:

- ¿Qué palabras en el conjunto de datos pertenecerán al vocabulario $\mathcal{V}$?
- ¿Cómo se descompondrán las oraciones en tókenes?

Este apartado de tareas se suele llamar **tokenización** (*tokenization*) y frecuentemente está más orientado a entregar palabras que caracteres.

2.10.1. Tokenización basada en espacios

Es una de las primeras formas para poder dividir un texto en palabras. Veamos, por ejemplo, la oración en inglés “Don’t you love transformers? We sure do.”. El resultado se puede apreciar en el módulo (1)

```
tokenization = ["Don't", "you", "love", "transformers?", "We",
               ↪ "sure", "do."]
```

Módulo 1: El resultado de aplicar la tokenización basada en espacios para la oración “Don’t you love transformers? We sure do.” como una lista de *strings*.

Notamos que los signos de puntuación para los casos “Don’t”, “transformers?” y “do.” están incluidos. Esto no es deseable porque significa que el modelo tendrá que aprender diferentes representaciones para una misma palabra. Si ahora tomáramos en cuenta estos símbolos, obtendríamos el resultado del módulo (2).

```
tokenization = ["Don", "'", "t", "you", "love", "Transformers",
    ↪  "?", "We", "sure", "do", "."]
```

Módulo 2: El resultado de aplicar la tokenización basada en espacios para la oración “Don’t you love transformers? We sure do.” considerando signos de puntuación como una lista de *strings*.

Resulta mejor, pero no es capaz de responder ante casos como la palabra “Don’t” que es una contracción de “do not”. Aquí es donde entra la arbitrariedad de cómo resolver estas y otras situaciones, porque en general dependen de dos cosas: el problema y el modelo. En casos donde existe un *corpus* masivo de textos, este tipo de tokenizadores (*tokenizers*) tienden a crear **vocabularios extremadamente grandes** como en el caso del modelo “Transformer XL”, cuyo tamaño alcanza los 267.735 elementos [Dai *et al.*, 2019]. Un vocabulario grande fuerza a tener matrices enormes en las capas de entrada y salida, que a la larga, ocasionan un uso desmedido de la memoria y una complejidad temporal alta.

2.10.2. Tokenización basada en caracteres

Busca ir un paso más allá dividiendo las oraciones de manera atómica. Si continuamos con el ejemplo anterior, veríamos algo similar al módulo (3).

```
tokenization = ["D", "o", "n", "'", "t", "y", "o", "u", "l", "o",
    ↪  "v", "e", "t", "r", "a", "n", "s", "f", "o", "r", "m", "e",
    ↪  "r", "s", "?", "W", "e", "s", "u", "r", "e", "d", "o"]
```

Módulo 3: El resultado de aplicar la tokenización basada en caracteres para la oración “Don’t you love transformers? We sure do.” como una lista de *strings*.

Los vocabularios obtenidos acá tienden a ser más reducidos que los obtenidos en la sección (2.10.2), además de que operan bajo reglas sencillas y óptimas a nivel de memoria, sin embargo, para los modelos resulta mucho más difícil aprender los significados y relaciones entre las entradas porque los caracteres no son capaces de entregar tanto contexto como las palabras.

Lo que se suele hacer en la práctica es usar estrategias híbridas a nivel de vocablos y caracteres materializadas a través de **algoritmos de tokenización por subpalabras** (*subword tokenization algorithms*).

2.10.3. Tokenización basada en subpalabras

Es una familia de algoritmos y se basan en la premisa de que las palabras más usadas no deberían ser divididas en subpalabras, y las que menos frecuencia posean, sí. Por ejemplo, “annoyingly” podría ser considerada una palabra rara, por lo que su descomposición sería algo como “annoying” y “ly”. Ambas subpalabras aparecerían más frecuentemente, y además, el significado de “annoyingly” se mantendría mediante una **composición de tókenes**.

Los vocabularios generados aquí tienden a tener tamaños razonables, permiten procesar palabras nunca antes vistas mediante la división de las entradas en subpalabras y son capaces de generar representaciones significativas independientemente del contexto de las oraciones.

2.10.4. WordPiece

Es un algoritmo de tokenización basado en subpalabras usado en BERT, DistilBERT y Electra. Se introdujo como una optimización en los procesos de búsqueda por voz para los idiomas japonés y coreano [Schuster y Nakajima, 2012]. Se caracteriza por aprender a combinar pares de símbolos que juntos resultan ser más frecuentes que separados. A continuación, describiremos una **aproximación del algoritmo** para el caso BERT, ya que el equipo de Google nunca ha dado a conocer abiertamente su código, así que es probable que existan discrepancias mínimas, pero después de todo, no deberían afectar a la idea en sí.

2.10.4.1. Generación del vocabulario Supongamos que contamos todas las palabras de nuestro conjunto de datos y obtenemos la distribución mostrada en el módulo (4).

```
word_frequencies = [("hug", 10), ("pug", 5), ("pun", 12), ("bun",  
↪ 4), ("hugs", 5)]
```

Módulo 4: Distribución de las frecuencias absolutas de cada palabra presente en un hipotético conjunto de datos. Se expresa como una lista de tuplas, donde el primer elemento es la palabra en sí, mientras que el segundo, la contabilización.

El primer paso es generar el **vocabulario inicial**. Para lograrlo, obtenemos el primer carácter de cada palabra, y el resto, los precedemos por el texto “##”. El resultado podemos verlo en el módulo (5).

Ahora, si combinamos cada uno de los caracteres de las listas y quitamos los duplicados, obtenemos el vocabulario inicial expresado por el módulo (6).

```
splits = {"hug": [("h", 10), ("##u", 10), ("##g", 10)],
          "pug": [("p", 5), ("##u", 5), ("##g", 5)],
          "pun": [("p", 12), ("##u", 12), ("##n", 12)],
          "bun": [("b", 4), ("##u", 4), ("##n", 4)],
          "hugs": [("h", 5), ("##u", 5), ("##g", 5), ("##s", 5)]}
```

Módulo 5: Distribución de las frecuencias absolutas de cada carácter de una palabra presente en un hipotético conjunto de datos. Se expresa como un diccionario, cuyos valores son una lista de tuplas (la primera posición es el carácter en sí, mientras que la segunda, la contabilización). El prefijo sirve para indicar que el elemento es una subpalabra.

```
vocabulary = ["b", "h", "p", "##g", "##n", "##s", "##u"]
```

Módulo 6: El vocabulario inicial como una lista de *strings*.

Para poder unir pares de elementos a y b , se debe calcular un **puntaje** determinado por la siguiente expresión:

$$\text{score}(a, b) = \frac{f_{ab}}{f_a \cdot f_b}, \quad (13)$$

donde:

- $f_{ab} \in \mathbb{N}$ es la frecuencia absoluta de aparición de la combinación ab en el actual *split*.
- $f_a \in \mathbb{N}$ es la frecuencia absoluta de aparición del elemento a en el actual *split*.
- $f_b \in \mathbb{N}$ es la frecuencia absoluta de aparición del elemento b en el actual *split*.

Los elementos que se podrán unir serán aquellos derivados de la misma palabra y lo harán de izquierda a derecha. En nuestro caso, las combinaciones posibles se enumeran en el módulo (7).

Por ejemplo, para calcular (13) usando la combinación “hu”, el actual *split* tomaría la forma del módulo (8).

Con lo anterior en mano, el puntaje se cuantifica así:

$$\text{score}(h, u) = \frac{10 + 5}{(10 + 5) \cdot (10 + 5 + 12 + 4 + 5)} \approx 0,027.$$

```

pairs = {"hug": ["hu", "##ug"],
        "pug": ["pu", "##ug"],
        "pun": ["pu", "##un"],
        "bun": ["bu", "##un"],
        "hugs": ["hu", "##ug", "##gs"]}

```

Módulo 7: Todas las combinaciones posibles de unión entre elementos de la misma palabra partiendo de izquierda a derecha. No es posible entrelazar componentes de distintas palabras (como “hug” y “pug”). Se expresa como un diccionario cuyos valores son listas de *strings*.

```

hu_pair_splits = {"hug": [("hu", 10), ("##g", 10)],
                  "pug": [("p", 5), ("##u", 5), ("##g", 5)],
                  "pun": [("p", 12), ("##u", 12), ("##n", 12)],
                  "bun": [("b", 4), ("##u", 4), ("##n", 4)],
                  "hugs": [("hu", 5), ("##g", 5), ("##s", 5)]}

```

Módulo 8: Distribución de las frecuencias absolutas de cada elemento en el *split* luego de combinar los caracteres “h” y “u” en un hipotético conjunto de datos. Se expresa como un diccionario, cuyos valores son una lista de tuplas, donde el primer elemento es el token en sí, mientras que el segundo la contabilización.

Si replicamos esto mismo para todos los casos, concluiremos que el mejor puntaje lo obtiene la combinación “##g” y “##s” (presente 10 veces) y no la combinación “##u” y “##g” (presente 20 veces). La razón radica en que la frecuencia absoluta de “##u” es muy alta (y por ende el cociente muy bajo), pero más allá de eso, porque (para el algoritmo) resulta más valioso tenerlo aislado que mezclado, ya que es capaz de representar más eficientemente las palabras del conjunto y otras fuera de él. Por lo tanto, la primera unión dejaría como resultado la subpalabra “##gs” que modificaría dos variables: los actuales *splits* y el vocabulario, tal y como podemos apreciar en el módulo (9). Este proceso se repite hasta completar el tamaño deseado del vocabulario.

Como nota, si dos o más combinaciones tienen el **mismo puntaje**, se elige solo una (la precedencia depende de la implementación).

2.10.4.2. Tokenización de las entradas Supongamos que luego de aplicar el algoritmo de la sección (2.10.4.1) obtenemos el vocabulario del módulo (10).

Para procesar la oración “Hong Tong!”, primero se aplica la tokenización por espacios considerando signos de puntuación dando como resultado el módulo (11).

A continuación, para cada token se busca la subpalabra más larga dentro del vocabulario que

```
# Añade el par ##gs
vocabulary = ["b", "h", "p", "##g", "##n", "##s", "##u", "##gs"]

# Combina los elementos ##g y ##s en todos los split
splits = {"hug": [("h", 10), ("##u", 10), ("##g", 10)],
          "pug": [("p", 5), ("##u", 5), ("##g", 5)],
          "pun": [("p", 12), ("##u", 12), ("##n", 12)],
          "bun": [("b", 4), ("##u", 4), ("##n", 4)],
          "hugs": [("h", 5), ("##u", 5), ("##gs", 5)]}
```

Módulo 9: El resultado de aplicar una iteración del algoritmo WordPiece a un hipotético conjunto de datos. Vemos que al vocabulario `vocabulary` se le añade el par “##gs” y la distribución de frecuencias absolutas `splits` combina los elementos “##g” y “##s”.

```
vocabulary = ["##a", "##b", "##c", "##d", "##e", "##f", "##g",
  ↳ "##h", "##i", "##k", "##l", "##m", "##n", "##o", "##p", "##r",
  ↳ "##s", "##t", "##u", "##v", "##w", "##y", "##z", ",", ".",
  ↳ "C", "F", "H", "T", "a", "b", "c", "g", "h", "i", "s", "t",
  ↳ "u", "w", "y", "ab", "##fu", "Fa", "Fac", "##ct", "##ful",
  ↳ "##full", "##fully", "Th", "ch", "##hm", "cha", "chap",
  ↳ "chapt", "##thm", "Hu", "Hug", "Hugg", "sh", "th", "is",
  ↳ "##thms", "##za", "##zat", "##ut"]
```

Módulo 10: Un hipotético vocabulario obtenido luego de aplicar el algoritmo de la sección (2.10.4.1) expresado como una lista de *strings*.

calce de inicio a fin, o bien, hasta una parte de él, e inmediatamente después, se genera una división que da lugar a otra nueva palabra que se someterá recursivamente al mismo proceso hasta que no quede nada más por seccionar. Estos resultados son concatenados en el orden de procesamiento, es decir, de izquierda a derecha.

Por ejemplo, para el token “Hong”, primero buscaríamos la subpalabra más larga que contenga toda la secuencia de caracteres, sin embargo, al no existir, debemos buscar alguna que lo haga para “Hon”, y como tampoco la hay, se itera con “Ho” hasta llegar a “H”, la cual sí existe, dando como resultado las subpalabras “H” y “##ong”. Ahora para “##ong” sucede lo mismo, así que pasamos desde “##on” hasta “##o” generando las subpalabras “##o” y “##ng”. La tokenización final de la palabra puede ser visualizada en el módulo (12).

Una situación diferente sucede con la palabra “!”, ya que no existe dentro del vocabulario. La implementación de BERT lo que hace es añadir un token especial denominado “[UNK]” para lidiar con aquellas situaciones donde no hay una subpalabra dentro del vocabulario que permita realizar la codificación. Si en alguna de las divisiones para palabras con más de un

```
tokenization = ["Hong", "Tong", "!"]
```

Módulo 11: Tokenización basada en espacios considerando signos de puntuación para la oración “Hong Tong!” en un formato de lista de *strings*.

```
tokenization = ["H", "##o", "##n", "##g"]
```

Módulo 12: Tokenización basada en WordPiece para la palabra “Hong” en un formato de lista de *strings*.

carácter se presenta esta misma situación, se retornará “[UNK]”, pese a que subpalabras anteriores sí hayan existido dentro del vocabulario. Por ejemplo, “hOpe” genera “h” y “##Ope”, pero ni “##Ope”, “##Op” o “##O” existen como tal, así que simplemente falla el proceso. El resultado final para “Hong tong!” queda ilustrado en el módulo (13).

```
tokenization = ["H", "##o", "##n", "##g", "T", "##o", "##n",  
↪ "##g", "[UNK]"]
```

Módulo 13: Tokenización basada en WordPiece para la oración “Hong Tong!” en un formato de lista de *strings*.

Existen algunos detalles con respecto a la implementación que tienen que ver con el tratamiento del relleno y la añadidura de otros tokens especiales para manejar entradas de tipo “pregunta y respuesta” (QA) que serán descritos más adelante.

2.11. Redes neuronales recurrentes

Una de las formas para poder enlazar temporalmente los resultados que genera una red es forzando que, al menos, la entrada de una neurona contenga a su propia salida. Si antes en la arquitectura definida en la sección (2.1) recibíamos un vector para generar una salida, ahora, recibimos una secuencia de vectores $(\mathbf{x}_t)_{t \in \mathcal{T}}$ a lo largo del tiempo t . Este diseño recurre a dos salidas interconectadas: la primera, actúa como una capa oculta que combina el valor de la secuencia en el tiempo t con las salidas generadas en el tiempo $t - 1$. Matemáticamente la formalizamos así:

$$\mathbf{h}_{t+1} = \vec{f}_h(W_x \cdot \mathbf{x}_t + W_h \cdot \mathbf{h}_t + \mathbf{b}_h), \quad (14)$$

donde:

- $W_x \in \mathbb{R}^{r_h \times r_x}$ es la matriz de pesos para la secuencia en el tiempo t . El escalar r_x es la cantidad de características **inherentes a la secuencia**, mientras que r_h la cantidad de unidades ocultas que utilizaremos internamente para describir los resultados que vayamos memorizando.
- $\mathbf{x}_t \in \mathbb{R}^{r_x}$ es la secuencia en el tiempo t .
- $W_h \in \mathbb{R}^{r_h \times r_h}$ es la matriz de pesos para las unidades ocultas.
- $\mathbf{h}_t \in \mathbb{R}^{r_h}$ es el vector de unidades ocultas en el tiempo t . Este término es el que nos permite memorizar los resultados obtenidos en tiempos anteriores a t .
- $\mathbf{b}_h \in \mathbb{R}^{r_h}$ es el vector de sesgos para la capa oculta.
- $\vec{f}_h : \mathbb{R}^{r_h} \rightarrow \mathbb{R}^{r_h}$ es la función de activación vectorial en la capa oculta.

La segunda utiliza la transformación obtenida en (14) para consolidar la salida final materializada como:

$$\mathbf{o}_t = \vec{f}_o(W_o \cdot \mathbf{h}_t + \mathbf{b}_o), \quad (15)$$

donde:

- $W_o \in \mathbb{R}^{1 \times r_h}$ es la matriz de pesos para el vector de unidades ocultas en el tiempo t .
- $\mathbf{b}_o \in \mathbb{R}^{r_h}$ es el vector de sesgos en la capa de salida.
- $\vec{f}_o : \mathbb{R}^{r_h} \rightarrow \mathbb{R}^{r_h}$ es la función de activación vectorial en la capa de salida.

Desde ya, notamos que las matrices W_x y W_o se comparten a través del tiempo, por ende, estas redes resultan ser más compactas en cuanto a la cantidad de parámetros a aprender. Por otro lado, los resultados obtenidos en las secciones (2.3), (2.5) y (2.6) requieren de modificaciones que las hagan dependientes del tiempo [Werbos, 1990, Bird y Polivoda, 2021]. En la literatura, se suelen ilustrar estas redes mediante dos formas: una no desplegada (*folded*) y una desplegada (*unfolded*) [Sherstinsky, 2018]. La diferencia está en que la segunda detalla la secuencia de pasos que sufren los estados ocultos $\mathbf{h}_t$ y las salidas $\mathbf{o}_t$ a través del tiempo t (ver figura (3)).

Uno de los grandes desafíos que nos encontraremos al utilizar esta definición primitiva es que la capacidad de generalización se verá degradada a medida que las secuencias crezcan. El factor más común suele estar relacionado con la **explosión o desvanecimiento de la norma del gradiente** para términos de largo plazo [Bengio *et al.*, 1994]. Durante su cálculo, nos veremos forzados a estimar el valor de la expresión $\partial \mathbf{h}_{t+1} / \partial \mathbf{h}_t$ y se puede demostrar por

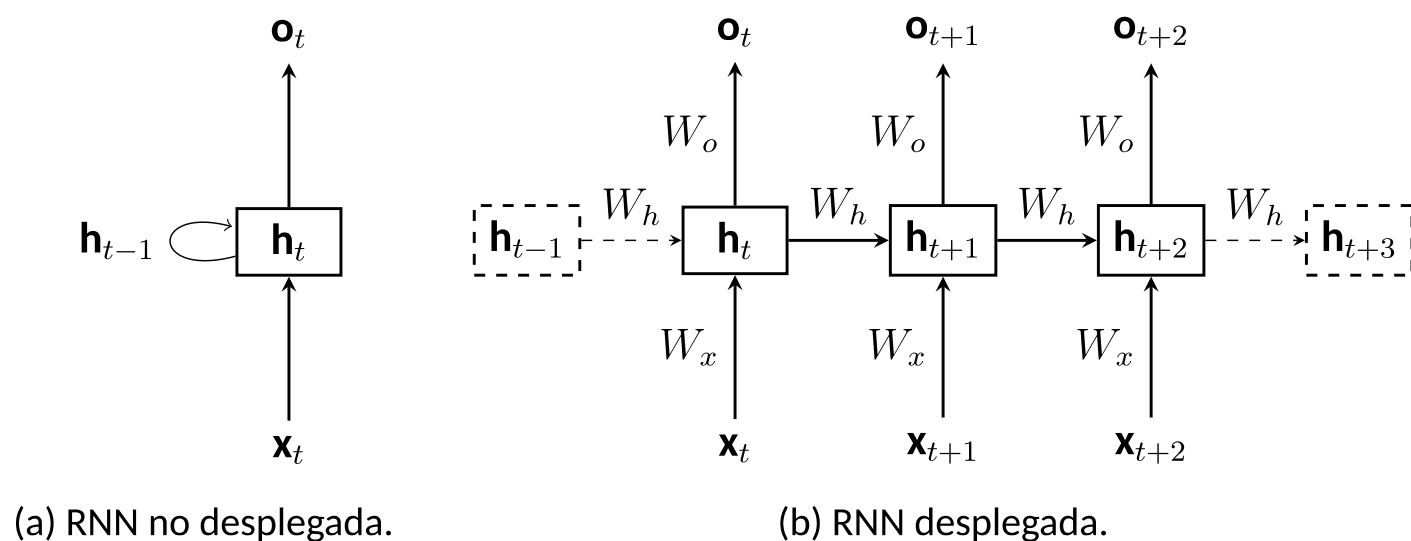


Figura 3: RNN con y sin desplegamiento.

inducción que para todo t , su norma euclídea siempre resulta ser menor que uno, en caso de que nos refiramos al desvanecimiento, o bien, siempre mayor que uno, en caso de que nos refiramos a la explosión [Pascanu *et al.*, 2012]. A raíz de esto, existen distintas propuestas que buscan aplacar estos problemas, refiérase a arquitecturas BRNN (*bidirectional recurrent neural networks*), GRU (*gated recurrent units*) o LSTM (*long short-term memory*) [Schuster y Paliwal, 1997, Cho *et al.*, 2014, Hochreiter y Schmidhuber, 1997].

2.12. LSTM

Dado que el desvanecimiento o explosión del gradiente es un problema inherente a la forma en como definimos la recurrencia en la sección (2.11), resulta necesario reestructurar los cálculos para que podamos aprender eficazmente secuencias de mayor longitud.

El cambio empieza por plantear dos tipos de memoria que trabajarán en conjunto:

- Una de **corto plazo** que será materializada mediante unidades ocultas $\mathbf{h}_t$ a través del tiempo t . Su salida estará en función de tres vectores:
 - La secuencia $\mathbf{x}_t$ en el tiempo t .
 - El estado previo de la memoria de corto plazo $\mathbf{h}_{t-1}$ en el tiempo $t - 1$.
 - El estado previo de la memoria de largo plazo $\mathbf{c}_{t-1}$ en el tiempo $t - 1$.

La forma en cómo se combinan estos tres vectores se produce de la siguiente manera:

1. Se aplica una transformación lineal utilizando alguna combinación de $\mathbf{x}_t$, $\mathbf{h}_{t-1}$ o $\mathbf{c}_{t-1}$.
2. Se aplica la función de activación sigmoide (ver ecuación (6)) en cada uno de los elementos del punto (1). El resultado será algo parecido a un “vector de pesos”.

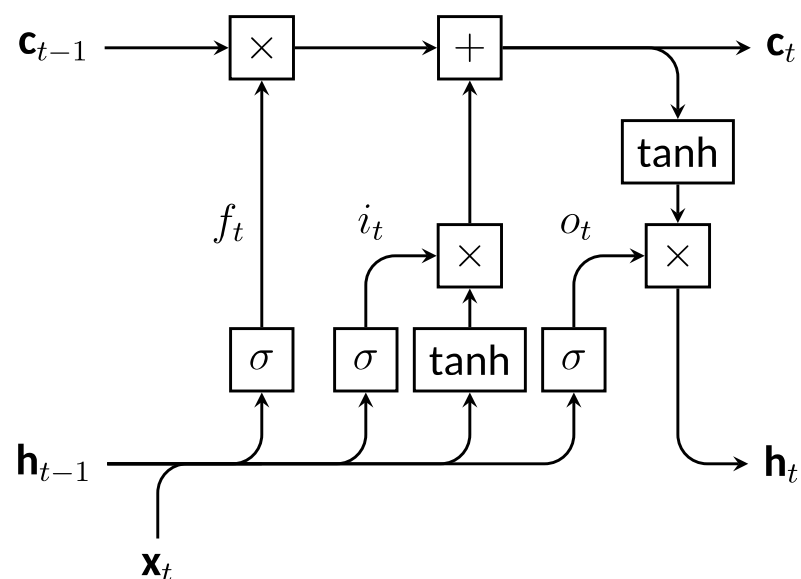


Figura 4: Arquitectura LSTM vista como nodos. Está inspirada en las ilustraciones de Christopher Olah [Olah, 2015]. Todas las intersecciones entre líneas simbolizan una concatenación, mientras las bifurcaciones, una duplicación. El símbolo “ $\times$ ” refiere una multiplicación tipo “Hadamard”, mientras que “ $+$ ”, una suma vectorial. Tanto “ $\tanh$ ” como “ σ ” aluden a las funciones “tangente hiperbólica” y “sigmoide” definidas en las ecuaciones (4) y (6). Tanto f_t , i_t como o_t son los “vectores de pesos” usados en las compuertas de olvido, entrada y salida.

3. Se multiplica elemento a elemento el vector del punto (2) con alguna otra transformación de igual dimensionalidad en la que se busque **filtrar** el aporte de sus componentes. Esto es lo que se conoce como una **compuerta**. De acuerdo a la definición original [Hochreiter y Schmidhuber, 1997], encontramos tres de estas:

- Compuerta de olvido: Decide qué componentes de la memoria de largo plazo contribuirán en la actualización de c_t .
- Compuerta de entrada: Decide qué componentes de la entrada y de la memoria de corto plazo actualizarán c_t .
- Compuerta de salida: Decide qué componentes de la entrada y de la memoria de corto plazo contribuirán en la actualización de h_t .

- Una de **largo plazo** que será materializada mediante celdas de estado c_t a través del tiempo t . Su salida estará en función de los mismos tres vectores mencionados recientemente.

A través de la figura (4) se pueden apreciar resumidamente los cálculos ejecutados dentro de una celda. Por lo demás, este es un modelo que matemáticamente resulta ser complejo tanto en la etapa de propagación hacia adelante como hacia atrás, por lo que enumerar sus detalles técnicos se puede tornar más dificultoso que provechoso, así que solo nos enfocaremos en sus **implicancias**.

Es de amplio saber que esta arquitectura ha resultado ser efectiva en el reconocimiento de patrones para secuencias de gran envergadura [Sak *et al.*, 2014, Greff *et al.*, 2015]. La razón radica en cómo descompone el término $\partial h_{t+1} / \partial h_t$ (mencionado en la sección (2.11)) en una

suma de subgradientes que están en función de las salidas generadas en las compuertas de olvido, entrada y salida [Hochreiter y Schmidhuber, 1997]. En concreto, para que se pudiera presenciar un desvanecimiento del gradiente, se deberían de cumplir (al menos) dos condiciones simultáneamente:

- Que la salida de la compuerta de olvido tienda a cero para todo paso k . Este término está incluido dentro de la suma de subgradientes, pero no está multiplicado por ninguna otra compuerta o variable.
- Que todos los subgradientes tiendan a cero para todo paso k .

2.13. Atención

Tiempo atrás, las redes RNN, LSTM y GRU establecieron el estado del arte en el modelamiento secuencial y en los problemas de transducción [Sutskever *et al.*, 2014, Bahdanau *et al.*, 2014, Cho *et al.*, 2014], sin embargo, la naturaleza secuencial de estos modelos resultó prohibitiva para la paralelización, lo cual es un aspecto crítico a medida que las muestras crecen, ya sea por el nivel de consumo de recursos o de capacidad de generalización. Durante este mismo periodo, los **mecanismos de atención** empezaron a popularizarse a través de distintos trabajos [Bahdanau *et al.*, 2014, Kim *et al.*, 2017, Parikh *et al.*, 2016], ya que lograban enlazar dependencias temporales independientemente del tamaño de las secuencias (un problema que si bien LSTM había conseguido aplacar satisfactoriamente, no resultó ser suficiente en tareas de traducción automatizada).

La atención puede ser entendida como un procedimiento donde dinámicamente se destacan partes de una información. Fue introducido por primera vez en una red Seq2Seq para traducir oraciones desde el inglés al francés [Bahdanau *et al.*, 2014] logrando alcanzar un puntaje BLEU de 21,50 unidades para un entrenamiento basado en secuencias de 30 palabras y 26,75 unidades para un entrenamiento basado en secuencias de 50 palabras, muy por encima de los resultados que una tradicional RNN de tipo codificador-decodificador lograba para ese entonces [Cho *et al.*, 2014]. No obstante, lo realmente fundamental de esta propuesta fue la **estabilidad** en puntajes que proporcionaba el enfoque atencional para secuencias de hasta 60 palabras, pues todas las soluciones clásicas tendían a decaer linealmente a partir de las 20 palabras [Bahdanau *et al.*, 2014].

2.13.1. La propuesta de Bahdanau et al.

La descripción matemática que se brindará a continuación no se corresponde exactamente con la implementación real en el *paper* [Bahdanau *et al.*, 2014] dentro de los siguientes puntos:

1. La forma en cómo se calculan las transformaciones lineales: Notará que, en casos, utilizamos una sola matriz que concatena todos los vectores, cuando en realidad, se utilizan muchas más.
2. La forma en cómo se calculan los estados ocultos: Acá las transformaciones lineales pasan directamente por una función de activación, pero el enfoque optado por los autores consiste en utilizar las salidas de una variante LSTM.
3. El tipo de red trabajada: Empleamos una RNN unidireccional siendo que originalmente se prefiere una bidireccional.

La motivación detrás de estas simplificaciones está en la captura del concepto más que del detalle. Dicho lo anterior, estos son los pasos:

1. Desde el lado del codificador, cada elemento de la secuencia de entrada $(\mathbf{x}_t)_{t \in \mathcal{T}}$ genera una secuencia de representaciones ocultas $(\mathbf{h}_t)_{t \in \mathcal{T}}$. A través del vector de estados ocultos del decodificador $\mathbf{h}'_{t'-1} \in \mathbb{R}^{r_h}$ se genera un **puntaje de alineación** $e_{t,t'} \in \mathbb{R}$ expresado en dos pasos:

- a) Se calculan los puntajes de alineación **no normalizados** mediante:

$$\tilde{e}_{t,t'}(\mathbf{h}_t, \mathbf{h}'_{t'-1}) = \mathbf{v}^T \cdot \tanh(W_h \cdot \mathbf{h}_t + W_{h'} \cdot \mathbf{h}'_{t'-1}),$$

donde:

- $\mathbf{v} \in \mathbb{R}^{r_h}$ es un vector entrenable que permite realizar el producto punto para transformar el resultado en un escalar.
- $W_h \in \mathbb{R}^{r_h \times r_h}$ es una matriz de pesos entrenables para los estados ocultos del codificador.
- $W_{h'} \in \mathbb{R}^{r_h \times r_h}$ es una matriz de pesos entrenables para los estados ocultos del decodificador.

- b) Se **normalizan** todos los $\tilde{e}_{t,t'}$ a través de una función softmax:

$$e_{t,t'} = \frac{\exp(\tilde{e}_{t,t'})}{\sum_{i \in \mathcal{T}} \exp(\tilde{e}_{i,t'})}.$$

El valor de $\mathbf{h}'_0$ es igual a $\mathbf{x}_{|\mathcal{T}|}$ (el último estado oculto de la secuencia).

2. Se estima el **vector de contexto** $\mathbf{c}_{t'} \in \mathbb{R}^{r_h}$ como:

$$\mathbf{c}_{t'} = \sum_{t \in \mathcal{T}} e_{t,t'} \cdot \mathbf{h}_t.$$

Este resultado es lo que genera la atención, ya que cada $e_{t,t'}$ determina qué tan relevante es su par $\mathbf{h}_t$ en la predicción.

3. Del lado del decodificador, se calcula el siguiente estado oculto $\mathbf{h}'_{t'} \in \mathbb{R}^{r_h}$ como:

$$\mathbf{h}'_{t'} = \tanh \left(W_d \cdot [\mathbf{y}_{t'-1}; \mathbf{h}'_{t'-1}; \mathbf{c}_{t'}]^\top + \mathbf{b}_d \right), \quad (16)$$

donde:

- $W_d \in \mathbb{R}^{r_h \times (r_e + 2r_h)}$ es una matriz de pesos.
- $\mathbf{y}_{t'-1} \in \mathbb{R}^{r_e}$ es la palabra predicha en el paso anterior. El escalar r_e es el tamaño del “incrustado” del idioma objetivo.
- $[\mathbf{y}_{t'-1}; \mathbf{h}'_{t'-1}; \mathbf{c}_{t'}] \in \mathbb{R}^{r_e + 2r_h}$ es la concatenación de los vectores $\mathbf{y}_{t'-1}$, $\mathbf{h}'_{t'-1}$ y $\mathbf{c}_{t'}$.
- $\mathbf{b}_d \in \mathbb{R}^{r_h}$ es el vector de sesgo.

El valor de $\mathbf{y}_0$ es igual a la **señal de comienzo**, que es un token especial comúnmente denominado “SOS” (*start of sequence*).

4. Se genera la siguiente palabra $\mathbf{y}_{t'} \in \mathbb{R}^{r_e}$ a través de una función no lineal $g(\cdot)$ que depende de $\mathbf{y}_{t'-1}$, el estado oculto del decodificador $\mathbf{h}'_{t'}$ y del vector de contexto $\mathbf{c}_{t'}$. Cuando $\mathbf{y}_{t'}$ es igual a la **señal de término**, la predicción termina. Aquí también se reserva un token especial usualmente llamado “EOS” (*end of sequence*).

2.13.2. La propuesta de Luong et al.

Tanto esta visión como la descrita en la sección (2.13.1) comparten la misma metodología para la aplicación de la atención en el sentido de, primero, obtener los estados ocultos del decodificador, segundo, los puntajes de alineación, tercero, el contexto, y cuarto, la predicción.

Lo que hace destacable a esta investigación es la incorporación de dos tipos de atención [Luong et al., 2015]:

- Atención global: Considera **todos** los estados ocultos $\mathbf{h}_t$ del codificador para derivar el vector de contexto $\mathbf{c}_{t'}$ en cada estado oculto $\tilde{\mathbf{h}}_{t'}$ de la unidad LSTM. Propone tres tipos de puntajes de alineación:

- De tipo “dot” (punto):

$$\tilde{e}_{t,t'}(\tilde{\mathbf{h}}_{t'}, \mathbf{h}_t) = (\tilde{\mathbf{h}}_{t'})^\top \cdot \mathbf{h}_t.$$

- De tipo “general”:

$$\tilde{e}_{t,t'}(\tilde{\mathbf{h}}_{t'}, \mathbf{h}_t) = (\tilde{\mathbf{h}}_{t'})^\top \cdot W_{\tilde{e}} \cdot \mathbf{h}_t,$$

donde:

- $W_{\tilde{e}} \in \mathbb{R}^{r_h \times r_h}$ es una matriz de pesos entrenable.
- De tipo “concat” (concatenación):

$$\tilde{e}_{t,t'}(\tilde{\mathbf{h}}_{t'}, \mathbf{h}_t) = \mathbf{v}^\top \cdot \tanh \left(W_{\tilde{e}} \cdot [\tilde{\mathbf{h}}_{t'}; \mathbf{h}_t]^\top \right),$$

donde:

- $\mathbf{v} \in \mathbb{R}^{r_h}$ es un vector entrenable.
- $W_{\tilde{e}} \in \mathbb{R}^{r_h \times 2r_h}$ es una matriz de pesos entrenable.
- $[\tilde{\mathbf{h}}_{t'}; \mathbf{h}_t] \in \mathbb{R}^{2r_h}$ es la concatenación de los vectores $\tilde{\mathbf{h}}_{t'}$ y $\mathbf{h}_t$.

Note que en la sección (2.13.1) se utiliza el estado oculto del **decodificador** ($\mathbf{h}'_{t'-1}$), pero acá, en cambio, se utiliza el estado oculto de la **unidad LSTM** ($\tilde{\mathbf{h}}_{t'}$), de esta forma, $\mathbf{h}'_{t'}$ puede ser derivado sin la necesidad de depender de tiempos anteriores.

- **Atención local:** Considera un **subconjunto** de todos los estados ocultos $\mathbf{h}_t$ del codificador para derivar el vector de contexto $\mathbf{c}_{t'}$. Esto se logra definiendo una posición central $p_{t'} \in [0, |\mathcal{T}'|] \subset \mathbb{N}_0$ y extrayendo $d \in \mathbb{N}$ palabras hacia adelante y hacia atrás, de esta manera, $t \in [p_{t'} - d, p_{t'} + d] \subset \mathbb{N}_0$. Si bien d se escoge de manera empírica, $p_{t'}$ se establece a través de dos variantes:

- **Alineación monótona:** Asume que ambos lenguajes son **monótonos** en el sentido de que cada palabra puede ser mapeada con otra bajo un único sentido. Por ejemplo, “the house” (inglés) y “la casa” (español) cumplen esta propiedad, pero “red car” (inglés) con “voiture rouge” (francés), no. En conclusión, tenemos que:

$$p_{t'} = t'.$$

- **Alineación predictiva:** Calcula $p_{t'}$ en función de los valores que tome la codificación $\mathbf{h}_{t'}$:

$$p_{t'} = |\mathcal{T}| \cdot \sigma \left(\mathbf{v}_p^\top \cdot \tanh \left(W_p \cdot \tilde{\mathbf{h}}_{t'}^\top \right) \right),$$

donde:

- $|\mathcal{T}|$ es el largo de la secuencia a traducir.
- $\mathbf{v}_p \in \mathbb{R}^{r_h}$ es un vector entrenable.
- $W_p \in \mathbb{R}^{r_h \times r_h}$ es una matriz de pesos entrenable.

Cualquiera sea el caso, la atención se calcula considerando un nuevo factor:

$$\hat{e}_{t,t'}(\tilde{\mathbf{h}}_{t'}, \mathbf{h}_t) = \tilde{e}_{t,t'}(\tilde{\mathbf{h}}_{t'}, \mathbf{h}_t) \cdot \exp \left(-\frac{(t - p_{t'})^2}{2 \cdot (d/2)^2} \right).$$

Se pretende que esta modificación favorezca a palabras que están cercanas a la posición $p_{t'}$.

Ahora, el estado oculto del decodificador $\mathbf{s}_{t'}$ se expresa como:

$$\mathbf{h}'_{t'} = W_s \cdot [\mathbf{c}_{t'}; \tilde{\mathbf{h}}_{t'}],$$

donde:

- $W_s \in \mathbb{R}^{r_h \times 2r_h}$ es una matriz de pesos entrenable.
- $[\mathbf{c}_{t'}; \tilde{\mathbf{h}}_{t'}] \in \mathbb{R}^{2r_h}$ es la concatenación de los vectores $\mathbf{c}_{t'}$ y $\tilde{\mathbf{h}}_{t'}$.

2.13.3. La propuesta de Vaswani et al.

Hasta el momento, todos los trabajos revisados utilizaron redes RNN para manejar las dependencias temporales de las secuencias. Los altos costos computacionales que involucraron los entrenamientos para textos de tamaños considerables, dieron pie a distintos modelos mejores optimizados, entre ellos, Extended General GPU [Kaiser y Bengio, 2016], ByteNet [Kalchbrenner *et al.*, 2016] y Conv2S [Gehring *et al.*, 2017], los cuales utilizaron **redes convolucionales** para procesar (de manera paralela) las representaciones ocultas de las entradas y salidas. Pese a todo, cuando la distancia entre dos señales altamente relacionadas crecía, la cantidad de operaciones también, a veces, en un formato lineal (caso Conv2S), o a veces, en un formato logarítmico (caso ByteNet). Principalmente, esto es lo que da origen a una alternativa revolucionaria en cuanto a todo lo que, hasta ese entonces, se había estado haciendo.

La idea consistió en **eliminar la recurrencia** y utilizar únicamente mecanismos de atención para trazar las dependencias entre las entradas y salidas. La materialización se da a través de la “atención propia” (*self-attention*) o “intraatención” (*intra-attention*) aplicada exitosamente en distintas tareas tales como compresión lectora, generación de resúmenes o vinculación textual [Cheng *et al.*, 2016, Parikh *et al.*, 2016, Paulus *et al.*, 2017, Lin *et al.*, 2017]. Opera relacionando diferentes palabras de un texto consigo mismo para generar una representación oculta. Presenta varias ventajas que se pueden resumir en que la complejidad computacional por capa se reduce a un número constante, así también el camino para relacionar dependencias de largo plazo, la gran mayoría de las operaciones son paralelizables y la interpretabilidad de los modelos mejora.

Al igual que en las propuestas enunciadas en las secciones (2.13.1) y (2.13.2), también posee un esquema codificador-decodificador [Vaswani *et al.*, 2017]. A continuación, describiremos todos los elementos que componen esta arquitectura.

2.13.3.1. Codificación posicional Al no existir recurrencias ni convoluciones, resulta necesario inyectar artificialmente algún tipo de orden que tenga relación con la posición de los

tókenes. Para este propósito, se utilizan las funciones $\sin(\cdot)$ y $\cos(\cdot)$ de la siguiente manera:

$$\text{PE}(i, j) = \begin{cases} \sin\left(\frac{i}{10000^{2j/\mathcal{L}(\mathbf{x}_i)}}\right) & \text{si } j \text{ es par ,} \\ \cos\left(\frac{i}{10000^{2j/\mathcal{L}(\mathbf{x}_i)}}\right) & \text{si } j \text{ es impar ,} \end{cases}$$

donde:

- $i \in \mathcal{T}$ es la posición del i -ésimo token con respecto a la secuencia de entrada del **codificador**. Si, en cambio, nos referimos a la secuencia de entrada del **decodificador**, tenemos que $i \in \mathcal{T}'$.
- $j \in [1, \mathcal{L}(\mathbf{x}_i)] \subset \mathbb{N}$ es el j -ésimo componente del **vector posicional** de entrada para el codificador o decodificador.

De esta forma, $\text{PE}(i, j)$ se suma a cada una de las secuencias de entrada para el codificador o decodificador.

2.13.3.2. Atención producto-punto escalado De ahora en adelante, cada uno de los elementos de la secuencia forma una “consulta” que busca conocer un “valor” que refleje su relación con el resto de las palabras a través de una “llave”, algo muy similar (conceptualmente hablando) a los diccionarios en Python. Las consultas serán cada uno de los elementos de la secuencia compactados a través de una matriz $Q \in \mathbb{R}^{|\mathcal{T}| \times d_K}$ de vectores columna. Las claves estarán contenidas en una matriz $K \in \mathbb{R}^{|\mathcal{T}| \times d_K}$ de vectores columna y los valores en una matriz $V \in \mathbb{R}^{|\mathcal{T}| \times d_V}$ de vectores columna. La intraatención entonces se calcula por:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^\top}{\sqrt{d_K}}\right) \cdot V.$$

Es importante notar que $d_K \neq \mathcal{L}(\mathbf{x}_i)$, ya que cada uno de los elementos de la secuencia sufre una transformación lineal que expande o contrae el espacio antes de ingresarlos a la función. Esta expresión está inspirada en forma a la propuesta de la sección (2.13.2).

2.13.3.3. Atención producto-punto escalada con enmascaramiento Fuerza a que la atención del i -ésimo elemento en la secuencia dependa únicamente de posiciones anteriores a i . Esto se logra haciendo que el triangular superior de la matriz dentro de la $\text{softmax}(\cdot)$ se rellene con infinitos negativos. Aunque no queda clara la implementación dentro del *paper*, una opción es definir una matriz identidad M en la que su triangular superior sean infinitos negativos y aplicar una multiplicación tipo Hadamard.

$$\text{MaskedAttention}(Q, K, V) = \text{softmax} \left(\frac{Q \cdot K^\top}{\sqrt{d_K}} \odot M \right) \cdot V.$$

2.13.3.4. Cabezal Aplica los mecanismos de intraatención definidos en las secciones (2.13.3.2) y (2.13.3.3). Recibe “consultas” desde una matriz de vectores columna $Q \in \mathbb{R}^{|\mathcal{T}| \times \mathcal{L}(\mathbf{x}_i)}$ que son proyectadas al espacio $W_Q \in \mathbb{R}^{\mathcal{L}(\mathbf{x}_i) \times d_K}$, utiliza las “llaves” de una matriz de vectores columna $K \in \mathbb{R}^{|\mathcal{V}| \times \mathcal{L}(\mathbf{x}_i)}$ que son proyectadas al espacio $W_K \in \mathbb{R}^{\mathcal{L}(\mathbf{x}_i) \times d_K}$ y los “valores” de una matriz de vectores columna $V \in \mathbb{R}^{|\mathcal{V}| \times \mathcal{L}(\mathbf{x}_i)}$ que son proyectados al espacio $W_V \in \mathbb{R}^{\mathcal{L}(\mathbf{x}_i) \times d_V}$. Su definición es la siguiente:

$$\text{head}(Q, K, V) = \text{Attention}(Q \cdot W_Q, K \cdot W_K, V \cdot W_V).$$

Se debe cumplir que $\mathcal{L}(\mathbf{x}_i) = \mathcal{L}(\mathbf{x}_j) \forall (i, j)$. Esta vez, los vectores que contienen las matrices Q y K son los “incrustados” de los elementos de la secuencia $\mathbf{x}_i$ y de las palabras en el vocabulario $\mathcal{V}$.

2.13.3.5. Atención multicabezal Es la aplicación reiterada de la función definida en la sección (2.13.3.4). Reescribiéndola, tenemos que:

$$\text{head}_i(Q, K, V) = \text{Attention}(Q \cdot W_{Q_i}, K \cdot W_{K_i}, V \cdot W_{V_i}),$$

donde:

- $W_{Q_i} \in \mathbb{R}^{\mathcal{L}(\mathbf{x}_i) \times d_K}$ es la matriz de proyección del i -ésimo cabezal para Q .
- $W_{K_i} \in \mathbb{R}^{|\mathcal{V}| \times \mathcal{L}(\mathbf{x}_i)}$ es la matriz de proyección del i -ésimo cabezal para K .
- $W_{V_i} \in \mathbb{R}^{|\mathcal{V}| \times \mathcal{L}(\mathbf{x}_i)}$ es la matriz de proyección del i -ésimo cabezal para V .

Así, la atención multicabezal resulta ser:

$$\text{MultiHead}(Q, K, V) = [\text{head}_1(Q, K, V); \dots; \text{head}_h(Q, K, V)] \cdot W_O,$$

donde:

- $[\text{head}_1(Q, K, V); \dots; \text{head}_n(Q, K, V)] \in \mathbb{R}^{|\mathcal{T}| \times (h \cdot d_V)}$ es la concatenación por columnas de los h cabezales.

- $W_O \in \mathbb{R}^{(h \cdot d_V) \times \mathcal{L}(\mathbf{x}_i)}$ es una matriz entrenable.

Cuando se utiliza específicamente la atención definida en la sección (2.13.3.3), se genera una **atención multicabezal enmascarada** [Vaswani *et al.*, 2017].

2.13.3.6. Conexiones residuales Es cuando la salida de una capa es sumada con su entrada, y luego, normalizada. Tanto la entrada como la salida deben ser vectores de igual dimensionalidad. Aunque los autores no entran en detalles de por qué las utilizan, una posible justificación se basaría en poder acelerar los tiempos de convergencia [He *et al.*, 2015].

2.13.3.7. Codificador Define una serie de capas idénticas, cada una, con dos subcapas. La primera implementa la atención multicabezal descrita en la sección (2.13.3.5), y la segunda, una FNN tradicional como la descrita en la sección (2.1). Cada una de las subcapas genera un vector de salida de dimensionalidad $\mathcal{L}(x_i)$ y utilizan conexiones residuales como las descritas en la sección (2.13.3.6).

2.13.3.8. Decodificador También define una serie de capas idénticas, esta vez, con tres subcapas. La primera y la segunda, implementan la atención multicabezal con y sin enmascaramiento descritas en la sección (2.13.3.5), mientras que la tercera, implementa una FNN tradicional como la descrita en la sección (2.1). Nuevamente, cada una de las subcapas genera un vector de salida de dimensionalidad $\mathcal{L}(x_i)$ y utilizan conexiones residuales como las descritas en la sección (2.13.3.6).

2.14. BERT

Las secciones (2.13.1), (2.13.2) y (2.13.3) comparten transversalmente dos aspectos:

1. Se encargan de generar texto.
2. Predicen las siguientes palabras analizando las entradas de izquierda a derecha.

Dado (1), muchas investigaciones empezaron a enfocarse más en **modelar lenguajes** que modelar tareas porque eso les permitía transferir el conocimiento y extrapolar sus resultados a otras áreas [Dai y Le, 2015, Howard y Ruder, 2018]. Algunos de los trabajos más conocidos que establecieron el estado del arte fueron ELMo y OpenAI GPT, ya que se presentaron como modelos **preentrenados** y de **propósito general** [Radford y Narasimhan, 2018, Peters *et al.*, 2018].

El problema, sin embargo, es que las técnicas empleadas para ese entonces usaban la estrategia (2) y restringían el poder de las representaciones aprendidas [Devlin *et al.*, 2018], sobre todo, para tareas donde se necesitaban contextos bidireccionales, véase, preguntas y respuestas (QA) o resumen de textos (y derivados). A raíz de lo anterior, el equipo de Google AI propone un nuevo modelo pre-entrenado de propósito general llamado BERT que logra elevar (y con suma diferencia) el estado del arte en once tareas NLP.

2.14.1. Arquitectura

Aunque está basada en la propuesta de la sección (2.13.3), solo utiliza el codificador, ya que el cálculo de la atención no tiene una orientación clara al tratarse de transformaciones lineales sin un enmascaramiento. El vocabulario es generado aplicando el algoritmo WordPiece descrito en la sección (2.10.4.1). Las entradas pueden ser pares de sentencias (para casos como preguntas y respuestas), o bien, sentencias aisladas (para casos como clasificaciones) que se procesan con el algoritmo descrito en la sección (2.10.4.2), esta vez, sujeto a unos pequeños cambios:

- El primer token será “[CLS]”.
- Entre el inicio de la primera sentencia y el comienzo de la segunda, se interpondrá el token “[SEP]”.
- El último token será “[SEP]”.
- Todos los rellenos se simbolizarán con el token “[PAD]”.

A esta secuencia de tokens, se le suman otros dos: uno, que permita diferenciar los tokens de la primera sentencia y la segunda, y otro, que permita inyectar algún tipo de precedencia entre tokens (tal y como vimos en la sección (2.13.3.1)). Haremos abuso de la notación para indicar que $\mathbf{x} \in \mathbb{R}^{\mathcal{L}(\mathbf{x})}$ es el resultado de este proceso y que puede ser descompuesto como:

$$\mathbf{x} = (\mathbf{x}_{[\text{CLS}]}, \underbrace{\mathbf{x}_1, \dots, \mathbf{x}_{[\text{SEP}]-1}}_{\text{Primera sentencia}}, \mathbf{x}_{[\text{SEP}]}, \underbrace{\mathbf{x}_{[\text{SEP}]+1}, \dots, \mathbf{x}_{[\text{SEP}]-1}}_{\text{Segunda sentencia}}, \mathbf{x}_{[\text{SEP}]}) . \quad (17)$$

La salida de esta red será una secuencia $\mathbf{y} \in \mathbb{R}^{\mathcal{L}(\mathbf{x})}$ que puede ser descompuesta de la misma forma que la expresión (17). Cómo se interprete dependerá de la tarea a desempeñar:

- Si se busca clasificar un par de sentencias, o bien una sentencia aislada, se utilizará $\mathbf{y}_{[\text{CLS}]}$ como un “incrustado” contextual.
- Si se busca responder una pregunta, se utilizará la secuencia $(\mathbf{y}_{[\text{SEP}]+1}, \dots, \mathbf{y}_{[\text{SEP}]-1})$.

- Si se busca clasificar (en palabras) una sentencia aislada, se utilizará la secuencia $(\mathbf{y}_1, \dots, \mathbf{y}_{[\text{SEP}]-1})$.

Originalmente, se proponen dos arquitecturas: una llamada **BERT Base** en la que la cantidad de capas idénticas del codificador es 12, la cantidad de cabezales es 12 y la dimensionalidad de las subcapas es 768; otra llamada **BERT Largo** en la que la cantidad de capas idénticas del codificador es 24, la cantidad de cabezales es 16 y la dimensionalidad de las subcapas es 1024.

2.14.2. Preentrenamiento

Se desarrolla a través de dos tareas no supervisadas:

- Modelado de lenguaje enmascarado (MLM): Busca enmascarar un porcentaje de los tokens de entrada y trata de predecirlos en función de los otros restantes. En números, dicha proporción asciende al 15 % de tal forma que:
 - Un 80 % de ese 15 % se reemplaza por el token especial “[MASK]”. Por ejemplo, la oración “my dog is hairy” quedaría como “my dog is [MASK]”.
 - Un 10 % de ese 15 % se reemplaza por otro token aleatorio. Por ejemplo, la oración “my dog is hairy” quedaría como “my dog is apple”
 - Un 10 % de ese 15 % no se les hace nada. Por ejemplo, la oración “my dog is hairy” quedaría como “my dog is hairy”.

La función de pérdida para esta tarea es una entropía cruzada que únicamente considera las predicciones de los tokens enmascarados.

- Predicción de la siguiente oración (NSP): Dado un par de sentencias A y B, trata de predecir si B viene después de A (recordemos que se distinguen por el token especial “[SEP]”). Durante el entrenamiento, se cumple que:
 - Un 50 % de las veces, la segunda oración viene después de la primera. La forma de categorizarlas es a través de la clase especial “IsNext”. Por ejemplo, si la primera sentencia es “the man went to the store” y la segunda “he bought a gallon of milk”, una alternativa sería la entrada “[CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]” con la clase “IsNext”.
 - Un 50 % de las veces, la segunda oración es completamente aleatoria. La forma de categorizarlas es a través de la clase especial “NotNext”. Por ejemplo, si la primera sentencia es “the man went to the store” y la segunda “he bought a gallon of milk”, una alternativa sería la entrada “[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight #less birds [SEP]” con la clase “NotNext”.

La función de pérdida para esta tarea es una entropía cruzada binaria.

Los errores de ambas tareas se suman para conformar una función de pérdida general que logre balancear el desempeño para MLM y NSP.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

Desarticularemos el problema de la detección de *grooming* en dos fases:

1. Generación de un modelo capaz de reconocer si un mensaje presenta características de *grooming* usando BERT.
2. Aplicación del modelo anterior sobre cada uno de los mensajes de una conversación para determinar si, en definitiva, un diálogo presenta características de *grooming*.

3.1. Primera fase

Como cada conversación consta de múltiples mensajes, agruparemos estos últimos en lotes de tamaño constante mediante el carácter “salto de línea”, ya que resulta inocuo en el proceso de tokenización. La idea es poder capturar el mayor contexto de lo que se está hablando. En caso de que no se puedan formar grupos perfectos, los mensajes no serán desechados, sino más bien asociados a un nuevo lote de tamaño variable. A continuación, afinaremos BERT para “clasificación de secuencias” utilizando las muestras anteriores¹⁰.

Terminado este proceso, los datos de entrenamiento se someterán una vez más a BERT, aunque en esta ocasión, para extraer únicamente los “incrustados” y no las predicciones, ya que teóricamente contendrán información suficiente para resumir los mensajes y determinar si son sospechosos o no.

Para poder entender de manera acabada el planteamiento, revisemos mediante un ejemplo el flujo de procesamiento desde la primera fase hacia la segunda fase (ver figura (5)). En él, encontramos tres conversaciones: la primera con tres mensajes (c_{00} , c_{01} y c_{02}), la segunda con dos mensajes (c_{10} y c_{11}) y la tercera con cuatro mensajes (c_{20} , c_{21} , c_{22} y c_{23}). Supongamos que se utiliza un agrupamiento de tamaño 2. Dado que en la primera conversación no es posible formar grupos exactos, se genera un remanente que contiene un único elemento: c_{02} . Todos los mensajes dentro de un grupo se concatenan de izquierda a derecha (simbolizado a través de la operación “+”).

De la primera conversación, se generan los mensajes concatenados c'_{00} y c'_{01} ; de la segunda conversación, únicamente c'_{10} y de la tercera conversación, c'_{20} y c'_{21} . Estos elementos son procesados a través de BERT (previamente afinado) y se extraen únicamente sus *embeddings*

¹⁰Como esta arquitectura modela lenguajes, no tiene una salida específica, por ende, es necesario instanciar clases de esta que agregan capas al final de la red para propósitos específicos. Entre las más conocidas encontramos **múltiples elecciones** (usado en tareas SWAG), **preguntas y respuestas** (usada en tareas SQuAD) y **clasificación de secuencias** (usada en tareas GLUE).

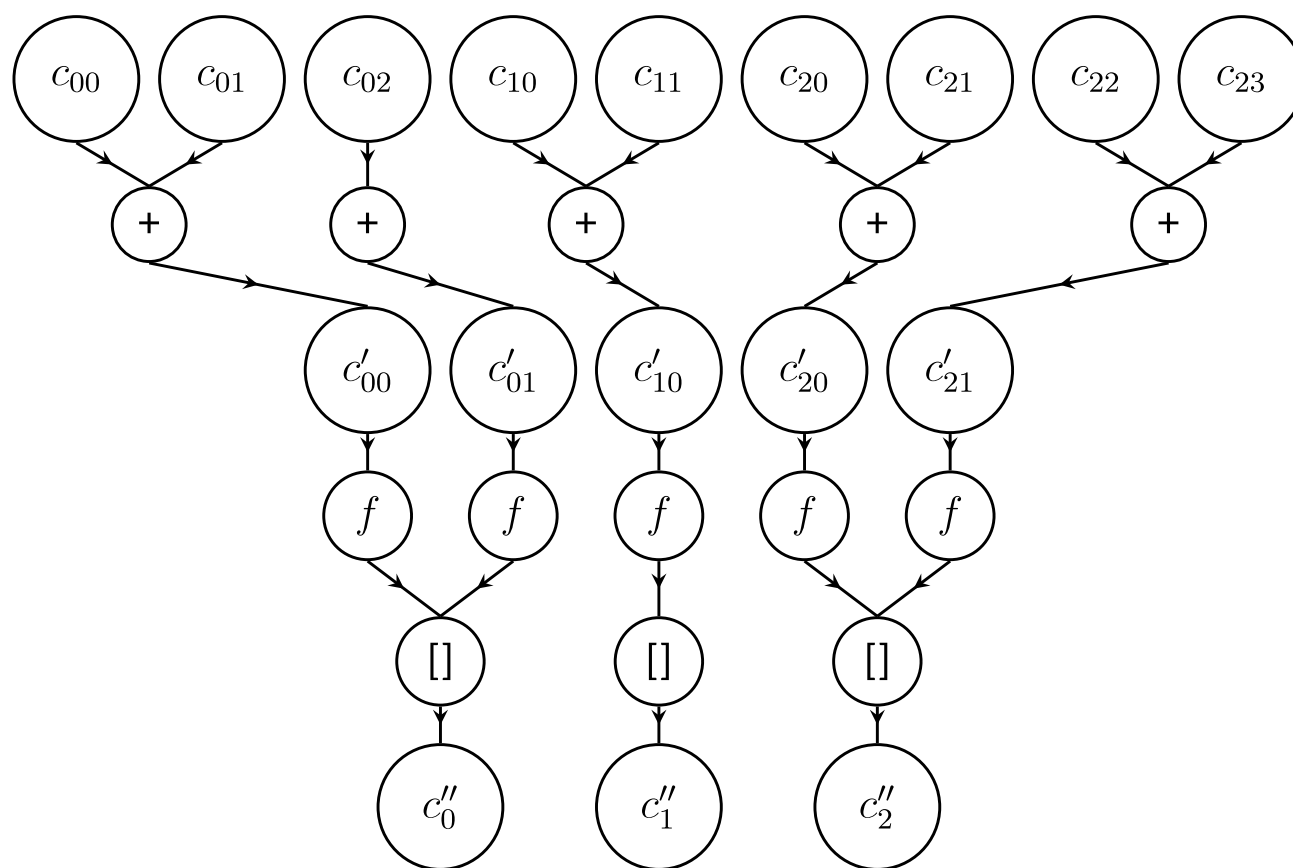


Figura 5: Flujo de procesamiento de conversaciones desde la primera fase hacia la segunda fase. Las constantes c_{ij} y c'_{ij} corresponden al j -ésimo mensaje de la i -ésima conversación, mientras que c''_i a la i -ésima secuencia de “incrustados” y los símbolos “+”, “ f ” y “[]” las operaciones de concatenación de mensajes, extracción de “incrustados” y apilamiento de elementos en una lista.

(simbolizado a través de la operación “ f ”). Finalmente, estos resultados son apilados a una lista para formar las “secuencias de incrustados” (simbolizado a través de la operación “[]”).

3.2. Segunda fase

Considerando que la cantidad de mensajes en una conversación puede abarcar grandes magnitudes, utilizar BERT nativamente no es óptimo porque, mientras más elementos captamos, mayor es el tamaño que las tokenizaciones (o vectores) tienen, lo cual se transforma inmediatamente en un mayor consumo de memoria RAM y GPU. Si a lo anterior le sumamos el hecho de que para agilizar los entrenamientos se deben de agrupar en lotes, el problema se exagera terminando (muchas veces) en errores OOM.

La solución a lo anterior se basa en utilizar una capa LSTM. Se espera que aprenda a reconocer los patrones típicos de una conversación normal y sospechosa a lo largo del tiempo materializado a través de los “incrustados” calculados previamente. Estas salidas serán procesadas por: una FNN para proyectarlas en un nuevo espacio, una capa de *dropout* que prevenga sobreajustes y una capa de clasificación binaria.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

El entrenamiento de cada uno de los modelos fue realizado a través de la plataforma Kaggle, una empresa subsidiaria de Google LLC. Es una comunidad en línea de científicos de datos y profesionales del aprendizaje automático que, entre sus servicios, ofrece un banco de trabajo basado en la nube materializado a través de instancias CPU, GPU y TPU. Sus usos están limitados mediante una cantidad fija de horas por semana. La instancia utilizada en esta memoria se apoda como “GPU P100”, la cual consta de una arquitectura x64 sobre una CPU modelo Intel(R) Xeon(R) CPU @ 2.00 GHz con un total de 13 GB de RAM y una GPU modelo NVIDIA Tesla P100.

4.1. Primera fase

El conjunto de datos utilizado fue “PAN2012”. En él encontramos dos directorios: uno para entrenamiento y otro para pruebas. En el de entrenamiento existen dos archivos de interés: uno, que enlista las conversaciones en un formato XML, y otro, que enumera todos los identificadores de los autores que, se sabe, son depredadores sexuales. En el de pruebas, estos dos archivos también están presentes.

El procesamiento de los datos fue el siguiente: se extrajeron todos los mensajes de cada conversación, se concatenaron en grupos de 24 mediante un salto de línea y se etiquetaron de acuerdo a la siguiente regla: si los mensajes provienen de un autor presente en el listado de depredadores sexuales, automáticamente la concatenación se etiqueta como “sospechosa”, en caso contrario, simplemente como “no sospechosa”. En fases preliminares se utilizaron otros tamaños de grupo tales como 2, 4, 8, 16, 28 y 32, pero ninguno obtuvo resultados óptimos. Las clases se balancearon de forma que la cantidad de mensajes “sospechosos” versus “no sospechosos” fuera la misma. Del conjunto de entrenamiento se reservó un 10 % para propósitos de validación.

El tokenizador (*tokenizer*) fue configurado para generar secuencias de 128 elementos truncados con rellenos a la derecha. El lector podría pensar que es irrelevante la dirección optada, sin embargo, tiene efectos a nivel de predicciones porque los “incrustados” están sujetos a la posición absoluta del token con respecto a la entrada (para más detalles, revisar la sección (2.13.3.1)). Se utilizó el optimizador AdamW con $\beta_1 = 0,9$, $\beta_2 = 0,999$ y $\lambda = 0,01$ (ver sección (5.2) para más detalles sobre los parámetros). Se programó la tasa de aprendizaje para que tuviera un “calentamiento” lineal durante el primer 10 % de los pasos y un decaimiento exponencial con una tasa $\beta = 0,97$ por cada mitad de los pasos que tomaba un *epoch* (ver sección (5.3) para más detalles sobre los parámetros).

Se generó un *notebook* cuyas entradas fueron dos: la tasa de aprendizaje inicial y el número

LR	EP	ET	ACC	LS	REC	PRE	FS
5e-5	1	8:35	96,15 %	0,1597	96,69 %	95,66 %	96,17 %
	2	10:50	94,37 %	0,1701	91,71 %	96,86 %	94,21 %
	3	12:46	94,88 %	0,1551	92,30 %	97,32 %	94,75 %
	4	13:56	96,37 %	0,1214	95,76 %	96,94 %	96,35 %
3e-5	1	7:44	95,41 %	0,1916	93,67 %	97,04 %	95,33 %
	2	9:44	95,30 %	0,1563	92,82 %	97,66 %	95,18 %
	3	12:46	95,26 %	0,1480	92,85 %	97,56 %	95,14 %
	4	13:56	95,96 %	0,1231	95,38 %	96,49 %	95,94 %
2e-5	1	7:52	95,52 %	0,1913	94,53 %	96,44 %	95,48 %
	2	9:44	95,12 %	0,1668	92,37 %	97,76 %	94,98 %
	3	11:52	96,25 %	0,1295	96,13 %	96,36 %	96,25 %
	4	14:20	93,42 %	0,1903	89,01 %	97,63 %	93,12 %

Tabla 1: Tiempos de ejecución y métricas para cada tasa de aprendizaje y *epoch* en la primera fase. La columna LR simboliza la tasa de aprendizaje; EP el número de *epochs*; ET el tiempo de ejecución en un formato “minutos:segundos”; ACC la exactitud; LS la pérdida; REC la recuperación; PRE la precisión; FS el Valor F.

de *epochs*. La salida fue un modelo Keras. En todas las instancias, el tamaño de los lotes trabajado fue de 32 (mayores valores implicaron errores OOM). Las tasas de aprendizaje inicial probadas fueron 5e-5, 3e-5 y 2e-5, mientras que los números de *epochs* 1, 2, 3 y 4. En total, se ejecutaron 12 experimentos. El tiempo neto rondó las 2 horas con 16 minutos. En promedio, las instancias de 1 *epoch* tomaron 8 minutos con 6 segundos; las de 2 *epochs*: 10 minutos con 5 segundos; las de 3 *epochs*: 12 minutos con 28 segundos; la de 4 *epochs*: 14 minutos con 33 segundos. En cada una de ellas, se obtuvieron las siguientes métricas: exactitud (ver ecuación (22)), pérdida (ver ecuación (23)), recuperación (ver ecuación (24)), precisión (ver ecuación (25)) y Valor F (ver ecuación (26)). Sus resultados se pueden encontrar a través de la tabla (1).

Las mejores combinaciones se dieron entre una tasa de aprendizaje 5e-5 con 1 y 4 *epochs*, y también, entre una tasa de aprendizaje 2e-5 con 3 *epochs*. Objetivamente, los tres modelos presentan métricas muy cercanas entre sí y cumplen las expectativas de desempeño, sin embargo, se prefiere la combinación entre una tasa de aprendizaje 5e-5 con 4 *epochs*, ya que cuenta con un Valor F relativamente superior y una pérdida significativamente inferior al resto. Por otra parte, es importante destacar que existe una leve tendencia a reducir los falsos positivos por sobre los negativos en un 1,18 % debido a una mejor precisión, lo cual podría ayudar a focalizar de manera más óptima los recursos utilizados en el análisis de los perfiles.

Para esta primera fase, los resultados nos demuestran que el modelo es capaz de entender la diferencia entre un mensaje con y sin características de *grooming*, por ende, los “incrustados” que genera están sesgados de forma que proporcionan un resumen binario suficiente. Sin embargo, cuando nos ponemos a analizar el 3.69 % de los mensajes que no fueron clasi-

ficados correctamente, nos encontramos con los siguientes casos:

- **Las clasificaciones son aparentemente acertivas (dentro del poco contexto que brindan los mensajes), pero las etiquetas no.** Resulta interesante saber que muchas de las conversaciones son, en efecto, falsos positivos y negativos intencionados por parte de los autores [Inches y Crestani, 2012], y pese a ello, los resultados dan a entender que el modelo es capaz de reconocerlos y descartarlos, incluso, no teniendo el contexto completo de la conversación (producto del agrupamiento). Basándonos en esto, es esperable que la exactitud nunca alcance valores cercanos al 98 % debido a la presencia de ruido. A continuación algunos ejemplos:
 - Su etiqueta real es “sospechoso” y la predicción fue “no sospechoso” con una probabilidad del 78,80 %.
 - Of course I am not mad at you! Why would I be? (¿Por supuesto que no estoy molesto contigo! ¿Por qué lo estaría?)
 - Su etiqueta real es “sospechoso” y la predicción fue “no sospechoso” con una probabilidad del 96,50 %.
 - bibiddi bobidi boo
 - something something blah blah blah from cinderella (algo algo “blah blah blah” de la Cenicienta)
 - Su etiqueta real es “sospechoso” y la predicción fue “no sospechoso” con una probabilidad del 87,11 %.
 - hello zoe blue (hola “zoe blue”)
 - how are you (quién eres)
 - you’re a poet? (¿eres un poeta?)
 - im sure you are a very good poet (estoy seguro que eres un buen poeta)
 - whered you get your nickname (de dónde conseguiste ese apodo)
 - Su etiqueta real es “sospechoso” y la predicción fue “no sospechoso” con una probabilidad del 62,79 %.
 - but its not to good to do it cuz you can get virusis on your comuter (aunque no es tan bueno hacerlo, ya que puedes conseguir virus en tu computadora)
 - oh thats bad (oh... eso es malo)
 - yep (sí)
 - Su etiqueta real es “sospechoso” y la predicción fue “no sospechoso” con una probabilidad del 74,31 %.
 - hey (oye)
 - hey (oye)
 - hey (oye)
 - i gave u the wrong # the other day (te di un número equivocado el otro día)

- **Los mensajes con longitudes largas tienden a desempeñarse peor.** Una de las razones de esta disociación es que una conversación pasa por distintos tópicos que no necesariamente se mantienen en el tiempo. Una conversación puede contener elementos de *grooming* muy focalizados que no necesariamente son capturados en todos los grupos de mensajes (pese a que la etiqueta diga lo contrario). Por otra parte, este modelo se centra más en capturar las relaciones entre palabras y no entre líneas, lo cual podría justificar el “por qué” muchos mensajes altamente sugestivos son “obviados” en puntos concretos de la conversación. Un tercer aspecto es que las tokenizaciones están limitadas a 128 elementos como máximo, lo cual podría significar que muchos de estos (principalmente localizados al final una conversación) sean simplemente ignorados [Beltagy *et al.*, 2020]. A continuación algunos ejemplos:
 - Su etiqueta real es “sospechoso” y la predicción fue “no sospechoso” con una probabilidad del 84,19 %.
 - ok....
 - what greenday songs do you like best? (¿qué canciones de Green Day te gustan más?)
 - american idiot
 - ok..how tall are you? (bien... ¿Cuánto mides?)
 - 5'1
 - nice (buena)
 - you seem like a very sweet girl (pareces una niña muy agradable)
 - i'm more your parent's age...is that ok with you? (soy más de la edad de tus padres... ¿Está bien eso?)
 - sure (claro)
 - ok...are they cool people? (bien... ¿Son buenas personas?)
 - i'm very open...so you can ask or talk to me about anything, ok? (yo soy superabierto... así que puedes preguntar o hablarme cualquier cosa, ¿vale?)
 - ok
 - can i add you to my list? (¿puedo agregarte a mis contactos?)
 - ok
 - do you watch cams on line? (¿tú ves “cámaras” en vivo? — con “cámaras” se refiere a personas que se graban (generalmente) en actos sexuales en vivo a través de internet)
 - i havent (no lo he hecho)
 - ok...they can be pretty sexy (ok... pueden ser superatractivas)
 - ok
 - if you like that sort of thing (si es que te gustan ese tipo de cosas)
 - yea (claro)
 - i have to get going (tengo que irme)

- ok
- what's your name/ (cuál es tu nombre)
- ame
- Su etiqueta real es “sospechoso” y la predicción fue “no sospechoso” con una probabilidad del 69,57 %.
 - what's on your mind? (¿qué tienes en mente?)
 - hmm
 - i can talk about whatever u want (puedo hablar de lo que quieras)
 - what's on your mind? (¿qué tienes en mente?)
 - hmm whens ur bday (¿cuándo es tu cumpleaños?)
 - soon (pronto)
 - am i suppose to guess? (¿se supone que debo adivinar?)
 - haha
 - hahahaha
 - if u want to (si así quieres)
 - tomorrow (mañana)
 - oh no
 - not that soon (no tan pronto)
 - the next day (pasado mañana)
 - i'll give u a clue (te daré una pista)
 - it's on a sunday (es un sábado)
 - in oct. (en octubre)
 - in 2 weeks (en dos semanas)
 - oct 12? (octubre 12)
 - mmhmm
 - haha cool (jaja buena)
 - yeah (sí)
 - i'll be even older:-SS (estaré mucho más viejo)
 - ya bdays make u older (sí, los cumpleaños te hacen más viejo)
- Su etiqueta real es “sospechoso” y la predicción fue “no sospechoso” con una probabilidad del 80,49 %.
 - hello gorgeous (hola precioso)
 - sup? (¿sí?)
 - asl? (edad/sexo/lugar)
 - 21 m ohio (21 años, hombre, Ohio)
 - you (tú)
 - 15 f oh (15 años, mujer, Ohio)
 - do you like older guys (te gustan los hombres mayores)

- yeah if they're cute. (sí, siempre que sean bonitos)
- you got a pic? (¿tienes alguna foto?)
- yeah (claro)
- wht do you think (qué piensas)
- cute (linda)
- :)
- would you like to get me naked (¿te gustaría desnudarme?)
- you mean for real? (¿hablas en serio?)
- yea (claro)
- yeah it could be fun. (sí, podría ser divertido)
- what would you do if you got me naked (qué harías si me tuvieras desnudo)
- stare at you lol....i don't know what would you want me to do? (mirarte fijamente lol... no sé qué te gustaría que haga)
- suck or ride me (chuparm... o montarte encima de mí)
- yeah (claro)
- would you ride or suck me (¿te gustaría montarte encima de mí o chuparm...?)
- yeah (claro)
- you a virgin (eres virgen)

4.2. Segunda fase

Se utiliza el mismo conjunto de datos de la primera fase, por lo que omitiremos su descripción. El preprocesamiento de los datos fue el siguiente: se extrajeron todos los mensajes de cada conversación, se concatenaron nuevamente en grupos de 24 mediante un salto de línea bajo el propósito de mantener una consistencia con el formato de las entradas de la mejor red obtenida durante la primera fase, se agregaron (en orden de aparición) a una lista y se etiquetaron bajo las mismas reglas descritas que en la primera fase. Con este formato, en vez de tener mensajes agrupados (pero aislados entre sí), tenemos una conversación completa. Las clases se balancearon de forma que la cantidad de mensajes “sospechosos” versus “no sospechosos” fuera la misma.

La implementación de BERT en la librería Hugging Face genera nativamente salidas para la pérdida, los “logits” (logaritmos de las predicciones), los estados ocultos y las atenciones (de cada cabezal y capa oculta), pero en ningún caso, los “incrustados” de los tókenes “[CLS]”. Para lograrlo, se extrajo la salida intermedia que venía después de la capa de *dropout* y antes de la FNN (que es donde justamente se realizan las clasificaciones). Esta “nueva red” se aplicó en cada uno de los mensajes agrupados de una conversación, resultando en una secuencia de vectores de 768 dimensiones. Como cada secuencia tiene una longitud diferente de mensajes agrupados, se utilizó un relleno de vectores de ceros de igual dimensionalidad

Hiperparámetro	Rango	Muestreo	Paso
Tasa de aprendizaje (SGD)	0,01 $\rightarrow$ 0,1	logarítmico	-
Unidades (LSTM)	32 $\rightarrow$ 128	lineal	32
Unidades (FNN)	768 $\rightarrow$ 1024	lineal	32
<i>Dropout</i>	0,1 $\rightarrow$ 0,6	lineal	0,1

Tabla 2: Rango de valores de los hiperparámetros para la segunda fase. El muestreo lineal secuencia los números por la ecuación (19) y el logarítmico por la ecuación (20). Los pasos omitidos se muestrean aleatoriamente mediante una distribución uniforme con mínimos y máximos iguales a los del rango.

que los “incrustados”. Inicialmente, se intentó unificar el tamaño de las secuencias con la de mayor longitud dentro del conjunto de datos, pero rápidamente se descartó, ya que generó un aumento en la memoria que se tradujo en mayores tiempos de entrenamiento. La alternativa optada consistió en reducir el relleno a nivel de lotes. En otras palabras, todas las secuencias de un (valga la redundancia) mismo lote, tienen la misma cantidad de “pasos”. Tanto para la hiperafinación como entrenamiento, se utilizaron lotes de tamaño 32 (mayores valores implicaron errores OOM). El tokenizador no sufrió cambios en su configuración. Toda esta cadena de procesamiento se ejecutó en un *notebook* aparte con el fin de optimizar los tiempos de ejecución, ya que sus salidas se reutilizaron en pasos posteriores. En total, tomó alrededor de 29 minutos con 58 segundos.

En la etapa de hiperafinación, se utilizó el método de búsqueda aleatoria con el fin de lograr una mayor diversificación en la exploración de los hiperparámetros. Los elementos considerados fueron los siguientes: la tasa de aprendizaje del optimizador SGD, la cantidad de unidades de la capa LSTM, la cantidad de unidades de la capa FNN y la tasa de *dropout* antes de la capa de clasificación. Los rangos de valores quedan descritos en la tabla (2). La métrica a optimizar fue la pérdida del conjunto de validación. Dichos datos correspondieron a un 10 % del 40 % del entrenamiento. En total, se ejecutaron 50 *trials*. Por cada modelo, la cantidad de *epochs* fue 5. Se reprodujo esta búsqueda dos veces utilizando semillas diferentes. En la primera, el tiempo rondó los 16 minutos con 13 segundos y se obtuvo una pérdida del 0,0793, mientras que en la segunda, el tiempo rondó los 15 minutos con 39 segundos y se obtuvo una pérdida del 0,1973. Los valores encontrados para los hiperparámetros se pueden apreciar en la tabla (3). En todos los casos, la tendencia fue tener tasas de aprendizaje con órdenes de magnitud 10^{-2} , pocas unidades LSTM y un alto *dropout*. En experimentos preliminares, se intentó incorporar un *dropout* adicional en la capa LSTM, pero siempre tendían a empeorar el desempeño.

En la etapa de entrenamiento, se materializó un modelo por cada uno de los hiperparámetros descritos en la tabla (3). Se utilizó un único *callback* cuya función fue la de respaldar los modelos luego de cada *epoch*. El optimizador SGD se configuró para minimizar la pérdida (aunque se trató de experimentar mediante la precisión, los modelos resultantes tendían a sobreajustarse a los datos y a tener poca capacidad de generalización). En la práctica notamos que 7 *epochs* fueron suficientes para entrenar a los modelos (mayores valores no generaron

Hiperparámetro	Repetición	Valor
Tasa de aprendizaje (SGD)	1	0,0990
	2	0,0314
Unidades (LSTM)	1	64
	2	96
Unidades (FNN)	1	768
	2	1024
<i>Dropout</i>	1	0,5
	2	0,5

Tabla 3: Valores de los hiperparámetros para la segunda fase. Se organizan de acuerdo al número de repetición.

ninguna mejora, pues se traducían, tarde o temprano, en mínimos locales que arrojaban las mismas métricas cíclicamente). El tiempo neto para el primer experimento rondó los 21 minutos con 45 segundos, mientras que para el segundo, los 21 minutos con 11 segundos. En cada uno de ellos, se obtuvieron las siguientes métricas: exactitud (ver ecuación (22)), pérdida (ver ecuación (23)), recuperación (ver ecuación (24)), precisión (ver ecuación (25)) y Valor F (ver ecuación (26)). Sus resultados se pueden encontrar a través de la tabla (4).

La mejor combinación se dio en la repetición 1 con 7 *epochs* alcanzando un 94,38 % de exactitud, un 93,68 % de recuperación, un 95,01 % de precisión y un 94,34 % de Valor F. En promedio, se evidenció una reducción general en la capacidad de generalización de un 2 % en comparación a los resultados obtenidos durante la primera fase. Pese a esto, la proporción entre las métricas se mantuvo, lo cual resulta ser beneficioso bajo la siguiente premisa: entendiendo que la obtención de una gran variedad de autores sospechosos es importante en el trabajo de un agente policial, lo que más genera valor es **la captura de perfiles que lleven correctamente hacia un “real” sospechoso (mayor precisión), en vez de a un “posible” sospechoso (mayor recuperación)**. Veamos por ejemplo las soluciones reportadas en la competencia “PAN 2012” (que tiene el mismo nombre del conjunto de datos) donde distintos participantes también se esmeraron en la detección. En total, se generaron 16 soluciones diferentes materializadas en 40 instancias. El mejor modelo reportó una precisión del 98,04 % (superior en un 3,03 %), una recuperación del 78,74 % (inferior en un 14,94 %) y un Valor F del 87,34 % (inferior en un 7,00 %) [Inches y Crestani, 2012]. Parte de su propuesta también consistió en dividir la clasificación en dos etapas. En la primera, se identificó si en la conversación **estuvo involucrado** un depredador sexual (esto para maximizar los **verdaderos positivos**), mientras que en la segunda, se identificó si en la conversación **no estuvo involucrado** un depredador sexual (esto para minimizar los **falsos negativos**). En adición a lo anterior, se utilizó un prefiltrado de todas aquellas conversaciones que presentaran uno de los siguientes patrones: presencia de un solo participante, usuarios con menos de 6 intervenciones y al menos tres mensajes largos con secuencias de caracteres desconocidos. Las conversaciones fueron representadas en términos de una “bolsa de palabras” (BOW) cuyas ponderaciones se originaron luego de aplicar la técnica “frecuencia de término – frecuencia inversa de documento” (TF-IDF) [Villatoro-Tello *et al.*, 2012]. Las clasificaciones se realizaron

RE	EP	ACC	LS	REC	PRE	FS
1	1	92,64 %	0,1733	89,84 %	95,16 %	92,42 %
	2	94,29 %	0,1637	93,23 %	95,25 %	94,23 %
	3	94,09 %	0,1705	92,55 %	95,48 %	93,99 %
	4	94,29 %	0,1682	93,20 %	95,27 %	94,22 %
	5	94,35 %	0,1652	93,66 %	94,98 %	94,31 %
	6	94,25 %	0,1765	92,69 %	95,67 %	94,16 %
	7	94,38 %	0,1670	93,68 %	95,01 %	94,34 %
2	1	88,58 %	0,2649	79,26 %	97,42 %	87,41 %
	2	90,97 %	0,1953	84,79 %	96,75 %	90,38 %
	3	91,76 %	0,1797	86,75 %	96,41 %	91,33 %
	4	92,69 %	0,1704	88,98 %	96,11 %	92,41 %
	5	94,22 %	0,1664	92,29 %	96,00 %	94,11 %
	6	94,30 %	0,1633	92,77 %	95,70 %	94,21 %
	7	94,27 %	0,1661	92,53 %	95,88 %	94,17 %

Tabla 4: Métricas para cada repetición utilizando distintos números de *epochs* en la segunda fase. La columna RE simboliza el número de repetición; EP el número de *epochs*; ACC la exactitud; LS la pérdida; REC la recuperación; PRE la precisión; FS el Valor F.

con máquinas de soporte vectorial (SVM) y redes neuronales (NN) donde, en el primer caso, se contaron con *kernels* lineales y polinomiales, mientras que en el segundo, con una simple FNN de 10 unidades ocultas. Esta solución se caracterizó por resaltar tanto cualidades léxicas como conductuales de los datos, a diferencia de la nuestra, que únicamente se centró en capturar las relaciones entre palabras (para la primera fase) y entre conversaciones (para la segunda fase). A pesar de ser un enfoque más sencillo, resulta tener un mejor desempeño. Ahora, es probable que en la realidad se deban realizar ajustes con respecto a “qué es” lo deseable y es por ello que se podrían utilizar variantes de las métricas tales como la **media armónica ponderada entre la precisión y la recuperación** definida en la ecuación (18).

$$F_{\beta} = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 \cdot P + R} \quad (18)$$

Las constantes P y R hacen mención a la (valga la redundancia) precisión y recuperación, mientras que β , al factor de ponderación. Cuando $\beta = 1$, coincide con el Valor F estándar y tanto la precisión como recuperación tienen igual importancia. Cuando $\beta < 1$, se privilegia más la precisión y cuando $\beta > 1$, se privilegia más la recuperación. Por ejemplo, si calculáramos (18) con $\beta = 0,5$ usando los resultados del mejor modelo de la competencia, obtendríamos un 93,46 % (aproximadamente un 6,12 % por encima de la cifra estándar) [Inches y Crestani, 2012].

4.3. Aplicación de principio a fin

Muchas de las conversaciones presentes en el conjunto de datos “PAN2012” provienen desde el sitio “Perverted Justice”, una organización que se dedicó por mucho tiempo a la captura de pedófilos en diversas “salas de conversación” de libre acceso. Una de las características que tenía era permitir catalogar en una escala del 1 al 5 qué tan “baboso” era un perfil basándose en una conversación dispuesta por los administradores. Esto permitió generar una especie de “*ranking*” que hasta la fecha se mantiene. Aplicaremos nuestro modelo sobre dicho *ranking* distinguiendo las siguientes conversaciones y perfiles:

1. DavieWants2 — Con una puntuación de 4,88, la conversación contó con alrededor de 356 mensajes que, una vez agrupados, se transformaron en 15. De acuerdo a los administradores, fue una de las conversaciones más perturbadoras publicadas en el sitio. El sujeto se declaró culpable por poner en peligro el bienestar del infante (considerado como delito menor) y se programó su sentencia el 29 de julio del año 2005. Correspondió a la captura número 21 por parte del equipo en el estado de New York. [Perverted Justice, 2005a].
2. fleet_captain_jaime_wolfe — Con una puntuación de 4,88, la conversación contó con alrededor de 1950 mensajes que, una vez agrupados, se transformaron en 82. El sujeto se declaró culpable (luego de haber dicho inicialmente que era inocente) por uno de los siete delitos graves de solicitud indecente y recibió dos años de libertad, una multa y se registró perpetuamente como delincuente sexual. Correspondió a la captura número 3 por parte del equipo en el estado de California [Perverted Justice, 2004].
3. vamale_692005 — Con una puntuación de 4,84, la conversación contó con alrededor de 1850 mensajes que, una vez agrupados, se transformaron en 78. Este caso fue polémico, pues lo protagonizó un sargento de la armada norteamericana. El sujeto se declaró culpable por tener conductas sexuales sobre un menor y fue sentenciado el 28 de julio del año 2006. Correspondió a la captura número 54 por parte del equipo en el estado de Virginia [Perverted Justice, 2006].
4. kalowoodsman — Con una puntuación de 4,84, la conversación contó con alrededor de 17.599 mensajes que, una vez agrupados, se transformaron en 734. El sujeto se declaró culpable de seis cargos de solicitud informática y un cargo de intento de libertades indecentes. Fue sentenciado a diez años con siete años de suspensión y se debió registrar como delincuente sexual. Correspondió a la captura número 274 por parte del equipo en el estado de Virginia [Justice, 2008].
5. thenewperson62 — Con una puntuación de 4,83, la conversación contó con alrededor de 3.415 mensajes que, una vez agrupados, se transformaron en 143. El sujeto recibió libertad condicional en septiembre de 2007 por posesión de pornografía infantil, aunque más tarde, fue enviado a prisión en enero del año 2009 por violaciones técnicas de su libertad condicional, cumpliendo una condena de seis años. Correspondió a la captura número 342 por parte del equipo en el estado de Texas [Perverted Justice, 2009].

6. jackman_9682 — Con una puntuación de 4,78, la conversación contó con alrededor de 1.901 mensajes que, una vez agrupados, se transformaron en 80. La mayoría de las conversaciones en chat las sostenía mientras un niño pequeño estaba en una cuna detrás de él (incluso estando desnudo). El sujeto fue sentenciado a ocho años de prisión y todos los niños bajo su cuidado fueron entregados al estado por peligro inminente. Correspondió a la captura número 534 por parte del equipo en el estado de Oklahoma [Justice, 2011].
7. smileman74 — Con una puntuación de 4,72, la conversación contó con alrededor de 6.092 mensajes que, una vez agrupados, se transformaron en 254. Se caracterizaba por tener un estado de ánimo bastante inestable (cercano a la actitud de un menor) y conductas sociopáticas. El sujeto fue sentenciado a ochenta y seis meses de prisión como mínimo y cadena perpetua como máximo (en el momento de la escritura, todavía se estaban analizando los factores psicológicos y conductuales). Correspondió a la captura número 26 por parte del equipo en el estado de Washington [Perverted Justice, 2007a].
8. can_i_rape_you_anally — Con una puntuación de 4,70, la conversación contó con alrededor de 753 mensajes que, una vez agrupados, se transformaron en 32. Presentaba una fuerte obsesión con todos los aspectos relacionados la violación. El sujeto fue sentenciado a libertad condicional por cinco años (crédito por los ciento veinte días que pasó en la cárcel) y se registró (de por vida) como delincuente sexual. Correspondió a la captura número 106 por parte del equipo en el estado de California [Justice, 2006].
9. shinelfmc2005 — Con una puntuación de 4,70, la conversación contó con alrededor de 918 mensajes que, una vez agrupados, se transformaron en 39. El sujeto fue arrestado y se le registró como delincuente sexual durante quince años; se le prohibió el contacto con menores y computadoras para no violar los términos de su libertad condicional de cinco años. Recibió una multa de doscientas horas de servicio comunitario. Correspondió a la captura número 96 por parte del equipo en el estado de Texas [Perverted Justice, 2005b].
10. swgamaleyess — Con una puntuación de 4,70, la conversación contó con alrededor de 6.306 mensajes que, una vez agrupados, se transformaron en 263. El sujeto fue sentenciado a veinticinco años y se registró como delincuente sexual de por vida. Correspondió a la captura número 240 por parte del equipo en el estado de Georgia [Perverted Justice, 2007b].

Los resultados de la experimentación se pueden encontrar a través de la tabla (5). La conversación del perfil (3) tuvo que ser truncada hasta los 512 mensajes, ya que no se contaba con suficiente memoria RAM por parte de las instancias. Podemos notar rápidamente que 9 de los 10 perfiles fueron considerados como sospechosos con casi absoluta certeza, lo cual es un logro considerable sabiendo la peligrosidad que significaron los imputados en la sociedad. Sin embargo, no quedan claras las razones de por qué el perfil (2) se catalogó como no sospechoso pese a que muchas de las características típicas del *grooming* existieron. Una

Perfil	Sospechoso
DavieWants2	99,76 %
fleet_captain_jaime_wolfe	0,00 %
vamale_692005	99,42 %
kalowoodsman	99,84 %
thenewperson62	99,99 %
jackman_9682	99,97 %
smileman74	99,75 %
can_i_rape_you_anally	99,96 %
shinelfmc2005	99,87 %
swgamaleyess	99,90 %

Tabla 5: Resultados de la aplicación del modelo sobre los diez perfiles más “babosos” según el sitio Perverted Justice

explicación podría ser que las cualidades demostradas corresponden a una nueva clase de acechadores poco o nada presentes dentro de los datos de entrenamiento. Descartamos que se deba a motivos relacionados con la longitud de la secuencia porque en casos como (1) (que cuenta con pocos mensajes) el desempeño se mantiene.

CAPÍTULO 5

CONCLUSIONES

Esta nueva era digital ha cambiado la forma de comunicarnos haciendo que, muchas de las interacciones, sean efectuadas a través de medios digitales. Es comprensible, por un lado, que muchas personas simplemente se rehúsan al cambio, sin embargo, olvidan que hay una población en vías de desarrollo que no cuenta con la madurez ni las herramientas suficientes para concebir los riesgos inherentes en Internet tales como *cyberbullying*, contacto con extraños, *sexting*, visualización de imágenes sexuales y *grooming*. Este último (discutido a lo largo de la memoria) nos ha permitido entender que, a través de la sobreexposición desmesurada de los menores, se pueden adquirir e imitar actitudes típicas, incluyendo el lenguaje, pensamientos y gustos personales, los que luego, son aprovechados por distintos acechadores para establecer una relación emocional y de confianza que, más tarde, se transforma en un escenario de manipulación, explotación o abuso sexual. Hay distintos factores en juego que pueden facilitar estos cometidos: la rápida introducción de Internet desde edades tempranas, el mayor acceso a dispositivos móviles, la COVID-19 al forzar que millones de niños pasen más tiempo conectados y el uso de plataformas digitales como un lugar donde estos exploran su sexualidad. Es importante que exista conciencia sobre esta nueva convivencia digital, porque son los niños quienes se han acostumbrado a coexistir en ella y han encontrado un lugar más donde poder establecer sus propias relaciones sociales y donde crear una imagen de sí mismos distinta a la que reciben de sus padres o profesores. Si no se actúa oportunamente, las consecuencias son claras: ansiedad, depresión, estrés postraumático, autolesiones, sentimientos de culpa, trastornos de alimentación, alcoholismo, drogadicción y hasta pensamientos suicidas. Mediante este párrafo, damos por cumplido el objetivo específico (1).

El problema de detectar pedófilos y abusadores sexuales en redes sociales se ha abordado de manera deficiente en la literatura debido a la falta de datos y aspectos legales que involucren el uso de contenido que está a disposición de las policías locales. Muchos investigadores han utilizado los trabajos llevados a cabo por la fundación Perverted Justice (PJ) como materia prima para entrenar sus modelos, sin embargo, sus resultados han dejado poco espacio para la comparación, ya que, por cada uno, se han utilizado distintos pasos y transformaciones que han dado lugar a nuevos conjuntos de datos, muchas veces, distantes entre sí y no disponibles públicamente. Producto de lo anterior, “PAN2012” se ha consolidado como un marco de referencia a la hora de medir la eficacia entre las distintas alternativas para la detección del *grooming*. Gran parte de las soluciones propuestas han utilizado modelos de procesamiento de lenguaje natural (NLP), ya que los focos de interacción se han dado mediante mensajería instantánea. La técnica más común evidenciada ha sido utilizar n-gramas para vectorizar los textos y apoyarse en clasificadores KNN o SVM. En general, el uso de redes neuronales como una propuesta de inicio a fin no ha sido el fuerte, principalmente, debido a la interpretabilidad de las decisiones tomadas en las capas ocultas y porque la naturaleza binaria del problema hace que los clasificadores tradicionales sean suficientes. Por lo tanto,

y mediante este párrafo, damos por cumplido el objetivo específico (2).

Las recientes mejoras empíricas debido a la transferencia de aprendizaje por parte de modelos de lenguaje no supervisados han demostrado ser una parte integral en la maquetación de sistemas robustos. En particular, BERT no ha sido la excepción. A través de entrenamientos con no más de quince minutos para la primera fase, se lograron resultados sobresalientes en cuanto al reconocimiento de patrones lingüísticos y actitudinales en mensajes aislados y típicos de un acechador sexual. Pese a que en el conjunto de datos “PAN2012” tan solo se contara con aproximadamente un 1 % de conversaciones sospechosas, tanto la precisión como la recuperación lograron un equilibrio justo. Estas conclusiones se repitieron en las métricas evidenciadas durante la segunda fase, pero bajo un leve sacrificio en el desempeño que, de todas formas, se supo sobreponer a las propuestas tradicionales, pese a que solo se utilizaran los “incrustados” dentro de una arquitectura simplista y sin un enfoque atencional. En conclusión, damos por cumplido el objetivo específico (3).

Al probar la solución en diez casos reales de conversaciones protagonizadas por acechadores sexuales, pudimos validar tres aspectos: eficacia, robustez y certinidad. El primero, porque se reconocieron adecuadamente distintas manifestaciones de la manipulación, explotación o abuso sexual hacia menores por parte de una inteligencia artificial. El segundo, porque pese a la limitada cantidad de información, muchas veces, con ruido, faltas ortográficas, ausencias de coherencia, cohesión y concordancia, no se evidenciaron degradaciones en la capacidad de generalización. El tercero, porque las decisiones manifestaron absoluta convicción (un aspecto deseable considerando la naturaleza de los ejemplos). Por lo tanto, damos por cumplido el objetivo específico (4).

Finalmente, contamos con todos los antecedentes para concluir que sí se logró desarrollar un sistema de procesamiento de texto capaz de discriminar sentencias con fines perversos hacia menores utilizando un modelo de inteligencia artificial BERT. A través de distintas comparaciones, revelamos una mejora sustancial en la capacidad de generalización con respecto a las propuestas tradicionales. Identificamos las limitaciones técnicas y supimos sacar provecho de ellas a través de una nueva arquitectura que supo mantener la proporcionalidad en las distintas métricas. Creemos que es un primer acercamiento hacia una herramienta que aprovecha todo el potencial del estado del arte en el procesamiento de lenguajes naturales para apoyar y facilitar el trabajo de los agentes policiales durante sus investigaciones, sobre todo, en un mundo acelerado donde, muchas veces, los padres y cuidadores (por diversos motivos) no tienen el control sobre las acciones de sus menores en Internet y en el momento que deciden actuar ya es demasiado tarde. Aunque muchas de las condenas por delitos son paliativas (es decir, post-acto), estamos convencidos de que, para estos casos, sí es posible contar con condenas preventivas (es decir, pre-acto) porque las evidencias están dadas bajo la propia naturaleza de la comunicación por mensajería instantánea.

Para cerrar este capítulo, invitamos a la investigación de nuevas propuestas que unifiquen ambas fases en una sola arquitectura. Comentamos en secciones anteriores que los problemas OOM fueron recurrentes dados los limitados recursos de las máquinas. Indagar en una propuesta que permita, por un lado, aprovechar las ventajas de los modelos atencionales, y

a su vez, manejar los recursos y la complejidad computacional de manera óptima, significaría una gran oportunidad para que distintos agentes (no solo policiales) incorporen una solución autocontenida.

Los códigos pueden ser encontrados a través del enlace <https://github.com/franco-parra-usm/memoria.git>.

ANEXOS

5.1. Keras Tuner

Cuando se indica un rango de valores a tomar, digamos $x_{\min}$ (el mínimo) y $x_{\max}$ (el máximo), el espacio de búsqueda se puede muestrear de distintas maneras, ya sea, lineal, logarítmico o logarítmico invertido. En todos ellos, se requiere de una variable $x \in [0, 1)$ y una función $s(x)$ que cuando $x = 0$, su valor sea $x_{\min}$ y que cuando $x = 1$, su valor sea $x_{\max}$. La cantidad de divisiones que se genere en esta recta está dada por el parámetro “paso”, que define la distancia entre un punto y otro. Si $s(x)$ retorna enteros, el paso aceptará enteros (si no se especifica un valor, se asumirá la unidad). En cambio, si $s(x)$ retorna flotantes, el paso aceptará flotantes (si no se especifica un valor, se asuma el mínimo común múltiplo entre $x_{\min}$ y $x_{\max}$).

Cuando se utiliza un **muestreo lineal**, la fórmula es la siguiente:

$$s(x) = x_{\min} + x \cdot (x_{\max} - x_{\min}) \quad (19)$$

Para el **muestreo logarítmico**, la fórmula es la siguiente:

$$s(x) = x_{\min} \left(\frac{x_{\max}}{x_{\min}} \right)^x \quad (20)$$

5.2. AdamW

Es el algoritmo de optimización Adam, pero en el que se usa una regularización con “decaimiento de pesos” (*weight decay*) desacoplada de la actualización del gradiente [Loshchilov y Hutter, 2017]. Este “desacople” viene por el hecho de que, en la literatura, la fórmula de actualización de pesos usando un optimizador SGD con regularización “L2”, o bien, con regularización de “decaimiento de pesos”, era la misma, pues utilizaban una sustitución que combinaba la tasa de aprendizaje y la tasa de decaimiento de la propuesta original de Hanson y Pratt [Hanson y Pratt, 1988, Loshchilov y Hutter, 2017]. Entre sus parámetros más importantes encontramos la tasa de aprendizaje α , la tasa de decaimiento exponencial para la estimación del primer *momentum* β_1 , la tasa de decaimiento exponencial para la estimación del segundo *momentum* β_2 y la tasa de decaimiento de pesos fija λ .

5.3. Programación de las tasas de aprendizaje

A veces, es deseable que la tasa de aprendizaje varíe en función del número de pasos o *epochs* (por ejemplo). Aunque depende de la implementación, se suelen definir dos valores: una tasa de aprendizaje inicial y una tasa de aprendizaje final. Los tipos más comunes de programación son:

- Lineal: La curva que describe la “caída” entre el inicio y fin es lineal. Qué tan rápido lo haga depende de la “variable de gatillado”.
- Exponencial: La curva que describe la “caída” entre el inicio y fin es exponencial. En esta memoria se utiliza la implementación de la librería “Keras” y su fórmula es la siguiente:

$$f(s) = \alpha \cdot \beta^{\frac{s}{\gamma}}, \quad (21)$$

donde:

- $\alpha \in \mathbb{R}$ es la tasa de aprendizaje inicial.
- $\beta \in \mathbb{R}$ es la tasa de decaimiento exponencial.
- $\gamma \in \mathbb{R}$ es “cada cuántos” pasos se aplica el decaimiento. Funciona únicamente cuando se utiliza un decaimiento exponencial escalonado, es decir, cada vez que $s \bmod \gamma = 0$. En caso contrario, el decaimiento es continuo.
- $s \in \mathbb{N}_0$ es el paso actual.

5.4. Métricas

La exactitud (*accuracy*) es la fracción de predicciones que el modelo realiza correctamente. Dado que nuestro problema involucra clasificaciones binarias, su materialización se da por:

$$\text{ACC}(t_p, t_n, f_p, f_n) = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}, \quad (22)$$

donde:

- $t_p \in \mathbb{N}_0$ es la cantidad de verdaderos positivos.
- $t_n \in \mathbb{N}_0$ es la cantidad de verdaderos negativos.
- $f_p \in \mathbb{N}_0$ es la cantidad de falsos positivos.

- $f_n \in \mathbb{N}_0$ es la cantidad de falsos negativos.

La pérdida (*loss*) mide la diferencia entre las predicciones del modelo y sus valores esperados. Aunque ya adelantamos esta definición en la sección (2.3), es necesario extenderla a una muestra completa. Dado que nuestro problema involucra clasificaciones binarias, su materialización se da a través de la siguiente expresión:

$$LS(\mathbf{y}^{(1:n)}, \mathbf{o}^{(1:n,m)}) = \frac{1}{n} \sum_{i=1}^n - \left(\mathbf{y}_1^{(i)} \cdot \ln(\mathbf{o}_1^{(i,m)}) + (1 - \mathbf{y}_1^{(i)}) \cdot \ln(1 - \mathbf{o}_1^{(i,m)}) \right), \quad (23)$$

donde:

- $n \in \mathbb{N}$ es el tamaño de la muestra.
- $m \in \mathbb{N}$ es el índice de la última capa de la red.
- $\mathbf{y}_1^{(i)} \in \{0, 1\}$ es el valor esperado de la i -ésima muestra.
- $\mathbf{o}_1^{(i,m)} \in \mathbb{R}$ es la predicción del modelo para la i -ésima muestra.

Notar que, a diferencia de la ecuación descrita en la sección (8), se prefiere el logaritmo natural. Este cambio solo se justifica porque en la muestra de resultados se utilizan las librerías Keras y TensorFlow que nativamente hacen uso de dicha función.

La recuperación (*recall*) es la fracción de verdaderos positivos identificados correctamente. Su materialización se da a través de la siguiente expresión:

$$REC(t_p, f_n) = \frac{t_p}{t_p + f_n}. \quad (24)$$

La precisión (*precision*) es la fracción de identificaciones positivas correctas. Su materialización se da a través de la siguiente expresión:

$$PRE(t_p, f_p) = \frac{t_p}{t_p + f_p}. \quad (25)$$

El Valor F (*F-Score*) es una media armónica entre la precisión y la recuperación. Su materialización se da a través de la siguiente expresión:

$$FS(t_p, t_n, f_p, f_n) = 2 \cdot \frac{PRE(t_p, f_p) \cdot REC(t_p, f_n)}{PRE(t_p, f_p) + REC(t_p, f_n)}. \quad (26)$$

REFERENCIAS BIBLIOGRÁFICAS

- [Abbate, 1994] Abbate, J. E. (1994). From arpanet to internet: A history of arpa-sponsored computer networks, 1966-1988.
- [Arab y Díaz, 2015] Arab, L. E. y Díaz, G. A. (2015). Impacto de las redes sociales e internet en la adolescencia: Aspectos positivos y negativos. *Revista Médica Clínica Las Condes*, 26(1):7-13.
- [Argamon *et al.*, 2009] Argamon, S., Koppel, M., Pennebaker, J. W., y Schler, J. (2009). Automatically profiling the author of an anonymous text. *Communications of the ACM*, 52(2):119-123.
- [Baeovski *et al.*, 2019] Baeovski, A., Edunov, S., Liu, Y., Zettlemoyer, L., y Auli, M. (2019). Cloze-driven pretraining of self-attention networks. *CoRR*, abs/1903.07785.
- [Bahdanau *et al.*, 2014] Bahdanau, D., Cho, K., y Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *ArXiv*, 1409.
- [Beltagy *et al.*, 2020] Beltagy, I., Peters, M. E., y Cohan, A. (2020). Longformer: The long-document transformer. *CoRR*, abs/2004.05150.
- [Ben-Yair, 2015] Ben-Yair, S. (2015). Introducing the newest member of our family, the youtube kids app-available on google play and the app store.
- [Bengio *et al.*, 1994] Bengio, Y., Simard, P., y Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157-166.
- [Berger *et al.*, 2019] Berger, J., Werbach, K., Fader, P., y Yildirim, P. (2019). The impact of social media: Is it irreplaceable?
- [Bird y Polivoda, 2021] Bird, G. y Polivoda, M. E. (2021). Backpropagation through time for networks with long-term dependencies. *CoRR*, abs/2103.15589.
- [Bogdanova *et al.*, 2012] Bogdanova, D., Rosso, P., y Solorio, T. (2012). On the impact of sentiment and emotion based features in detecting online sexual predators. En *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, pp. 110-118, Jeju, Korea. Association for Computational Linguistics.
- [Bogdanova *et al.*, 2014] Bogdanova, D., Rosso, P., y Solorio, T. (2014). Exploring high-level features for detecting cyberpedophilia. *Computer Speech & Language*, 28(1):108-120.
- [Bowles y Keller, 2019] Bowles, N. y Keller, M. H. (2019). Video games and online chats are 'hunting grounds' for sexual predators.

- [Buchanan y Whitty, 2013] Buchanan, T. y Whitty, M. T. (2013). The online dating romance scam: Causes and consequences of victimhood. *Psychology, Crime & Law*, 20(3):261–283.
- [Cano *et al.*, 2014] Cano, A. E., Fernandez, M., y Alani, H. (2014). Detecting child grooming behaviour patterns on social media. *Lecture Notes in Computer Science*, p. 412–427.
- [Caspi y Gorsky, 2006] Caspi, A. y Gorsky, P. (2006). Online deception: Prevalence, motivation, and emotion. *CyberPsychology & Behavior*, 9(1):54–59.
- [Castro, 2022] Castro, S. (2022). Relaciones interpersonales... mejor o peor pero ¡relacionémonos!
- [Ceci, 2022] Ceci, L. (2022). Distribution of tiktok users in the united states in 2022, by age group.
- [CEDEB, 2021] CEDEB (2021). Abuso sexual infantil: detección y herramientas para la prevención.
- [Centro de Psicología López de Fez, 2022] Centro de Psicología López de Fez (2022). ¿por qué queremos tener pareja?
- [CESOLAA, 2017] CESOLAA (2017). Características del abuso sexual en adolescentes y jóvenes.
- [Cheng *et al.*, 2016] Cheng, J., Dong, L., y Lapata, M. (2016). Long short-term memory-networks for machine reading. *CoRR*, abs/1601.06733.
- [Cho *et al.*, 2014] Cho, K., van Merriënboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., y Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.
- [Conneau *et al.*, 2019] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., y Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.
- [Cortico, 2018] Cortico (2018). Measuring the health of our public conversations.
- [Cotterell *et al.*, 2019] Cotterell, R., Kumar, A., y Schütze, H. (2019). Morphological segmentation inside-out. *CoRR*, abs/1911.04916.
- [CSIRT, 2021a] CSIRT (2021a). Ciberconsejos para una navegación segura.
- [CSIRT, 2021b] CSIRT (2021b). Navegación segura para niños, niñas y adolescentes.
- [Código Penal de Chile, 2022] Código Penal de Chile (2022). Artículo 362 del código penal de Chile.
- [Dai y Le, 2015] Dai, A. M. y Le, Q. V. (2015). Semi-supervised sequence learning. *CoRR*, abs/1511.01432.

- [Dai *et al.*, 2019] Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., y Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.
- [DataReportal, 2022] DataReportal (2022). Global social media statistics.
- [Demilie, 2020] Demilie, W. B. (2020). Analysis of implemented part of speech tagger approaches: The case of ethiopian languages. *Indian Journal of Science and Technology*, 13(48):4661–4671.
- [Devlin *et al.*, 2018] Devlin, J., Chang, M., Lee, K., y Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.
- [Dixon, 2022a] Dixon, S. (2022a). Global daily social media usage 2022.
- [Dixon, 2022b] Dixon, S. (2022b). Global facebook user age: gender distribution 2022.
- [Dixon, 2022c] Dixon, S. (2022c). Instagram: Age distribution of global audiences 2022.
- [Donath, 1996] Donath, J. (1996). Identity and deception in the virtual community.
- [Dronda, 2005] Dronda, B. (2005). *Dossier para una educación intercultural: El concepto de identidad*. 1 edición.
- [Drotner y Livingstone, 2008] Drotner, K. y Livingstone, S. (2008). The international handbook of children, media and culture.
- [Ebrahimi *et al.*, 2016] Ebrahimi, M., Suen, C. Y., y Ormandjieva, O. (2016). Detecting predatory conversations in social media by deep convolutional neural networks. *Digital Investigation*, 18:33–49.
- [Editorial Etecé, 2022] Editorial Etecé (2022). Oración: qué es, partes, tipos, características y frases.
- [eSafety Commissioner, 2021] eSafety Commissioner (2021). Safety by design.
- [Evgeniou y Pontil, 2001] Evgeniou, T. y Pontil, M. (2001). Support vector machines: Theory and applications. volumen 2049, pp. 249–257.
- [Facebook, 2020] Facebook (2020). How facebook uses super-efficient ai models to detect hate speech.
- [Fairhurst y Da Costa-Abreu, 2011] Fairhurst, M. y Da Costa-Abreu, M. (2011). Using keystroke dynamics for gender identification in social network environment. *Imaging for Crime Detection and Prevention 2011 (ICDP 2011), 4th International Conference on*, p. 1 – 6.
- [Fernández, 2022] Fernández, R. (2022). Internet: Número de usuarios mundial 2005-2021.

- [Gao *et al.*, 2015] Gao, T., Dontcheva, M., Adar, E., Liu, Z., y Karahalios, K. G. (2015). Datatone: Managing ambiguity in natural language interfaces for data visualization. En *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, p. 489–500, New York, NY, USA. Association for Computing Machinery.
- [Gehring *et al.*, 2017] Gehring, J., Auli, M., Grangier, D., Yarats, D., y Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.
- [Giot *et al.*, 2009] Giot, R., El-Abed, M., y Christophe, R. (2009). Greyc keystroke: a benchmark for keystroke dynamics biometric systems. En *IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009)*, Washington, District of Columbia, USA. IEEE Computer Society.
- [Giot y Rosenberger, 2012] Giot, R. y Rosenberger, C. (2012). A new soft biometric approach for keystroke dynamics based on gender recognition. *International Journal of Information Technology and Management*, 11(1/2):35.
- [Goswami *et al.*, 2009] Goswami, S., Sarkar, S., y Rustagi, M. (2009). Stylometric analysis of bloggers' age and gender. *Proceedings of the International AAAI Conference on Web and Social Media*, 3(1):214–217.
- [Greff *et al.*, 2015] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., y Schmidhuber, J. (2015). LSTM: A search space odyssey. *CoRR*, abs/1503.04069.
- [Gu *et al.*, 2018] Gu, J., Wang, Y., Chen, Y., Cho, K., y Li, V. O. K. (2018). Meta-learning for low-resource neural machine translation.
- [Hallebeek, 1986] Hallebeek, J. (1986). volumen 1 de 34-35, p. 205–216. *Boletín AEPE*, 1 edición.
- [Hancock y Woodworth, 2013] Hancock, J. y Woodworth, M. (2013). *An “Eye” for an “I”: The Challenges and Opportunities for Spotting Credibility in a Digital World*, pp. 325–340. Springer New York, New York, NY.
- [Hanson y Pratt, 1988] Hanson, S. y Pratt, L. (1988). Comparing biases for minimal network construction with back-propagation. pp. 177–185.
- [Hawk *et al.*, 2019] Hawk, S. T., van den Eijnden, R. J., van Lissa, C. J., y ter Bogt, T. F. (2019). Narcissistic adolescents' attention-seeking following social rejection: Links with social media disclosure, problematic social media use, and smartphone stress. *Computers in Human Behavior*, 92:65–75.
- [He *et al.*, 2015] He, K., Zhang, X., Ren, S., y Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.
- [Hicks y Gasca, 2019] Hicks, D. y Gasca, D. (2019). A healthier twitter: Progress and more to do.

- [Hochreiter y Schmidhuber, 1997] Hochreiter, S. y Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- [Howard y Ruder, 2018] Howard, J. y Ruder, S. (2018). Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.
- [Inches y Crestani, 2012] Inches, G. y Crestani, F. (2012). Overview of the international sexual predator identification competition at pan-2012.
- [Instagram, 2021] Instagram (2021). Continuing to make instagram safer for the youngest members of our community.
- [Instagram, 2022a] Instagram (2022a). How we address potentially harmful content on feed and stories.
- [Instagram, 2022b] Instagram (2022b). Normas comunitarias.
- [Internet Society, 2011] Internet Society (2011). Understanding your online identity: An overview of identity.
- [Joty *et al.*, 2019] Joty, S., Carenini, G., Ng, R., y Murray, G. (2019). Discourse analysis and its applications. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pp. 12–17, Florence, Italy. Association for Computational Linguistics.
- [Justice, 2006] Justice, P. (2006). The largest and best anti-predator organization online: chatlogs from can_i_rape_you_anally.
- [Justice, 2008] Justice, P. (2008). The largest and best anti-predator organization online: chatlogs from kalowoodsman.
- [Justice, 2011] Justice, P. (2011). The largest and best anti-predator organization online: chatlogs from jackman_9682.
- [Kaiser y Bengio, 2016] Kaiser, L. y Bengio, S. (2016). Can active memory replace attention? *CoRR*, abs/1610.08613.
- [Kalchbrenner *et al.*, 2016] Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., y Kavukcuoglu, K. (2016). Neural machine translation in linear time. *CoRR*, abs/1610.10099.
- [Kemp, 2022] Kemp, S. (2022). The global state of digital in july 2022.
- [Kim *et al.*, 2017] Kim, Y., Denton, C., Hoang, L., y Rush, A. M. (2017). Structured attention networks. *CoRR*, abs/1702.00887.
- [Lample *et al.*, 2018] Lample, G., Ott, M., Conneau, A., Denoyer, L., y Ranzato, M. (2018). Phrase-based & neural unsupervised machine translation. *CoRR*, abs/1804.07755.

- [Li *et al.*, 2021] Li, M. Q., Fung, B. C. M., y Abusitta, A. (2021). On the effectiveness of interpretable feedforward neural network.
- [Lin *et al.*, 2017] Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., y Bengio, Y. (2017). A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.
- [Liu *et al.*, 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., y Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.
- [Loshchilov y Hutter, 2017] Loshchilov, I. y Hutter, F. (2017). Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.
- [Luong *et al.*, 2015] Luong, M., Pham, H., y Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.
- [Martín *et al.*, 1998] Martín, J., Bernal, A., y Moral, M. d. I. V. (1998). La construcción de la conexión entre percepción de la autoimagen física en adolescentes y la identidad psicosocial. *Aula abierta*, ISSN 0210-2773, N° 71, 1998, pags. 145-172.
- [McCulloch y Pitts, 1943] McCulloch, W. S. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.
- [McGhee *et al.*, 2011] McGhee, I., Bayzick, J., Kontostathis, A., Edwards, L., McBride, A., y Jakubowski, E. (2011). Learning to identify internet sexual predation. *International Journal of Electronic Commerce*, 15(3):103–122.
- [Mikolov *et al.*, 2013a] Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013a). Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013.
- [Mikolov *et al.*, 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., y Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.
- [Misra *et al.*, 2019] Misra, K., Devarapalli, H., Ringenberg, T. R., y Rayz, J. T. (2019). Authorship analysis of online predatory conversations using character level convolution neural networks. En *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 623–628.
- [Muñoz *et al.*, 2021] Muñoz, F., Isaza, G., y Castillo, L. (2021). SMARTSEC4COP: Smart Cyber-Grooming Detection Using Natural Language Processing and Convolutional Neural Networks, pp. 11–20.
- [Naher y Minar, 2018] Naher, J. y Minar, M. R. (2018). Impact of social media posts in real life violence: A case study in bangladesh. *CoRR*, abs/1812.08660.
- [National Center for Missing & Exploited Children, 2016] National Center for Missing & Exploited Children (2016). Trends identified in cybertripline sextortion reports. *National Center for Missing & Exploited Children*.

- [Ngejane *et al.*, 2021] Ngejane, C., Eloff, J., Sefara, T., y Marivate, V. (2021). Digital forensics supported by machine learning for the detection of online sexual predatory chats. *Forensic Science International: Digital Investigation*, 36:301109.
- [NSPCC, 2022] NSPCC (2022). What parents need to know about sexual grooming.
- [Olah, 2015] Olah, C. (2015). Colah's blog: Understanding lstm networks.
- [Olejniczak y Šulc, 2023] Olejniczak, K. y Šulc, M. (2023). Text detection forgot about document ocr.
- [Orrù, 2014] Orrù, P. (2014). Racist discourse on social networks: a discourse analysis of facebook posts in italy. *Rhesis: International Journal of Linguistics, Philology and Literature*, 51:113–133.
- [O'Donnell, 2021] O'Donnell, B. (2021). Online enticement reports skyrocket in 2020.
- [Packeer y Kannangara, 2022] Packeer, S. y Kannangara, D. (2022). Detection of pedophilia content online: A case study using telegram. *Iraqi Journal for Computer Science and Mathematics*.
- [Panchenko *et al.*, 2012] Panchenko, A., Beaufort, R., y Fairon, C. (2012). Detection of child sexual abuse media on p2p networks: Normalization and classification of associated filenames.
- [Parapar *et al.*, 2014] Parapar, J., Losada, D. E., y Barreiro, J. (2014). Combining psycholinguistic, content-based and chat-based features to detect predation in chatrooms. *Journal of Universal Computer Science*, 20(2):213 – 239.
- [Parikh *et al.*, 2016] Parikh, A. P., Täckström, O., Das, D., y Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *CoRR*, abs/1606.01933.
- [Pascanu *et al.*, 2012] Pascanu, R., Mikolov, T., y Bengio, Y. (2012). Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.
- [Paulus *et al.*, 2017] Paulus, R., Xiong, C., y Socher, R. (2017). A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.
- [Peersman *et al.*, 2011] Peersman, C., Daelemans, W., y Vaerenbergh, L. (2011). Predicting age and gender in online social networks. pp. 37–44.
- [Pendar, 2007] Pendar, N. (2007). Toward spotting the pedophile telling victim from predator in text chats. pp. 235 – 241.
- [Pennebaker *et al.*, 2003] Pennebaker, J. W., Mehl, M. R., y Niederhoffer, K. G. (2003). Psychological aspects of natural language use: Our words, our selves. *Annual Review of Psychology*, 54:547 – 577.

- [Perverted Justice, 2003] Perverted Justice (2003). Perverted justice: The largest and best anti-predator organization online.
- [Perverted Justice, 2004] Perverted Justice (2004). The largest and best anti-predator organization online: chatlogs from fleet_captain_jaime_wolfe.
- [Perverted Justice, 2005a] Perverted Justice (2005a). The largest and best anti-predator organization online: chatlogs from daviewants2.
- [Perverted Justice, 2005b] Perverted Justice (2005b). The largest and best anti-predator organization online: chatlogs from shinelfmc2005.
- [Perverted Justice, 2006] Perverted Justice (2006). The largest and best anti-predator organization online: chatlogs from vamale_692005.
- [Perverted Justice, 2007a] Perverted Justice (2007a). The largest and best anti-predator organization online: chatlogs from smileman74.
- [Perverted Justice, 2007b] Perverted Justice (2007b). The largest and best anti-predator organization online: chatlogs from swgamaleyess.
- [Perverted Justice, 2009] Perverted Justice (2009). The largest and best anti-predator organization online: chatlogs from thenewperson62.
- [Peters *et al.*, 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., y Zettlemoyer, L. (2018). Deep contextualized word representations. *CoRR*, abs/1802.05365.
- [Ponzetto y Poesio, 2009] Ponzetto, S. P. y Poesio, M. (2009). State-of-the-art NLP approaches to coreference resolution: Theory and practical recipes. En *Tutorial Abstracts of ACL-IJCNLP 2009*, p. 6, Suntec, Singapore. Association for Computational Linguistics.
- [Primer, 2022] Primer (2022). A new state of the art for named entity recognition.
- [Radford y Narasimhan, 2018] Radford, A. y Narasimhan, K. (2018). Improving language understanding by generative pre-training.
- [Rangel *et al.*, 2013] Rangel, F., Rosso, P., Koppel, M., Stamatatos, E., e Inches, G. (2013). Overview of the author profiling task at pan 2013. En *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*, pp. 352–365. CELCT.
- [Rangel *et al.*, 2016] Rangel, F., Rosso, P., Verhoeven, B., Daelemans, W., Potthast, M., y Stein, B. (2016). Overview of the 4th author profiling task at pan 2016: cross-genre evaluations. En *Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings/Balog, Krisztian [edit.]; et al.*, pp. 750–784.
- [Retkowski, 2023] Retkowski, F. (2023). The current state of summarization.

- [Rodriguez *et al.*, 2022] Rodriguez, J. A., Vazquez, D., Laradji, I., Pedersoli, M., y Rodriguez, P. (2022). Ocr-vqgan: Taming text-within-image generation.
- [Sak *et al.*, 2014] Sak, H., Senior, A. W., y Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128.
- [Schuster y Nakajima, 2012] Schuster, M. y Nakajima, K. (2012). Japanese and korean voice search. En *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152.
- [Schuster y Paliwal, 1997] Schuster, M. y Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.
- [Shabahang *et al.*, 2022] Shabahang, R., Shim, H., Aruguete, M., y Zsila, Z. (2022). Oversharing on social media: Anxiety, attention-seeking, and social media addiction predict the breadth and depth of sharing. *Psychological Reports*, p. 003329412211228.
- [Shemtov, 1997] Shemtov, H. (1997). Ambiguity management in natural language generation. *Stanford University*.
- [Sherstinsky, 2018] Sherstinsky, A. (2018). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314.
- [Simkin, 2013] Simkin, Hugo Becerra, G. (2013). El proceso de socialización. apuntes para su exploración en el campo psicosocial. *Ciencia, Docencia y Tecnología*.
- [Singh *et al.*, 2013] Singh, J., Joshi, N., y Mathur, I. (2013). Part of speech tagging of marathi text using trigram method. *CoRR*, abs/1307.4299.
- [Smahel *et al.*, 2020] Smahel, D., Machackova, H., Mascheroni, G., Dedkova, L., Staksrud, E., Ólafsson, K., Livingstone, S., y Hasebrink, U. (2020). Eu kids online 2020: Survey results from 19 countries.
- [Snyder, 2000] Snyder, H. (2000). Sexual assault of young children as reported to law enforcement: Victim, incident, and offender characteristics. *Bureau of Justice Statistics*.
- [Sutskever *et al.*, 2014] Sutskever, I., Vinyals, O., y Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.
- [Syed Idrus *et al.*, 2014] Syed Idrus, S. Z., Cherrier, E., Rosenberger, C., y Bours, P. (2014). Soft biometrics for keystroke dynamics: Profiling individuals while typing passwords. *Computers & Security*, 45:147–155.
- [Sánchez-Martínez *et al.*, 2008] Sánchez-Martínez, F., Pérez-Ortiz, J. A., y Forcada, M. L. (2008). Using target-language information to train part-of-speech taggers for machine translation. *Machine Translation*, 22(1/2):29–66.
- [TikTok, 2021] TikTok (2021). Teen privacy and safety settings.

- [Umber y Sarwar, 2011] Umber, A. y Sarwar, I. (2011). Minimizing ambiguity in natural language software requirements specification.
- [United Nations, 2021] United Nations (2021). Global issues: Youth.
- [Valkenburg y Peter, 2011] Valkenburg, P. M. y Peter, J. (2011). Online communication among adolescents: An integrated model of its attraction, opportunities, and risks. *Journal of Adolescent Health*, 48(2):121–127.
- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.
- [Vendemia et al., 2019] Vendemia, M. A., Bond, R. M., y DeAndrea, D. C. (2019). The strategic presentation of user comments affects how political messages are evaluated on social media sites: Evidence for robust effects across party lines. *Computers in Human Behavior*, 91:279–289.
- [Villatoro-Tello et al., 2012] Villatoro-Tello, E., Juárez-González, A., Escalante, H. J., y Gómez, M. M., y Villaseñor-Pineda, L. (2012). A two-step approach for effective detection of misbehaving users in chats. En *Conference and Labs of the Evaluation Forum*.
- [V.K et al., 2021] V.K, L., Das, S., T, M., K.K, R., PaulDhinakaran, D., y Vijai, C. (2021). The impact of social networking sites on youth. *Annals of the Romanian Society for Cell Biology*, 25:5419–5431.
- [Wang et al., 2020] Wang, S., Li, B. Z., Khabsa, M., Fang, H., y Ma, H. (2020). Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.
- [WeProtect, 2021] WeProtect (2021). Global threat assesment 2021. *WeProtect*.
- [Werbos, 1990] Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.
- [Whitty, 2002] Whitty, M. T. (2002). Liar, liar! an examination of how open, supportive and honest people are in chat rooms. *Computers in Human Behavior*, 18(4):343–352.
- [Whitty, 2008] Whitty, M. T. (2008). Revealing the ‘real’ me, searching for the ‘actual’ you: Presentations of self on an internet dating site. *Computers in Human Behavior*, 24(4):1707–1723.
- [YouTube, 2021] YouTube (2021). What is a supervised experience on youtube?
- [Zuo et al., 2019] Zuo, Z., Li, J., Wei, B., Yang, L., Chao, F., y Naik, N. (2019). Adaptive activation function generation for artificial neural networks through fuzzy inference with application in grooming text categorisation. En *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6.