Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática
Trabajo de Titulación

Resumen paper base tesis

Nombre alumno:
Eduardo Abarca

Correo:
eduardo.abarca.c@usach.cl

Profesor guía:
Francisco Muñoz

17 de noviembre de 2023

Índice

1. Resumen

LLM → Cambio de paradigma en campo NLP → Capacidades notables para generar respuestas de texto similares a las humanas → Arma de doble filo: Desinformación, fake news, falta a la integridad académica.

Idea básica: Detectar si un texto fué generado por IA o no.

Idea básica paralela: Presentar las limitantes que presentan los detectores frente a escenarios de engaño.

Objetivo del paper: Realizar una revisión exhaustiva de las estrategias de detección abordadas hasta el día de hoy, categorizando y entregando una visión general, con sus capacidades y limitantes.

2. Introducción

Pregunta crucial:

¿Cómo podemos aprovechar los beneficios de los LLM y, al mismo tiempo, mitigar sus posibles inconvenientes?

Chakraborty et al. (2023) destaca que siempre debe ser posible detectar texto generado por IA aumentando la longitud del texto, a menos que las distribuciones de los textos generados por humanos y por máquinas sean las mismas en todo el soporte → el modelo de lenguaje es teóricamente óptimo.

2.1. Nuestro foco

- **Hacia las posibilidades de AI-GTD**: Marcos notables y recientes diseñados para detectar texto generado por IA. 2 grupos principales según su facilidad de uso: Preemtive (Prepared, uso preventivo) (Sec. 4.1) y post-hoc (Sec. 4.2). Además, clasificamos los detectores preemtive en dos subgrupos principales según el principio de funcionamiento: Basados en marcas de agua (Sec. 4.1.1) y basados en recuperación (Sec. 4.1.4). En función del principio de muestreo, los detectores de marcas de agua pueden granularizarse aún más en marcas de agua de muestreo sesgado y marcas de agua pseudoaleatorias. Por otro lado, los detectores post-hoc abarcan principalmente la detección zero-shot (Sec. 4.2.1) y la detección basada en fine-tuning sobre clasificadores (Sec. 4.2.2).
- **Hacia las imposibilidades de AI-GTD**: Diferentes esquemas de ataque diseñados para evadir la detección. Subdividimos los ataques en tres categorías principales: ataques basados en paráfrasis (Sec. 5.1), ataques de copiar y pegar (Sec. 5.3), ataques generativos (Sec. 5.2) y ataques de suplantación (Sec. 5.4).

3. Preliminares: Detección de texto generado por IA

3.1. Modelo de lenguaje (LM)

Definición matemática de LM:

- V set de vocabulario
- $\mathcal{L}_\theta \rightarrow$ Lenguaje de modelo parametrizado por θ
- Tokens de entrada: $h := \{h_1, h_2, ..., h_N\}$, donde $h_i \in V$
- Generación de tokens de salida:
  - Token $s_0 \in V \rightarrow$ Input = [h]
  - Token $s_1 \in V \rightarrow$ Input = [h, s_0]
  - Token generalizado $s_t \in V \rightarrow$: $[h, s_{1:t-1}]$
- Para generar cada token de salida, el LM arroja como salida un vector |V|-dimensional ℓ_t correspondiente a cada token en V:

$$\ell_t = \mathcal{L}_\theta([h, s_{1:t-1}])$$

- Luego, se usa una función *softmax* para mapear el vector ℓ_t a una distribución de probabilidad sobre V, el cuál se denota:

$$\mathbb{P}_{\mathcal{L}_\theta}(s_t = \cdot|[h : s_{1:t-1}])$$

- La probabilidad de muestrear el token $s_i \in V$ está dado por:

$$p_t(i) = \frac{e^{\ell_t(i)}}{\sum_{v \in V} e^{\ell_t(v)}}$$

- Finalmente, el token s_t se genera aplicando muestreo sobre la distribución p_t. Algunos ejemplos de técnicas de muestreo: Greedy Sampling, Random Sampling, Beam Search, Top-p sampling y Top-k sampling.

3.2. Detección de texto generado por IA

Comúnmente abordado como un problema de clasificación binaria, el objetivo es clasificar si un texto fué generado por IA o escrito por un humano. Una idea básica con notaciones matemáticos:

- Sea S conjunto de todas las posibles secuencias, $s \in S$ una secuencia y $\mathcal{D}(s) : S \rightarrow \mathbb{R}$ un detector de texto que retorna un escalar, el cuál se compara con un umbral δ:
  - $\mathcal{D}(s) \geq \delta \Rightarrow$ Contenido generado por IA
  - $\mathcal{D}(s) < \delta \Rightarrow$ Contenido de otras fuentes

4. Revisión teórica de detección de texto generado por IA

4.1. Resultado de imposibilidad (Impossibility Result)

- Los actuales detectores de texto generado por IA de última generación, basados en el uso de esquemas de marcas de agua (Kirchenbauer et al., 2023a) y clasificadores zero-shot (Mitchell et al., 2023), no son fiables en escenarios prácticos → Un parafraseo simple basado en una red neuronal ligera, como PEGASUS (Zhang et al., 2020), puede reducir significativamente la precisión de detección de los detectores de texto basados en marcas de agua (Kirchenbauer et al., 2023a) sin causar un cambio drástico en la puntuación de perplejidad.

- Cuando hay una fuerte superposición entre las distribuciones de texto generadas por humanos y por el modelo, incluso el mejor detector de texto tiene un rendimiento marginalmente mejor que un clasificador aleatorio

- **Entendimiento teórico:** Formalmente, sea M y H las distribuciones de texto de un modelo y un humano. El área bajo la curva ROC[1] de un detector post-hoc D se puede acotar como:

$$AUROC(D) \leq \frac{1}{2} + TV(M,H) - \frac{TV(M,H)^2}{2} \tag{1}$$

- Donde TV indica la distancia de variación total entre las dos distribuciones definidas como, para cualquier evento E:

$$TV(M,H) = max_E |P_{s\sim H}[s \in E] - P_{s\sim M}[s \in E]| \tag{2}$$

- Desde el límite, fuerte superposición entre distribuciones de texto generadas por máquina (M) y por humanos (H) $\Rightarrow TV(M,H) \to 0 \Rightarrow$ Límite superior $AUROC(D) = \frac{1}{2} \Rightarrow$ Clasificador binario.

- La marca de agua resuelve este problema, ya que puede introducir sintéticamente un cambio de distribución entre las dos distribuciones. Sin embargo, la detección aún puede resultar difícil parafraseando el texto (Sadasivan et al., 2023; Krishna et al., 2023). Sea M* distribución de muestras parafraseadas, para cualquier esquema de marca de agua W, Sadasivan et al. (2023) proporciona el siguiente límite:

$$P_{s\sim M*}[s \text{ watermarked using } W] \leq P_{s\sim H}[s \text{ watermarked using } W] + TV(M*,H) \tag{3}$$

  - Distribuciones M* y H son cercanas ⇒ TV (M*,H) es mínimo
  - $P_{s\sim H}[s$ watermarked using $W]$ Alto ⇒ Incremento en falsos positivos (Clasificar texto como generado por máquina, cuando en verdad no era así)
  - $P_{s\sim M*}[s$ watermarked using $W]$ Bajo ⇒ Incremento en errores de clasificación de texto bajo marca de agua como hecho por humano

[1](Área bajo la curva característica de funcionamiento del receptor) (AUROC, Area Under the Receiver Operating Characteristic Curve). Una curva ROC es una gráfica que muestra la tasa de verdaderos positivos (TPR) en función de la tasa de falsos positivos (FPR) para diferentes valores del umbral de decisión. El umbral de decisión es el valor a partir del cual el clasificador predice que una muestra pertenece a una clase determinada.

4.2. Posibilidades ocultas de la detección de texto (Hidden Possibilities of Text Detection)

- Sección anterior es muy conservativa y no considera estos puntos: Potencial cantidad de secuencias de texto disponibles (la posibilidad de recopilar múltiples secuencias) o la longitud de la secuencia de texto
- Además, el supuesto de Sadasivan et al. (2023) que todos los escritos humanos siguen una distribución lingüística "monolítica" no es realizable en la práctica → Más bien las distribuciones son heterogéneas.
- Ejemplo Bot de twitter (Cuenta es bot o humana):

4.2.1. Escenario IID (Distribuciones independientes e idénticas)

- Dada una colección de n ejemplos de texto independiente e idénticamente distribuidos $\{s_i\}_{i=1}^n$, AUROC se puede definir de la siguiente forma:

$$AUROC(D) \leq \frac{1}{2} + TV(M^{\otimes n}, H^{\otimes n}) - \frac{TV(M^{\otimes n}, H^{\otimes n})^2}{2} \tag{4}$$

- $M^{\otimes n}$ y $H^{\otimes n}$: Distribución producto de máquina y humano
- De acuerdo a la teoría de la gran desviación, la distancia total de variación $TV(M^{\otimes n}, H^{\otimes n})$ se acerca a 1 de forma exponencial respecto al número de muestras (I_c Representa la información de Chernoff):

$$TV(M^{\otimes n}, H^{\otimes n}) = 1 - e^{n * I_c(M,H) + o(n)} \tag{5}$$

- Si se incrementa el número de muestras $n \to \infty \Rightarrow$ Incrementa el límite superior de AUROC y mejora las posibilidades de detección
- Si la distribución de texto humano y máquina están γ cerca el uno del otro ($TV(M, H) = \gamma$), para lograr un AUROC ϵ, las muestras requeridas son:

$$n = \Omega(\frac{1}{\gamma} log(\frac{1}{1-\epsilon})) \tag{6}$$

- La ecuación indica que aumenta el número de muestras necesarias para realizar la detección exponencialmente con un aumento en la superposición entre la distribución humana y la máquina.

4.2.2. No IID

- Para este escenario, el número de muestras necesarias es ($TV(m,h) = \gamma > 0$, $\epsilon \in [0{,}5, 1)$ y ρ_j, c_j muestras independientes $\forall j \in (1, 2, ..., L)$):

$$n = \Omega(\frac{1}{\delta^2} log(\frac{1}{1-\epsilon}) + \frac{1}{\gamma}\sum_{j=1}^{L}(c_j - 1)\rho_j + \sqrt{\frac{1}{\gamma^2} log(\frac{1}{1-\epsilon}) * \frac{1}{\gamma}\sum_{j=1}^{L}(c_j - 1)\rho_j}) \quad (7)$$

- n es dependiente, los casos independientes fueron analizados en Chakraborty et al. (2023)

- En resumen, el análisis de Sadasivan et al. (2023) destaca que cuando el modelo y la distribución humana se superponen o están cerca entre sí, es decir, cuando la distancia de variación total entre las distribuciones es baja, el rendimiento de la detección solo puede ser aleatorio. Sin embargo, cuando se tiene en cuenta el número o la longitud de las muestras disponibles en el momento de la detección, el análisis de Chakraborty et al. (2023) encuentra que para cualquier nivel de cercanía, existe una serie de muestras que proporcionan un sólido rendimiento de detección. Las afirmaciones teóricas se validaron en (Chakraborty et al., 2023) para conjuntos de datos reales Xsum, Squad, IMDb y Fake News con generadores y detectores de última generación. Las afirmaciones han sido validadas adicionalmente también en el contexto de marcas de agua confiables en Kirchenbauer et al. (2023b). Por lo tanto, observamos que la afirmación de posibilidad o imposibilidad requiere un contexto adicional sobre el problema, lo cual es extremadamente importante.

5. Hacia las posibilidades de detección de texto generado por IA

- El esquema de detección preemtive incluye principalmente marcas de agua y detectores basados en recuperación, que implica la participación proactiva del propietario del modelo durante el proceso de generación de texto. Por el contrario, los detectores post-hoc abarcan técnicas de detección zero-shot o clasificadores ajustados que pueden ser utilizados por terceros.

5.1. Detectores preparados (Prepared detectors)

5.1.1. Descripción general de detectores basados en marcas de agua

- Detector que usa técnicas de modelos de máscaras de relleno
- Formalmente, un esquema general de marca de agua se puede definir como el que consta de dos algoritmos probabilísticos tiempo-polinómicos:

  1. *Watermark*: Este algoritmo toma un modelo de lenguaje (indicado como $\mathcal{L}$) como entrada y modifica las salidas del modelo para codificar una señal en el texto generado. Básicamente, incorpora un identificador o señal único en el texto generado por el modelo de lenguaje.
  2. *Detect*: Este algoritmo se utiliza durante el proceso de inferencia. Dada una secuencia de texto (indicada como s) y una clave de detección (indicada como k), el algoritmo de detección determina si la secuencia de texto de entrada fué generada por el modelo de lenguaje con la marca de agua (indicada como $\hat{\mathcal{L}}$) o por cualquier otro modelo. Si el texto de entrada fue generado por el modelo de lenguaje con marca de agua, el algoritmo genera 1; de lo contrario, genera 0.

- Idealmente, un detector basado en marcas de agua debería exhibir las siguientes propiedades: (1) Preservar la distribución del texto original, (2) Ser detectable sin acceso al modelo de lenguaje y (3) Ser robusto ante perturbaciones y cambios de distribución (Kuditipudi et al., 2023a). Cristo y otros. (2023a) también destaca que (4) lo ideal es que la marca de agua sea indetectable para cualquier parte que no tenga el control de la clave secreta que define la marca de agua para que la detección sea amplia y fácilmente aplicable.
- Estos detectores pueden ser subclasificados en:

  1. Muestreo sesgado: La distribución de salida $\mathcal{L}$ del modelo con marca de agua es una distribución de probabilidad modificada en el set de vocabulario V:

  $$\hat{p_t} = \mathbb{P}_{\hat{\mathcal{L}_\theta}}(s_t = \cdot | [h, s_{1:t-1}])$$

  2. Pseudoaleatorio: La probabilidad de salida del modelo no se modifica formalmente, sino que se colapsa en una muestra particular de $\mathbb{P}_{\hat{\mathcal{L}_\theta}}$, elegida pseudoaleatoriamente según el contexto local

5.1.2. Marcas de agua con muestreador sesgado (Biased-Sampler Watermarks)

La marca de agua Biased-Sampler modifica la distribución del token en cada paso de tiempo para fomentar el muestreo de tokens de una categoría predeterminada (una lista verde en cada ficha). Como ventaja, dado que el muestreo sesgado crea un cambio genérico en la distribución de probabilidad, se puede combinar con cualquier esquema de muestreo. Sin embargo, sesgar la distribución del producto también puede conducir a mayor perplejidad y degradación de la calidad del texto. A continuación, resumimos algunos estudios notables que aprovechan un esquema de marca de agua de muestra sesgada para detectar texto generado por IA.

- **A Watermark for Large Language Models**

  - Fué la primera propuesta de una marca de agua basada en un muestreo sesgado de grandes modelos de lenguaje y derivar un detector de coincidencias para texto generado por IA basado en métricas estadísticas. Considerar que los modelos de lenguaje de marcas de agua requieren acceso a la distribución de probabilidad generada sobre el vocabulario en cada paso de tiempo.
  - Sea $s_{1:t-1} = \{s_1, s_2, ..., s_{t-1}\}$ la salida del LM hasta el paso $t-1$ y $\ell_t \in \mathbb{R}^{|V|}$ las puntuaciones logit generadas en el t-ésimo paso. Ahora, la operación de marca de agua se puede describir con los siguientes pasos:
    1. Primero, las fichas del conjunto de vocabulario V se dividen en dos subconjuntos separados, a saber, la lista roja y la verde. Este etiquetado de membresía está controlado por una semilla pseudoaleatoria dependiente del contexto generada aplicando hash al conjunto de tokens en los últimos c pasos de tiempo $\{s_{t-c}, ..., s_{t-1}\}$, donde c es el ancho del contexto. En este artículo, el ancho del contexto se establece principalmente en c = 1, es decir, solo el token en el paso de tiempo anterior s_{t-1} se usa para modular la semilla. Según esta división, la lista verde consta de $\gamma|$V$|$ tokens donde $\gamma \in (0, 1)$ representa la proporción de tokens en la lista verde y en la lista roja.
    2. Para el muestreo de s_t, Kirchenbauer et al. (2023a) analiza dos esquemas principales para codificar una marca de agua con base en estas listas verdes: (a) Marca de agua "dura" y (b) Marca de agua "suave". marca de agua dura está diseñado para muestrear siempre un token de la lista verde y nunca generar ningún token de la lista roja. Un inconveniente importante es que para secuencias de baja entropía donde el siguiente token es casi Las marcas de agua duras y deterministas pueden impedir que el modelo de lenguaje las produzca, lo que resulta en en la degradación de la calidad del texto con marca de agua. Con este fin, Kirchenbauer et al. (2023a) propone una versión "más suave" como su principal estrategia de marca de agua, en la que por cada token perteneciente al lista verde, se agrega una constante escalar (α) a las puntuaciones logit:

$$\tilde{\ell}_t[v] = \ell_t[v] + \alpha \mathbb{I}[v \in \text{green}] \tag{8}$$

       donde $\mathbb{I}$ indica una función indicadora. Esto ayuda a crear un sesgo hacia el muestreo de tokens de las listas verdes con mayor frecuencia en comparación con las listas rojas. Después de la modificación, las puntuaciones logit generalmente se pasan a través de una capa softmax y el token st se muestrea de la distribución modificada con cualquier esquema de muestreo deseado.

- Dado un pasaje de texto, para comprobar si hay una marca de agua no es necesario acceder al modelo fuente, ya que la semilla aleatoria se puede volver a generar a partir de los tokens anteriores. Para identificar si una secuencia de texto tiene marca de agua o no, basta con volver a calcular las semillas pseudoaleatorias dependientes del contexto y contar el número de fichas verdes.
- Sea $s = \{s_1, s_2, ..., s_N\}$ una secuencia de tamaño N. El número de tokens verdes en s se calcula como:

$$|s_G| = \sum_{i=1}^{n} \mathbb{I}[s[i] \in \text{green}]$$

- Luego, la detección se lleva a cabo evaluando la hipótesis nula que supone que la secuencia se genera por un escritor natural sin conocimiento de la lista verde o roja. Para probar la hipótesis nula, la puntuación z es calculada como (γ es la proporción de la lista verde):

$$z = \frac{|s_g| - \gamma N}{\sqrt{N\gamma(1-\gamma)}} \tag{9}$$

- Un pasaje de texto se clasifica como "marca de agua"si $z \geq \delta$, donde δ es el umbral de clasificación.
- Cuando la hipótesis nula es verdadera, es decir, la secuencia de texto es generada por un escritor natural, $|s_G|$ (número de fichas verdes) sigue una distribución binomial con una media de N/2 y varianza de N/4. Esto se debe a que los escritos sin conocimiento del esquema de marca de agua tendría una probabilidad γ (por ejemplo, 50 %) de que cada token sea muestreado de una lista verde y $1-\gamma$ de una lista roja.
- Con un aumento en el número de tokens, según el teorema del límite central, s_G puede aproximarse con una distribución gaussiana y, por lo tanto, puede aproximarse con una puntuación z. Se espera que el texto generado por humanos sin el conocimiento de la regla de la lista verde tenga una menor cantidad de tokens verdes y, por lo tanto, una puntuación z más baja en comparación con el texto generado por máquinas. Sin embargo, para textos más cortos el supuesto gaussiano ya no se cumple, Fernández et al. (2023) ha demostrado que la detección aún es factible utilizando una prueba real de la distribución binomial verdadera. Kirchenbauer et al. (2023a) y Fernández et al. (2023) también señalan que en caso de que un documento consista en textos repetitivos, las puntuaciones de detección definidas anteriormente podrían ser erróneas. Esto se debe a que el criterio de independencia necesario para calcular las puntuaciones z no se cumple en textos autosimilares, con muchos contextos repetidos, modificando así artificialmente la puntuación. Como solución, Fernández et al.; Kirchenbauer et al. proponer calificar solo aquellos tokens durante la detección para los cuales el contexto de la marca de agua más el token en sí $s_{t-c}, ..., s_{t-1}, s_t$ no han sido vistos anteriormente y Fernández et al. (2023) muestran que esta modificación, junto con las pruebas con una distribución binomial, conduce a tasas de falsos positivos analíticos que coinciden estrechamente con las estimaciones empíricas.

- **Provable Robust Watermarking for AI-Generated Text**

  - Estudios recientes por (Sadasivan et al., 2023; Krishna et al., 2023) han cuestionado la solidez de los modelos de detección basados en marcas de agua frente a modificaciones del texto. A la luz de esto, Zhao et al. (2023a) propusieron un modelo robusto marco basado en marcas de agua llamado GPTWatermark. GPTWatermark es similar al esquema de marca de agua introducido en Kirchenbauer et al. (2023a), excepto que para cada token utilizan una división fija que controla el lista verde y roja, es decir, ancho de contexto de 0, mientras que en Kirchenbauer et al. (2023a), la división está controlada por una semilla pseudoaleatoria basada en el hash de los tokens anteriores.
  - Una división fija como en Zhao et al. (2023a) es óptimo en términos de solidez ante las ediciones del texto. Esta robustez viene con una compensación en la detectabilidad. Como se señala en Christ et al. (2023a), la longitud del contexto (y la entropía de la secuencia) determinan qué tan detectable es una señal de marca de agua, es decir, qué tan probable es que un atacante para recuperar la marca de agua al observar texto con marca de agua. La detectabilidad también implica ese impacto en la calidad de la generación de texto o impacto en los usuarios (quienes podrían observar que es menos probable que el modelo use ciertas palabras). No se puede descartar, pero Zhao et al. (2023a) muestran que estos efectos probablemente sean pequeños en la práctica.
  - Como mejora con respecto a Kirchenbauer et al. (2023a), Zhao et al. También proporciona una comprensión teórica de las propiedades de robustez del esquema de marca de agua. Como modelo de amenaza, el artículo considera adversarios que solo tienen acceso de caja negra al modelo de lenguaje. Dado un adversario A y una secuencia con marca de agua $s = \{s_1, s_2, ..., s_n\}$, sea s_A la salida modificada adversariamente. Zhao et al asume que la edición La distancia entre s y s_A está limitada superiormente por alguna constante η tal que $ED(s, s_A) < \eta$. donde ED representa la distancia de edición y se define como el número de operaciones necesarias para transformar s en s_A. Entonces la distancia máxima de edición de texto requerida para evadir la marca de agua GPT puede venir dada por:

$$\eta = \frac{\sqrt{n}(z_s - \delta)}{1 + \frac{\gamma}{2}} \mathbb{I}[z_s - \delta < \frac{\gamma\sqrt{n}}{1 + \frac{\gamma}{2}}]$$

  - Donde z_s representa la puntuación z correspondiente a la secuencia con marca de agua s, γ representa la relación de la lista verde y δ representa el umbral de clasificación del detector. De manera similar, la distancia de edición máxima requerida para evadir el esquema de marca de agua de referencia introducido en Kirchenbauer et al. (2023a) el detector puede venir dado por:

$$\eta = \frac{\sqrt{n}(z_s - \delta)}{2 + \frac{\gamma}{2}} \mathbb{I}[z_s - \delta < \frac{\gamma\sqrt{n}}{2 + \frac{\gamma}{2}}]$$

  - , al establecer un ancho de contexto de 1. Las expresiones anteriores resaltan que se necesitan el doble de ediciones para evadir GPTWatermark en comparación con Kirchenbauer et al. (2023a). Tenga en cuenta que una serie de ataques, como la paráfrasis y los ataques generativos, no asumen que un atacante estaría restringido a realizar solo η ediciones en el texto.

- **On the Reliability of Watermarks for Large Language Models**

  - Como refinamiento del esquema básico de marca de agua introducido en Kirchenbauer et al. (2023a), en este artículo, Kirchenbauer et al. propuso un esquema de hash más robusto y mejorado para generar texto con marcas de agua.
  - Recordemos que en Kirchenbauer et al. (2023a), el esquema de hash se centra en un ancho de contexto de c = 1, es decir, para el t-ésimo paso de tiempo, la semilla aleatoria para dividir el vocabulario se genera aplicando hash al token solo en el t - 1-ésimo paso. Denominemos este esquema de hash como LeftHash. Un esquema de hash alternativo (denominado SelfHash) también utiliza el token en la posición t además de los tokens a la izquierda de t. Luego, ambos esquemas se pueden utilizar con un ancho de contexto arbitrario c. El análisis de Kirchenbauer et al. (2023b) sugiere que el uso de anchos de contexto más pequeños c proporciona más solidez para parafrasear ataques, pero está relacionado con los hallazgos de Zhao et al. (2023a) y Cristo et al. (2023a), una mayor amplitud de contexto conduce a una marca de agua menos detectable, lo que tiene un impacto menor en la calidad del texto y una mayor solidez contra ataques que intentan aplicar ingeniería inversa a la marca de agua.
  - Formalmente, una función hash se define como $f : \mathbb{N}^c \to \mathbb{N}$, que recibe un conjunto de fichas (tokens) $\{s_{t-c}, ..., s_{t-1}\}$ como entrada y genera un número pseudoaleatorio. Sea $a \in \mathbb{N}$ ser un valor de sal secreto y $P : \mathbb{N} \to \mathbb{N}$ un número entero de una función pseudoaleatoria. En este artículo, Kirchenbauer et al. introduce los siguientes esquemas de hash:
    1. Aditivo: Esta es una versión mejorada del esquema de hash introducido en Kirchenbauer et al. (2023a). Específicamente, se aumenta el ancho del contexto utilizado para el hash. La función hash se define como: $f(s) = P(a \sum_{i=1}^{c} s_i$. Esta forma de hash no es resistente a ataques que puedan eliminar un token si del contexto.
    2. Omitir: este esquema de hash solo tiene en cuenta el token más a la izquierda en el contexto: $f(s) = P(as_c)$
    3. Min: para este esquema, como sugiere el nombre, la función hash se define como el mínimo del valor hash generado usando cada token s_i en el contexto: $f(s) = \min_{i \in \{1,...,c\}} P(as_i)$
  - El estudio empírico de Kirchenbauer et al. (2023b) muestra que para anchos de contexto más grandes, las variantes Min y Skip son mucho más robustas para parafrasear ataques en comparación con el esquema de hash aditivo. Sin embargo, en términos de la diversidad del texto generado, el esquema Aditivo ocupa una mejor posición en comparación con otras variantes, en línea con Christ et al. (2023a).
  - Además, Kirchenbauer et al. (2023b) también analizan la cuestión de detectar un texto con marca de agua incrustado dentro de un pasaje mucho más grande sin marca de agua. Allí, calcular el puntaje z globalmente no proporcionaría una medida precisa. A la luz de esto, Kirchenbauer et al. (2023b) proponen una evaluación de puntuación z en ventana para detectar secuencias con marcas de agua en documentos largos. Sea s un pasaje de texto de longitud T que consta de fichas con marcas de agua. La detección se lleva a cabo generando primero un vector binario $x \in 0, 1^N$ que indica la etiqueta de membresía de cada token. Sea $p_k = \sum_{i=1}^{k} x_i$ sea la suma de

los aciertos binarios y γ sea la fracción de la lista verde. Entonces la puntuación de WinMax se calcula como:

$$z_{\text{WinMax}} = \max_{i,j} \frac{(p_j - p_i) - \gamma(j-i)}{\sqrt{\gamma(1-\gamma)(j-i)}} \text{ for } i < j$$

- **Natural language watermarking via paraphraser-based lexical substitution**

  - Qiang et al. proponer un marco de marca de agua incorporando un método de sustitución léxica basado en paráfrasis. La idea central implica generar candidatos sustitutos para cada token en una secuencia utilizando un parafraseador e incrustándolos de manera que preserven el significado.
  - Sea s_i representa el i-ésimo token en una secuencia s que se codificará utilizando el marco de marca de agua. También, sea $m \in 0, 1^N$ una secuencia de marca de agua binaria fija. El procedimiento de marca de agua implica los siguientes pasos: (1) Primero, utilizando un parafraseador previamente entrenado, se generan candidatos sustitutos para el token s_i. Sea y_i el token sustituto más probable generado por el parafraseador. (2) A continuación, el algoritmo comprueba la validez de y_i como token sustituto de s_i. Si $s_i ==$ Para(y_i), donde Para representa al parafraseador, entonces y_i se considera un token sustituto válido. (3) A continuación, la lista C = y_i, s_i se ordena en orden alfabético ascendente. Finalmente, el token codificado $(\tilde{s_i})$ se establece usando la secuencia de marca de agua m como: $\tilde{s_i}$ = C[m[i]].
  - Una vez generado el texto con marca de agua, se puede verificar la marca de agua en una secuencia determinada extrayendo la secuencia binaria de marca de agua m utilizando la técnica analizada anteriormente. El texto generado a partir de fuentes distintas al modelo de lenguaje con marca de agua daría como resultado una secuencia binaria diferente. El artículo carece de un análisis detallado para comprender el efecto de los ataques de parafraseo en este esquema de marca de agua.

5.1.3. Marcas de agua pseudoaleatorias (Pseudo-Random Watermarks)

Los esquemas de marcas de agua pseudoaleatorias (Aaronson, 2023; Christ et al., 2023a; Kuditipudi et al., 2023a) operan minimizando la distancia entre la marca de agua y la distribución original, con el objetivo de hacer que la marca de agua sea indetectable e imparcial en expectativas. A continuación, resumimos brevemente algunos marcos que aprovechan marcas de agua pseudoaleatorias para AI-GTD.

- **Aaronson Watermark**

  - Aaronson propone un esquema de marca de agua en el que, dados los tokens generados hasta t-1 pasos $s_{1:t-1} = s_1, s_2, ..., s_{t-1}$, el token en el t-ésimo paso se muestrea como $s_t = \arg\max_{v \in V} r_v^{\frac{1}{Pt(v)}}$. $r_v \in [0,1] \forall v \in V$ son números reales secretos generados usando una función pseudoaleatoria usando una clave secreta sk, $r_v = f_{sk}(s_{t-c}, ..., s_{t-1}, v)$. Dada una secuencia de prueba $s = \{s_1, s_2, ..., s_N\}$, la presencia de marca de agua se identifica mediante un umbral en la puntuación de detección, que se calcula como:

$$d = \sum_{i=1}^{N} ln(\frac{1}{1 - r'_i})$$

- donde $r'_i = f_{sk}(s_{t-c}, ..., s_{t-1}, s_t)$. Fernández et al. (2023) destacaron una propiedad interesante de este esquema de marca de agua. Teniendo en cuenta la aleatoriedad del vector secreto $r = [r1, ..., r|V|]$, la probabilidad de que s_t sea el token v es exactamente $p_t(v)$, es decir, $\mathbb{E}[\mathbb{P}(s_t = v)] = p_t(v)$. Significa que, a la espera de r, este esquema de marca de agua no sesga la distribución.

- **Undetectable Watermarks for Language Models**

  - Los enfoques anteriores de marcas de agua introducidos en Kirchenbauer et al. (2023a;b); Zhao y cols. (2023a) operan alterando la distribución del resultado del modelo, lo que lleva a la degradación de la calidad del texto generado (Christ et al., 2023a). En cambio, Cristo et al. centrarse en desarrollar un esquema de marcas de agua tal que un texto con marca de agua sea computacionalmente indistinguible de un texto sin marca de agua, una propiedad que además no implica degradación en la calidad del texto (de lo contrario, la presencia de la marca de agua sería detectable mediante la observación de una calidad reducida). Formalizan la noción criptográfica de marca de agua cuando es computacionalmente difícil distinguir un texto con marca de agua del resultado original, y luego desarrollan una marca de agua que cumple con estos requisitos. Efectivamente, la marca de agua de Christ et al. (2023b) es intratable de observar para cualquier parte que no esté en posesión de la clave secreta utilizada para codificarlo.
  - Dado un conjunto de vocabulario V, sea $\bar{V}$ represent la versión binarizada del conjunto de vocabulario donde cada token $v \in V$ está codificado como un vector binario en $\{0,1\}^{log(|V|)}$. Sea F_{sk} una función pseudoaleatoria modulada por la clave secreta $sk \in \{0,1\}^{\lambda}$ donde λ representa el parámetro de seguridad. Tener en cuenta que la salida de F_{sk} se puede interpretar como un número real en [0, 1]. Ahora, consideremos el proceso de generación del i-ésimo bit, dado que los bits anteriores $b1, b2, ..., b_{i-1}$ ya están decididos. Sea $p_i(1)$ la probabilidad de que el i-ésimo bit sea 1 según el modelo original. Para el i-ésimo bit, el modelo de marca de agua genera $b_i = 1$ si $F_{sk}(\{b_1, b_2, ..., b_{i-1}\}, i) \leq p_i(1)$, de lo contrario $b_i = 0$. Tener en cuenta que la probabilidad de que el i-ésimo bit sea 1 es exactamente $p_i(1)$ ya que la salida de F_{sk} se extrae uniformemente de [0, 1]. Por lo tanto, el texto con marca de agua generado sigue la misma distribución que el texto original, lo que lo hace computacionalmente indistinguible.
  - Para detectar la marca de agua en una secuencia determinada, se aprovecha la clave secreta sk. Sea $x = x_1, ..., x_L$ una secuencia de prueba codificada en binario. Para cada bit de texto x_i, el detector calcula una puntuación utilizando el bit actual y la clave secreta sk:

$$m(x_i, s_k) = \begin{cases} ln(F_{sk}(r_i, i)) & if \quad x_i = 1 \\ \\ ln(\frac{1}{1-F_{sk}(r_i,i)}) & if \quad x_i = 0 \end{cases}$$

  - donde $r_i = x_1, ..., x_{i-1}$. Finalmente, la puntuación se suma sobre todos los bits de texto, $c(x) = \sum_{i=1}^{L} m(x_i, s_k)$. A diferencia del texto con marca de agua, donde el valor de $F_{sk}(r_i, i)$ está correlacionado con x_i, en el texto sin marca de agua $F_{sk}(r_i, i)$ es independiente de x_i. Por lo tanto, el valor esperado de c(x) sería mayor en un texto con marca de agua en comparación con un texto sin marca de agua.

- Aunque indetectable, Christ et al. Tenga en cuenta que los esquemas de marcas de agua, en general, no se pueden hacer inamovibles y siempre se pueden evadir mediante fuertes ataques generativos que modifican todo el texto. Sin embargo, Cristo et al. (2023a) no proporcionan ningún análisis empírico para comprender la solidez real de los esquemas de marcas de agua indetectables propuestos, por ejemplo, ataques de paráfrasis parciales, lo que deja incierta la solidez empírica de las ediciones.

5.1.4. Detectores basados en recuperación

- En un estudio reciente, Krishna et al. muestran que los algoritmos de detección (Kirchenbauer et al., 2023a; Mitchell et al., 2023), a pesar de su impresionante rendimiento, pueden ser vulnerables a los ataques de parafraseo. Por lo tanto, para protegerse contra la paráfrasis, en este artículo Krishna et al. propone una estrategia de detección basada en la recuperación. Este enfoque de defensa requiere una base de datos que almacene todo el texto generado por máquina a partir de un modelo en particular y, como tal, también es una defensa que solo puede ser montada por el propietario del modelo. Específicamente, dada una solicitud de entrada y la salida correspondiente generada por el modelo de lenguaje, la API necesita almacenar tanto la solicitud como la secuencia de salida en una base de datos.

- Durante la inferencia, dada una secuencia de texto, se calculan puntuaciones de similitud entre las muestras almacenadas en la base de datos y la secuencia dada. Una alta similitud indica que la secuencia dada es generada por el mismo modelo de lenguaje. Intuitivamente, es más probable que los escritos humanos alcancen una puntuación de similitud más baja en comparación con el texto generado por máquina (teniendo en cuenta que los modelos de origen y de destino son los mismos). En general, este tipo de marco de detección basado en la recuperación es confiable y hasta cierto punto robusto ante posibles cambios en el texto causados por ataques de parafraseo (Kirchenbauer et al., 2023b), ya que los pasajes del texto pueden compararse en función de características semánticas. Este esquema también puede ser atacado y en un trabajo reciente, Sadasivan et al. (2023) muestran que, con una duración de detección fija, cinco rondas de parafraseo recursivo pueden causar una caída del 75 % en la precisión de la detección de los métodos basados en la recuperación. Sin embargo, no está claro si el fallo de una coincidencia semántica significa que la paráfrasis recursiva ha modificado demasiado el texto y ha eliminado su significado original, o si el texto es funcionalmente el mismo y está parafraseado de una manera que el comparador semántico no detecta. al. Es necesaria más investigación en esta dirección para caracterizar con precisión la solidez de la detección basada en la recuperación.

- Otra cuestión relacionada con la detección basada en la recuperación son las preocupaciones sobre la privacidad. El uso de este enfoque de detección no siempre es realizable, ya que las limitaciones y regulaciones locales, como el GDPR, pueden limitar la cantidad de datos que pueden almacenar las empresas modelo (Krishna et al., 2023). Sin embargo, la detección basada en la recuperación podría ser el enfoque de detección más preciso en jurisdicciones donde sea factible, y también podría emplearse durante procedimientos judiciales, cuando las partes pueden solicitar acceso a la base de datos a las empresas modelo.

5.2. Detectores post-evento (Post-hoc detectors)

5.2.1. Zero-shot

Son detectores que no requieren acceso a muestras de texto generadas por máquinas o escritas por humanos. Idea base: Las secuencias de texto generadas por el modelo contienen información que puede ser tomada y clasificada por un detector. escarte: Pésimo rendimiento en detección (Revisar paper de evaluación detectores).

- **GLTR: Statistical Detection and Visualization of Generated Text**: GLTR, un marco de detección introducido por Gehrmann et al. (2019), tiene como objetivo identificar texto generado por máquina aprovechando el análisis estadístico. El marco se basa en la premisa de que los modelos lingüísticos a menudo seleccionan palabras con alta probabilidad, lo que conduce a ciertos patrones. Para lograr la detección de texto automatizada, se realizan tres pruebas en una secuencia determinada. Estos incluyen evaluar la probabilidad y el rango de los tokens generados, así como evaluar la entropía de la distribución generada. Las puntuaciones altas en las dos primeras pruebas sugieren que el token se deriva de la parte superior de la distribución, mientras que una puntuación baja en la tercera prueba indica la alta confianza del modelo en el token generado. El marco opera en función de las puntuaciones de detección definidas: dprob, bebió y dentropy, donde cada puntuación tiene un propósito específico en el proceso de detección. Los hallazgos empíricos del estudio revelan que los escritores humanos están más inclinados a utilizar palabras con menor probabilidad y rango en comparación con los modelos de lenguaje. El mecanismo de detección de GLTR aprovecha estas distinciones entre texto generado por humanos y por máquinas para identificar eficazmente este último.

- **DetectGPT**:

  - DetectGPT, un algoritmo de detección de disparo cero propuesto por Mitchell et al. (2023), tiene como objetivo abordar las limitaciones de los marcos existentes que se basan en umbrales de probabilidad logarítmica para identificar texto generado por máquina. Los autores sostienen que estos marcos pasan por alto la estructura local de la función de probabilidad. DetectGPT opera basándose en la premisa de que el texto generado por modelos de lenguaje (LLM) tiende a ocupar regiones de curvatura negativa en la función de probabilidad logarítmica del modelo. El algoritmo se inicia generando k perturbaciones de la secuencia de entrada utilizando una función de perturbación, utilizándose el modelo T5 para las perturbaciones en el estudio. El proceso de detección implica calcular una métrica de discrepancia, que considera la probabilidad logarítmica de la muestra original y sus versiones perturbadas, junto con la varianza del texto perturbado. Los resultados ilustran que el texto generado por máquina muestra una métrica de discrepancia más alta en comparación con el texto generado por humanos, lo que respalda la eficacia del algoritmo en la detección.
  - Sin embargo, DetectGPT tiene limitaciones notables. Asume acceso de caja blanca a los parámetros del modelo, lo que no siempre es factible en aplicaciones del mundo real. Además, en comparación con otros detectores de disparo cero basados en métodos estadísticos de detección de valores atípicos, DetectGPT es computacionalmente más intensivo. Además, Krishna et al. (2023) demuestran que incluso un solo pase de parafraseo con un modelo diferente puede reducir significativamente la precisión de detección de DetectGPT, destacando su vulnerabilidad a ataques adversarios.

- **Bot or Human? Detecting ChatGPT Imposters with A Single Question**: Los autores se centran en el desafío de distinguir entre robots conversacionales basados en LM y usuarios humanos en escenarios en tiempo real. Argumentan que los métodos tradicionales como los CAPTCHA son inadecuados para identificar la naturaleza de los usuarios, dados los recientes avances en la generación de texto basada en LM. Para abordar este problema, introducen un marco de tiro cero llamado FLAIR, que implica crear preguntas que exploten las fortalezas y debilidades únicas de los LLM. Los autores destacan las diferencias en las capacidades entre los robots conversacionales basados en LLM y los humanos, como los robots que luchan con tareas que involucran manipulación, filtrado de ruido y aleatoriedad, mientras sobresalen en memorización y cálculos. Para demostrar estas diferencias, presentan preguntas de muestra que apuntan a estas áreas específicas, ilustrando efectivamente las distinciones entre las dos entidades.

- **DNA-GPT**:

  - Los autores abordan las limitaciones de los marcos de detección de disparo cero como DetectGPT que dependen del acceso a los parámetros del modelo. Al reconocer el desafío que plantea la falta de disponibilidad de distribuciones de probabilidad de tokens, especialmente en modelos como GPT-3.5, proponen una estrategia de detección para escenarios de caja negra donde los detectores solo tienen acceso a nivel API.
  - El método propuesto implica truncar la secuencia de entrada en dos partes, aprovechando la singularidad de los modelos de lenguaje para generar n-gramas comparables. Esta estrategia alimenta la primera parte de la secuencia como un mensaje de entrada al modelo de lenguaje, generando múltiples secuencias de salida. El proceso de detección implica comparar la similitud de n-gramas entre las secuencias generadas y la segunda parte de la secuencia original. La puntuación de detección se calcula utilizando una fórmula que incorpora una función de ponderación para diferentes n-gramas, con parámetros predeterminados establecidos como f(n) = nlog(n), n0 = 4 y N = 25. Una puntuación de detección más alta indica que la Es probable que la secuencia sea generada por una máquina.
  - En la configuración de caja blanca, donde está disponible el acceso a las probabilidades de token generadas, los autores utilizan la distribución de probabilidad generada para calcular la puntuación de detección. Esta puntuación se calcula en función de la probabilidad logarítmica de la segunda parte de la secuencia dada la primera parte, en relación con la probabilidad de cada secuencia generada dada la primera parte. Los métodos propuestos tienen como objetivo facilitar la detección en escenarios donde el acceso directo a los parámetros del modelo podría no ser factible, ofreciendo información valiosa para detectar texto generado por máquina.

- **Smaller Language Models are Better Black-box AI-GTD**:

  - Los autores se centran en el desafío de detectar texto generado por IA en una configuración de modelo cruzado de caja negra, donde el detector carece de acceso a los parámetros del modelo fuente. Para abordar esto, utilizan la prueba de optimización local de Mitchell et al. (2023), operando bajo el supuesto de que el texto generado por máquina tiende a ser localmente óptimo en comparación con el texto escrito por

humanos. Esto implica generar k perturbaciones de una secuencia determinada utilizando un modelo de perturbación como T5 (Raffel et al., 2020) y, posteriormente, calcular la curvatura como: $d(s) = log\mathcal{L}_\phi(s) - \frac{1}{k}\sum_i log\mathcal{L}_\phi(\bar{s}_i)$, donde $\mathcal{L}_\phi$ representa un modelo de lenguaje desconocido distinto del modelo fuente.

- El artículo evalúa 27 modelos de detectores diferentes con distintos tamaños de parámetros que van desde 70M hasta 6,7B. El análisis empírico revela dos tendencias significativas. En primer lugar, para la detección entre modelos, los modelos más pequeños exhiben un rendimiento superior en comparación con los modelos de mayor capacidad. Además, el estudio destaca que una familia de arquitectura y una superposición de conjuntos de datos entre el modelo de generador y detector conducen a una mejora en el rendimiento de la detección. Además, los resultados indican que los modelos parcialmente entrenados superan a los modelos completamente entrenados. Los autores observan que el punto de control final, que representa la culminación del proceso de capacitación, produce consistentemente el peor desempeño en términos de detección de texto generado por máquina. Sugieren que este fenómeno podría atribuirse al sobreajuste durante el entrenamiento prolongado.

5.2.2. Fine-tuning sobre clasificadores

- Clasificación binaria usando características extraídas de un LM pre-entrenado para la detección de texto generado por máquina
- Requiere aprendizaje supervisado: Muestras de texto generado por máquina y humano
- Contras:
  - Recolectar la información para entrenar el modelo
  - Por avances de LMs, texto generado por máquina y humano son muy parecidos
  - Tasa de falsos positivos difícil de calcular, depende de la distribución de datos utilizada para entrenar el detector
  - Si el modelo del detector es más pequeño que el modelo analizado, se reduce la eficacia de la detección
- Ejemplos:

1. **GPT-Sentinel**
   - Abordando el problema de la detección de texto generado por máquinas, Chen et al. (2023) proponen dos enfoques simples: (1) entrenar un clasificador lineal simple sobre un modelo RoBERTa previamente entrenado (Liu et al., 2019) y (2) ajustar un modelo T5 (Raffel et al., 2020). . Para ello, Chen et al. (2023) seleccionan un conjunto de datos, a saber, OpenGPTtext, parafraseando muestras textuales del corpus OpenWebText (Gokaslan et al., 2019) utilizando el modelo GPT-3.5-turbo. Específicamente, OpenGPTtext consta de 29,395 muestras, donde cada muestra textual corresponde a un texto escrito por humanos del corpus OpenWebText. Demuestran que tanto el ajuste fino como la técnica de sondeo lineal en modelos previamente entrenados dan como resultado un rendimiento mejorado en la identificación de texto generado por máquina en comparación con técnicas de referencia como las presentadas por Tian (2023),

OpenAI (2023) y Solaiman et al. (2019). El enfoque de ajuste fino muestra una mayor precisión en la detección de texto que el método de sondeo lineal.

2. **LLMDet: A Large Language Models Detection Tool**
   - Para detectar texto generado por máquina, Wu et al. proponer un marco de detección de texto llamado LLMDet. Un estudio previo de Mitchell et al. (2023) ha destacado la utilidad de la perplejidad en la detección de texto generado por máquinas. Sin embargo, el cálculo de la perplejidad requiere un acceso de caja blanca a los parámetros del modelo original, lo que puede ser irrealizable en la práctica. Para aprovechar la utilidad de la perplejidad sin asumir un acceso de caja blanca, Wu et al. introducir la noción de calcular una puntuación proxy de la perplejidad. El marco de detección de texto consta de dos fases: (1) la fase de Diccionario y (2) la fase de Capacitación.
   - En la fase de diccionario, los n-gramas y sus correspondientes probabilidades se almacenan como claves y valores respectivamente en un diccionario. Primero, se solicita repetidamente al modelo de lenguaje que genere una colección de muestras de texto. A continuación, se calculan las estadísticas de frecuencia de palabras de n-gramas sobre el conjunto de texto generado. Finalmente, los n-gramas y sus correspondientes probabilidades se almacenan como claves y valores respectivamente en un diccionario. La probabilidad de un n-grama se calcula en función de las estadísticas de frecuencia del texto generadas por el modelo de lenguaje. En la fase de entrenamiento, la información almacenada en la fase del diccionario se utiliza para calcular la perplejidad del proxy. Finalmente, estas perplejidades de los proxy se utilizan para entrenar un clasificador de texto para distinguir entre texto generado por máquina y escritura humana.
3. **How Close is ChatGPT to Human Experts**
   - Las contribuciones de Guo et al. son esencialmente dobles: (1) En primer lugar, su objetivo es arrojar luz sobre qué tan cerca están los LLM de última generación de los escritores humanos. Específicamente, brindan una comprensión de las propiedades del texto generado por ChatGPT y en qué se diferencia de los escritos humanos. (2) En segundo lugar, para minimizar los riesgos relacionados con el contenido generado por IA, proponen diferentes sistemas de detección. Para comprender las diferencias en el texto generado por ChatGPT y escritores humanos, elaboran un corpus de comparación de ChatGPT humano. En el corpus, cada instancia es una pregunta seguida de respuestas de un experto humano y una respuesta generada mediante ChatGPT. Utilizando este conjunto de datos, realizan dos pruebas específicas. En la primera prueba, conocida como prueba de Turing, a la que se le da una pregunta y la respuesta correspondiente (puede ser de un escritor humano o ChatGPT), un evaluador humano tiene que identificar si la respuesta es generada por una máquina o no. En la segunda configuración, conocida como prueba de utilidad, dada una pregunta y respuestas de un escritor humano y ChatGPT, la tarea es evaluar qué respuesta entre las dos es más útil. Con base en este análisis, observan: (1) Las respuestas generadas por ChatGPT son identificadas correctamente por un evaluador humano aproximadamente el 80 % de las veces. (2) Las respuestas generadas por ChatGPT son en general más concretas y útiles que las de los escritores humanos, especialmente para preguntas del ámbito de las finanzas y la psicología. Sin embargo,

para las preguntas relacionadas con el ámbito médico, las respuestas generadas por ChatGPT no son de mucha ayuda y están mal elaboradas en comparación con los escritos humanos.

- Además, con el fin de detectar texto generado por IA, Guo et al. (2023) consideran: (1) entrenar un clasificador de regresión logística con las características obtenidas de la prueba GLTR (Gehrmann et al., 2019) y (2) ajustar un potente transformador previamente entrenado, específicamente RoBERTa (Liu et al., 2019).). Descubrieron que el detector basado en RoBERTa es más robusto y también muestra un rendimiento de detección más fuerte en comparación con GLTR.

4. **Ghostbuster: Detecting Text Ghostwritten by Large Language Models**

   - Varios estudios, como se analizó en la subsección anterior (Mitchell et al., 2023) han explorado la detección de texto generado por máquina mediante el análisis de probabilidades de registro de tokens. Sin embargo, un desafío común al generar las probabilidades de los tokens implica asumir el acceso de caja blanca al modelo de destino. En su trabajo reciente, Verma et al. (2023) proponen Ghostbuster, un marco de detección que no requiere acceso a las probabilidades simbólicas del modelo objetivo. Esto permite la detección de texto generado a partir de modelos desconocidos. Para conocer las probabilidades de registro de tokens, durante el entrenamiento, Ghostbuster pasa cada documento a través de una serie de modelos de lenguaje menos potentes. Sin embargo, en lugar de realizar la clasificación directamente en función de las probabilidades de registro generadas, las probabilidades de los tokens pasan a través de una serie de operaciones vectoriales y escalares. Estas operaciones combinan las probabilidades de los tokens dando lugar a características sintéticas adicionales.
   - Para ilustrar, considere que p_1 y p_2 son los vectores de probabilidad generados para dos tokens cualesquiera. Algunos casos de operaciones vectoriales incluyen la suma ($p_1 + p_2$), la resta (p_1 - p_2), la multiplicación ($p_1 \cdot p_2$) y la división (p_1/p_2). Las operaciones escalares incluyen cálculos como máximo (max p), mínimo (min p), longitud (|p|) y norma l-2 ($||p||_2$). Aparte de estas características sintéticas, Verma et al. propuso utilizar un conjunto de características diseñadas manualmente que incorporan conocimientos cualitativos y heurísticas observadas en el texto generado por IA. Finalmente, se entrena un clasificador de regresión logística sobre estas características para detectar texto generado por máquina.

5. **RADAR: Robust AI-Text Detection via Adversarial Learning**

   - Un desafío importante a la hora de identificar texto generado por máquinas implica abordar los ataques de parafraseo. Además, estudios anteriores (Krishna et al., 2023; Sadasivan et al., 2023) han demostrado que los ataques de parafraseo (recursivos) pueden reducir significativamente el rendimiento de la detección. Para mitigar este problema, Hu et al. (2023) propusieron un nuevo marco de detección llamado RADAR. Este marco emplea un enfoque de aprendizaje adversario para entrenar simultáneamente a un detector y un parafraseador. En un nivel superior, el objetivo del detector es diferenciar con precisión entre contenido escrito por humanos y texto generado por máquinas. Por el contrario, el entrenamiento del parafraseador implica la generación de texto plausible y realista con el objetivo de eludir la detección por parte del detector.

- Sean $\mathcal{L}_\theta, \mathcal{D}_\phi, \mathcal{G}_\sigma$ el LM objetivo, el detector y el parafraseador parametrizados por θ, ϕ y σ respectivamente. Tener en cuenta que el modelo de lenguaje objetivo $\mathcal{L}_\theta$ está congelado en su totalidad y no implica ningún tipo de capacitación. El detector $\mathcal{D}_\phi$ y el parafraseador $\mathcal{G}_\sigma$ se inicializan con modelos T5-grande y RoBERTa-grande previamente entrenados, respectivamente. Sea H un corpus de texto humano, generado a partir de muestras de 160.000 documentos de WebText (Gokaslan et al., 2019). Sea M un corpus de texto generado por IA utilizando el modelo de lenguaje objetivo $\mathcal{L}_\theta$, completando el texto utilizando los primeros 30 tokens como indicación. El procedimiento de formación consta de dos componentes:
  - Entrenando el parafraseador: Primero, las muestras de texto generadas por IA $s_m \sim M$ se envían al parafraseador $\mathcal{G}_\sigma$ como entrada. Sea s_p la salida del parafraseador y $\mathcal{P}$ el corpus de todas las muestras parafraseadas. Los parámetros del parafraseador se actualizan mediante la optimización de política próxima (Schulman et al., 2017) con la retroalimentación de recompensa del detector $\mathcal{D}_\phi$. La recompensa devuelta por s_p es la salida de $\mathcal{D}_\phi(s_p)$, es decir, la probabilidad prevista de que s_p sea escrito por humanos.
  - Entrenando el detector: Como en un entrenamiento GAN típico, el detector tiene la tarea de distinguir con precisión las muestras escritas por humanos de las contrapartes parafraseadas y generadas por IA. El detector se entrena utilizando pérdida logística definida como: $L_{\mathcal{D}_\phi} = L_H + \lambda L_M + \lambda L_\mathcal{P}$, donde $L_\mathcal{H} = -\mathbb{E}_{s_h \sim \mathcal{H}} log(\mathcal{D}_\phi(s_h))$ representa la pérdida para mejorar la exactitud de la predicción de $s_h \sim \mathcal{H}$ escrita por humanos, $L_M = -\mathbb{E}_{s_m \sim M} log(1 - \mathcal{D}_\phi(s_m))$ y $L_\mathcal{P} = -\mathbb{E}_{s_p \sim \mathcal{P}} log(1 - \mathcal{D}_\phi(s_p))$ representan la pérdida para predecir s_m y s_p al ser predichos como texto humano.
- El análisis empírico destaca el sólido rendimiento de detección de texto de RADAR en un conjunto de 8 LLM de destino. Específicamente en el conjunto de datos XSum, cuando las muestras se parafrasean utilizando una API OpenAI GPT-3.5-Turbo (que es diferente del parafraseador que se usa durante el entrenamiento RADAR), RADAR mejora el rendimiento de detección en un $16,6\,\%$ y un $59,5\,\%$ en comparación con un detector fino basado en Roberta. -sintonizado en WebText (Gokaslan et al., 2019) y DetectGPT (Mitchell et al., 2023).

6. Hacia las imposibilidades de detección de texto generado por IA

6.1. Ataques por parafraseo

6.1.1. Red Teaming Language Model Detectors with Language Models (Shi et al., 2023)

- Evadir clasificador parafraseando texto usando un LM
- Para efectos del trabajo, se asume que el sistema de detección es capaz de detectar el parafraseo
- Fundamento matemático:
  - Mensaje de entrada de texto h, $s = \{s_1, s_2, ..., s_N\}$ secuencia de salida de lado N generado por un modelo de lenguaje $\mathcal{L}_\theta$, $\mathcal{G}_\phi$ Lenguaje de modelo usado para la generación del ataque, considerar 2 ataques:
    1. Modificar la salida s para generar 's. Esto se hace reemplazando ciertos tokens en s con palabras sustitutas manteniendo la naturalidad y el significado semántico original. Para reemplazar un token s_k en la salida original s, se solicita a $\mathcal{G}_\phi$ que genere candidatos sustitutos con el mismo significado que s_k. Sea $\tilde{s_k}$ el conjunto de tokens candidatos sustitutos generados por $\mathcal{G}_\phi$. Dado un presupuesto de sustitución ϵ, el marco de minimización se define como:

$$
\begin{aligned}
& s' = \arg\min_{s'} \mathcal{D}('s) \\
& \text{s.t. } s'_k \in \{s_k\} \cup \tilde{s_k}, \\
& \sum_{i=1}^{N} \mathbb{I}[s_i \neq s'_i] \leq \epsilon N
\end{aligned}
$$

       donde $\mathcal{D}$ representa el detector de texto que está siendo atacado y $\mathbb{I}$ representa una función indicadora.
    2. Modificar la entrada h para generar 'h, así generando 's. La idea principal es modificar el mensaje de entrada para cambiar la distribución del texto generado por el idioma, evadiendo así el detector. Esto se hace agregando un mensaje h_p adicional que se puede aprender a la secuencia de entrada original h, lo que lleva a un nuevo mensaje $h' = [h, h_p]$. El mensaje adicional h_p se busca consultando el detector varias veces usando m mensajes de entrada $\{h_1, ..., h_m\}$. La función objetivo de la búsqueda se define como:

$$\arg\min_{h_p} \frac{1}{n} \sum_{i=1} \mathbb{I}[\mathbf{D}(\mathbf{L}_\theta([h_i, h_p])) \geq \delta]$$

       En palabras simples, la función de búsqueda tiene como objetivo encontrar un mensaje h_p adicional de modo que la tasa de detección promedio se minimice para los nuevos resultados generados. Esta forma de ataque está dirigida principalmente a marcos de detección basados en el ajuste o entrenamiento de un clasificador neuronal.

- El análisis empírico de Shi et al. (2023) muestra que un ataque basado en la modificación de la secuencia de salida provoca una degradación del $88,6\,\%$ en AUROC para el detector basado en DetectGPT, una mejora del $41,8\,\%$ en comparación con los ataques basados en parafraseo. Aquí el ataque se basa en el modelo GPT-2-XL y el conjunto de datos XSum.

6.1.2. Paraphrasing Evades Detectors of AI-generated Text (Krishna et al., 2023).

Los métodos de parafraseo existentes no simulan con precisión escenarios del mundo real debido a dos limitaciones clave: se entrenan en oraciones en lugar de considerar un contexto más amplio a nivel de párrafo, y carecen de la capacidad de controlar la diversidad de resultados. En consecuencia, estos métodos pueden no ser adecuados para probar exhaustivamente los algoritmos de detección de texto, ya que una mayor paráfrasis puede provocar una divergencia significativa en el significado del texto original, reduciendo su utilidad para los atacantes.

6.1.3. Paraphrasing by Humans

En una configuración similar, que probablemente ocurra en la práctica, el "atacante" podría ser simplemente un escritor humano que está parafraseando un texto con una marca de agua con el propósito de evadir la detección. Curiosamente, el análisis de Kirchenbauer et al. (2023b) muestra que los escritores humanos son realmente buenos parafraseadores y son más capaces de engañar a un detector en comparación con los parafraseadores basados en máquinas, lo que a su vez significa que el texto parafraseado por escritores humanos debe ser significativamente más largo, para la marca de agua de Kirchenbauer et al. . (2023b) para ser detectado.

6.2. Ataques generativos

- Estos ataques explotan la capacidad de los grandes modelos lingüísticos para el aprendizaje en contexto, lo que incita a estos modelos modifiquen sus respuestas de una manera que sea a la vez previsible y fácilmente reversible.

6.3. Ataques de copiar-pegar

- Se trata de una forma de evaluación sintética en la que pasajes de texto generados por IA se integran en un documento más extenso escrito por humanos.

6.4. Ataques de suplantación de identidad

- Cuando un humano adversario genera deliberadamente un pasaje de texto para que se detecte como generado por IA, es conocido como un ataque de suplantación de identidad
- El objetivo de estos ataques podría ser engañar a un detector para clasificar falsamente escritos humanos despectivos como generados por IA con la intención maliciosa de dañar la reputación de un desarrollador de LLM.

7. Preguntas abiertas

- ¿Qué tipo de regulación se puede aplicar sobre los LLM para limitar sus usos indeseados?
- ¿Cómo podemos evaluar con precisión los detectores en partes más amplias de la distribución del lenguaje humano?
- ¿Podemos mejorar el rendimiento de la detección sin asumir la disponibilidad de muestras adicionales?
- ¿Cómo evitar que los detectores de IA estén sesgados, y así culpen falsamente a escritores de inglés no nativos?
- Entonces, para resumir, la viabilidad de la detección está intrínsecamente ligada a los requisitos específicos del usuario, tales como la cantidad de texto que debe detectarse, las tasas aceptables de falsos positivos y el umbral aceptable de evidencia.
- Escenario académico: En algunos escenarios específicos, como en los sistemas educativos, la detección incorrecta puede llevar a penalización falsa de los estudiantes, lo que tiene repercusiones drásticas en sus carreras. Por eso, para tales escenarios, resulta crucial minimizar los falsos positivos mientras se detecta el contenido generado por IA. Sin embargo, observamos que, curiosamente, en estos escenarios, dado que las indicaciones siempre están disponibles para el instructor, es posible aprender la distribución de la máquina consultando modelos de código abierto. Por lo tanto, la detección se vuelve más fácil y se reduce a identificar muestras fuera de distribución. Además, el instructor puede solicitar un requisito mínimo de longitud de la oración para mejorar la detección (Chakraborty et al., 2023). Así, aunque el problema de detección es un desafío, existen alternativas para abordarlo, lo cual es una dirección interesante para futuras investigaciones.
- ¿Permite el conocimiento previo del mensaje de entrada (para ser utilizado por un adversario) simplificar la tarea de detección?
- ¿Es posible la marca de agua semántica?