UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Adaptación de modelos de lenguaje grandes para la generación de lenguaje natural a partir de palabras clave en sistemas aumentativos y alternativos de comunicación

TRABAJO FIN DE GRADO

Grado en Ciencia de Datos

Autor: Silvia Alegre Villa

Tutor: Jorge Civera Saiz

Curso 2023-2024

Resum

Els Sistemes Augmentatius i Alternatius de Comunicació (SAAC) son eines essencials per a facilitar la comunicació de les persones amb dificultats en la utilització del llenguatge. Aquest sistemes permeten a l'usuari la selecció de pictogrames associats a paraules claus que conformaran l'oració que es desitja comunicar. Posteriorment, aquesta oració pot ser sintetitzada amb veu humana. Els recents avanços en l'àrea del processament del llenguatge natural i, en concret, la proliferació de models de llenguatge grans ofereixen noves perspectives per a millorar els SAAC. En particular, aquest treball explorarà com aquests models de llenguatge poden millorar l'expressivitat de la comunicació dels SAAC quan s'utilitzen per a la generació de llenguatge natural a partir de les paraules clau (pictogrames) seleccionades per l'usuari. D'aquesta manera, aquest treball evaluar'a el rendiment d'aquests models quan són adaptats per a la seua integració en els SAAC. Awuesta evaluació es durà a terme utilitzant conjunts de test reals en espanyol i anglés extrets del portal del Centre Aragonés per a la Comunicació Augmentativa i Alternativa.

Paraules clau: ????, ?????????, ????, ??????????????

Resumen

Los Sistemas Aumentativos y Alternativos de Comunicación (SAAC) son herramientas vitales para facilitar la comunicación de las personas con dificultades en la utilización del lenguaje. Estos sistemas permiten al usuario la selección de pictogramas asociados a palabras clave que conformarán la oración que se desea comunicar. Posteriormente, esta oración puede ser sintetizada con voz humana. Los recientes avances en el área del procesamiento de lenguaje natural y, en concreto, la proliferación de modelos de lenguaje grandes ofrece nuevas perspectivas para mejorar los SAAC. En particular, este trabajo explorará cómo estos modelos de lenguaje pueden mejorar la expresividad de la comunicación de los SAAC cuando se utilizan para la generación de lenguaje natural a partir de las palabras clave (pictogramas) seleccionadas por el usuario. De esta forma, este trabajo evaluará el rendimiento de estos modelos cuando son adaptados para su integración en los SAAC. Esta evaluación se llevará a cabo utilizando conjuntos de test reales en español e inglés extraídos del portal del Centro Aragonés para la Comunicación Aumentativa y Alternativa.

Palabras clave: ?????, ???, ??????????????

Abstract

Augmentative and Alternative Communication (AAC) systems are vital tools for facilitating communication for individuals with difficulties using language. These systems allow users to select pictograms associated with key words that will form the sentence that is wished to communicate. Then, the sentence can be synthesized with a human voice. Recent advances in the field of natural language processing, and specifically the proliferation of large language models, offer new perspectives for improving AAC systems. In particular, this work will explore how these language models can enhance the expressiveness of AAC communication when used to generate natural language from the key words (pictograms) selected by the user. In this way, this work will evaluate the performance of these models when adapted for integration into AAC systems. This evaluation will be carried out using real test sets in spanish and english extracted from the portal of the Aragonese Center for Augmentative and Alternative Communication.

Key words: ?????, ????? ?????, ?????????????

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

ESCRIBIR BIEN !!! En este primer capítulo introductorio presentamos la motivación y objetivos del trabajo. También explicaremos cómo será la estructura del contenido de este.

1.1 Motivación

La comunicación y el lenguaje son dos pilares fundamentales de la sociedad actual, pues constituyen la base de las relaciones interpersonales, permitiendo el intercambio de ideas e información. Gracias a ello podemos transmitir a los demás nuestros pensamientos, emociones y necesidades, permitiéndonos participar en la vida en sociedad. Sin embargo, para algunas personas, el hecho de comunicarse de manera satisfactoria puede resultar realmente complicado debido a distintas causas. Es aquí donde entran en juego los Sistemas Aumentativos y Alternativos de Comunicación (SAAC).

Tal y como explica el Centro Aragonés para la Comunicación Aumentativa y Alternativa (ARASAAC), "los Sistemas Aumentativos y Alternativos de Comunicación son formas de expresión distintas al lenguaje hablado que tienen como objetivo aumentar el nivel de expresión y/o compensar las dificultades de comunicación y lenguaje de las personas que tienen dificultades en este ámbito". Hay distintas razones por las cuales una persona podría necesitar utilizar un SAAC. Entre ellas encontramos la parálisis cerebral, la discapacidad intelectual, los trastornos del espectro autista, algunas enfermedades neurológicas, las distrofias musculares o las afasias, entre otras.

Aunque hay muchos tipos de SAAC, todos se caracterizan por estar basados en sistemas de símbolos, ya sean gráficos (fotografías, dibujos, pictogramas, palabras o letras) o gestuales (mímica o símbolos manuales). En este trabajo nos centraremos en los comunicadores electrónicos. Los comunicadores electrónicos son herramientas tecnológicas que pueden ser utilizados en cualquier tipo de dispositivo electrónico. Por lo general, consisten en un tablero donde aparecen distintos símbolos gráficos (pictogramas) que representan palabras.

Estos pictogramas pueden ser organizados y adaptados dependiendo de las necesidades de cada persona, permitiendo que cada usuario añada aquellos que necesite. El funcionamiento es simple: el usuario clica sobre los pictogramas y el programa se encarga de dar la salida del mensaje, generalmente en forma de habla digitalizada o en formato escrito.

Así, estas herramientas resultan verdaderamente útiles para solventar los problemas de comunicación de las personas. Sin embargo, presentan una limitación que puede afectar a la fluidez y al nivel de expresividad con que se realiza la comunicación: la dificultad

Figura 1.1: Comunicador AsTeRISCS Grid desarrollado por ARASAAC

para conjugar las frases de manera adecuada, pues para facilitar y simplificar el uso de la herramienta, en este tipo de tableros se suelen incluir solamente palabras clave en su forma simple, sin distinción de número, género o tiempo verbal.

Algunos de los comunicadores que existen actualmente en el mercado ya emplean diferentes métodos para abordar este problema, pero existe todavía un amplio margen de mejora. Los modelos de lenguaje grandes prometen ofrecer buenos resultados en este área.

1.2 Objetivos

Los objetivos de este proyecto son los siguientes:

1. Investigar sobre los enfoques actuales para la generación de frases en el sector de los SAAC.
2. Implementar y evaluar modelos de lenguaje grandes para este tipo de herramientas.
3. Comparar los resultados que se obtienen con los modelos de lenguaje grandes respecto a otros comunicadores que permiten la generación de frases que se encuentran en el mercado, utilizando para ello las métricas de BLEU y COMET.

1.3 Estructura de la memoria

????? ????????????? ????????????? ????????????? ????????????? ?????????????

CAPÍTULO 2
Fundamentos

REESCRIBIR: Los modelos de lenguaje se encuadran dentro de las técnicas de procesamiento de lenguaje natural, las cuales forman parte del ámbito del aprendizaje automático. Antes de profundizar en las tareas específicas que realizan los modelos de lenguaje grandes, debemos introducir los conceptos básicos del aprendizaje automático, del aprendizaje profunod y del procesamiento de lenguaje natural. Así, en este capítulo abordaremos las distintas tareas y enfoques del aprendizaje automático, proporcionando el contexto necesario para comprender el funcionamiento y las aplicaciones de los modelos de lenguaje.

2.1 Aprendizaje automático

El aprendizaje automático (en inglés, *machine learning*) es una disciplina dentro de la inteligencia artificial que se centra en el desarrollo y estudio de algoritmos y modelos que permiten que los sistemas puedan relizar tareas específicas sin haber sido explícitamente programados para ello. Este término fue acuñado por Arthur Samuel en el año 1959, quien creo uno de los primeros programas exitosos de esta disciplina, conocido como *the Samuel Ckeckers-playing Program* [10] (el programa de juego de damas de Samuel).

Tom Mitchell [8] define el proceso de aprendizaje de los programas en el campo del *machine learning* de la siguiente manera:

"Se dice que un programa aprende de la experiencia E con respecto a alguna clase de tarea T, y medida de rendimiento P, si su rendimiento en tareas en T, medido por P, mejora con la experiencia E."

Aunque la idea principal del aprendizaje automático es esta, econtramos diferentes tipos de aprendizaje automático, dependiendo del tipo de tarea que se quiera llevar a cabo, de la naturaleza de la medida del rendimiento que se utiliza para evaluar el sistema y de el tipo de entrenamiento o experiencia que le proporcionamos a este.

Generalmente, los enfoques para el entrenamiento de algoritmos se agrupan en:

- **Aprendizaje supervisado**, cuyo objetivo es, a partir de unos datos de entrenamiento, encontrar la función f que realice el mejor mapeo posible entre un conjunto de entradas X y sus salidas correspondientes Y, de manera que $(X, Y) = (X, f(Y))$

- **Aprendizaje no supervisado**, que trata de modelar la estructura subyacente de un conjunto de datos para identificar relaciones y patrones, permitiendo así un entendimiento más profundo de los mismos

- **Aprendizaje semi-supervisado**, que cae entre el supervisado y el no supervisado y utiliza una porción de datos etiquetados y no etiquetados

- **Aprendizaje por refuerzo**, donde el algoritmo aprende a través de retroalimentaciones que va recibiendo, ajustando sus acciones con el objetivo de maximizar una recompensa acumulada a lo largo del tiempo. [7]

La tarea principal de este trabajo se realizará utilizando las técincas del aprendizaje supervisado.

Otro concepto importante dentro del aprendizaje automático es el aprendizaje profundo (*deep learning*). El aprendizaje profundo es subconjunto dentro del aprendizaje automático que emplea algoritmos basados en redes neuronales. Dentro de este encontramos métodos como las redes neuronales profundas, las redes neuronales recurrentes, las redes neuronales convulacionales y los transformers, entre otros. Estos métodos tienen aplicaciones significativas en una gran variedad de ámbitos, entre los que se encuentra el procesamiento de lenguaje natural, disciplina en la que se enmarca este trabajo. En las siguientes secciones explicaremos con detalle los conceptos de redes neuronales y transformers.

2.2 Redes neuronales

El primer modelo de red neuronal artificial fue creado en 1943 por Warren McCulloch y Walter Pitts, y se conoce como *McCulloch-Pitts neuron* [6]. Este era un modelo simplificado que imitaba el comportamiento del cerebro humano. A partir de este dio inicio a un proceso de investigación de las redes neuronales desde dos enfoques distintos: uno centrado en los procesos biológicos del cerebro y otro en la aplicación de las redes neuronales para la inteligencia artificial. Las redes neuronales modernas, aunque inspiradas en estas primeras ideas, han evolucionado significatvamente y ya no se basan directamente en las inspiraciones biológicas iniciales.

Las redes neuronales modernas están compuestas por pequeñas unidades de cómputo conocidas como neuronas o nodos, conectadas entre si a través de enlaces para permitir la transmisión de señales entre estas. Los nodos están organizados en capas, de manera que un nodo en una capa está conectado a todos los nodos de la capa siguiente. Existen tres tipos de capa: capa de entrada (*input layer*), capas ocultas (*hidden layers*) y capa de salida (*output layer*).

La arquitectura más simple de red neuronal es la de perceptrón. Este tipo de modelo se utiliza para tareas de clasificación binaria, en las que se debe decidir si un determinado *input* pertenece o no a una clase. Es un tipo de clasificador lineal, por lo que hace sus predicciones basandose en una función de predicción lineal que combina un conjunto de pesos con el vector de entrada. Este tipo de arquitectura sirve como base para arquitecturas de redes neuronales mucho más complejas.

Tal y como vemos en la figura, en la arquitectura de perceptrón encontramos solamente una neurona, que toma un vector X como entrada. La neurona calcula la combinación lineal de los elementos del vector $x_1, x_2, ..., x_n$ con los pesos correspondientes $w_1, w_2, ..., w_n$, añadiendo al resultado un valor conocido como umbral o *bias term*:

$$z = b + \sum_{i=1}^{n} w_i x_i \tag{2.1}$$

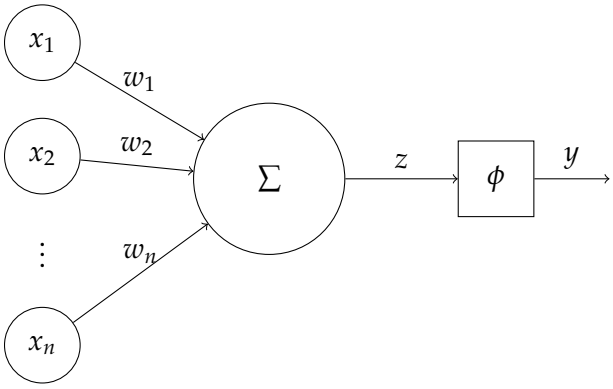

Figura 2.1: Esquema de perceptrón simple

A este resultado se le aplica una función ϕ conocida como función de activación, y finalmente se devuelve un solo valor y como salida:

$$y = \phi(z) = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases} \tag{2.2}$$

El entrenamiento de las redes neuronales consiste en ajustar los distintos pesos de la red de manera que produzca las salidas más acertadas posibles. En el caso de las redes de perceptrón, al contar con solamente una neurona, este proceso resulta relativamente sencillo. El primer paso es inicializar el vector de pesos con valores aleatorios y calcular la salida de cada vector de entrada del conjunto de entrenamiento. A continuación, se comprueba si la predicción ha sido correcta. Si no lo ha sido, el vector de pesos se modifica utilizando la siguiente fórmula:

$$w_i = w_i - \lambda(\hat{Y}^t - Y^t)X^t \tag{2.3}$$

Donde λ es la tasa de aprendizaje, $\hat{Y}$ es el *output* predicho por el modelo y Y es la clasificación real.

Estos pasos se repiten durante un número determinado de iteraciones o hasta que el modelo converge.

Este modelo de perceptrón tiene una limitación principal: el modelo solo converge si las dos clases en las cuales debe clasificar las muestras son linealmente separables. En caso de que no lo sean, los pesos oscilarán indefinidamente, hasta que el número máximo de iteraciones se alcance. Para solventar esta limitación existen modelos de redes neuronales mucho más complejos, con más neuronas y que utilizan funciones de activación no lineales. El modelo más conocido de este tipo es el de perceptrón multicapa (MLP, por su nombre en inglés: *Multi-Layer Perceptron*).

2.2.1. Perceptrón multicapa

El modelo de perceptrón multicapa es una evolución del modelo de perceptrón simple que aparece con intención de poder resolver problemas no lineales. La idea principal detrás de este es la combinación de varios perceptrones simples en un único modelo. En esta arquitectura podemos encontrar un número elevado de neuronas, conectadas entre si y divididas en capas. Encontramos tres tipos de capas: una capa de entrada (*input layer*), una o más capas ocultas (*hidden layers*) y una capa de salida (*output layer*) (ver figura 2.2).

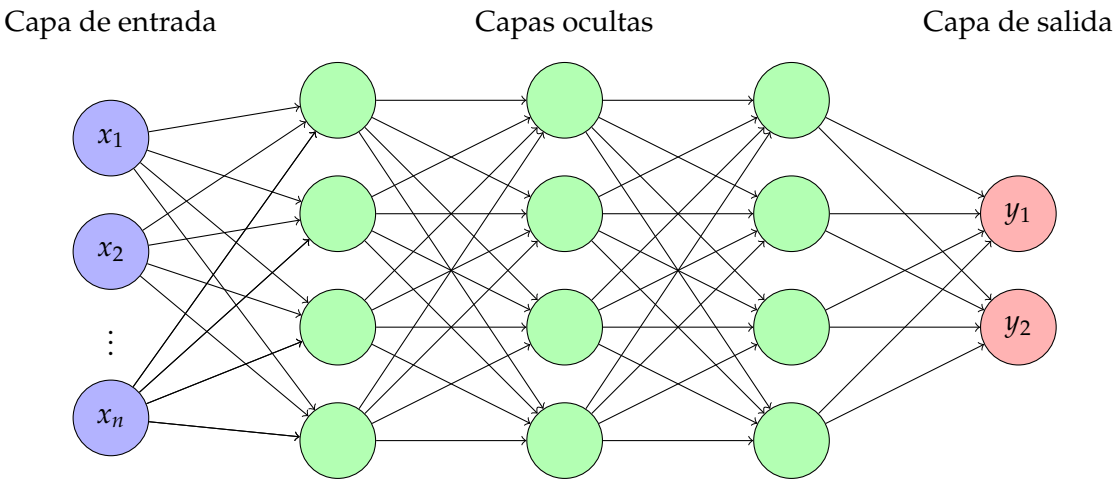

Figura 2.2: Esquema de perceptrón multicapa

Cada una de las neuronas que se encuentra en una capa está conectada a todas las neuronas de la capa siguiente, y cada uno de los enlaces tiene asociado un peso w. De manera similar al modelo de perceptrón simple, los datos entran al modelo en forma de vector a través de la capa de entrada, y se calcula el valor z_i de cada entrada utilizando la fórmula 2.1. A continuación se aplica la función de activación ϕ. Así, el resultado que pasa a la neurona siguiente es $\phi(t)$. En este tipo de modelos se utilizan funciones no lineales como función de activación, para poder aplicarse a problemas no lineales. Entre las funciones de activación más comunes se encuentran la función sigmoide, la tangente hiperbólica (*tanh*) y la rectificada lineal (*ReLU*).

Este proceso se repite en todas las capas, hasta llegar a la capa de salida y producir el *output* final.

2.2.2. Redes neuronales para secuencias de texto

El procesamiento de secuencias de texto utilizando redes neuronales es una técnica fundamental en el campo del procesamiento de lenguaje natural. Dado que las redes neuronales operan utilizando vectores numéricos, es necesario transformar las secuencias de texto en vectores antes de poder procesarlas. Este proceso se realiza en varias etapas, que se detallan a continuación:

1. **Tokenización y conversión a índices**: el primer paso consiste en convertir la frase en una secuencia de palabras o tokens. Una vez obtenida la lista de tokens, se asigna a cada uno un número que servirá como índice.

2. **Transformación de palabras en vectores**: los índices numéricos se transforman en vectores utilizando *embeddings*. Los *embeddings* son representaciones vectoriales densas y de baja dimensión que capturan las características semánticas de las palabras.

3. **Creación de la matriz de *embeddings***: una vez que se han obtenido los vectores asociados a cada una de las palabras, se puede construir una matriz de *embeddings* que representa la frase completa. Esta matriz facilita la entrada secuencial de los vectores en la red neuronal.

Cada palabra en la secuencia de texto es procesada por la red neuronal de manera secuencial. A continuación, se presentan dos tipos de red neuronal que son utilizadas para el procesamiento de este tipo de datos.

Redes neuronales recurrentes (RNN)

Las redes neuronales recurrentes (en inglés, *recurrent neural networks* (RNN)) son un tipo de red neuronal que mapean desde un conjunto de entrada a otro de salida de manera que la predicción y_t depende no solo de la de entrada x_t sino también del estado oculto del sistema, h_t [9]. El estado oculto h_t es una representación interna de la red en el momento de tiempo t que captura información relevante que ha sido procesada anteriormente. Se actualiza con el tiempo a medida que se procesan los datos. Este tipo de modelos se pueden utilizar para la generación, clasificación y traducción de texto.

El proceso que siguen este tipo de redes es el siguiente. En primer lugar, se inicializa la red con un estado oculto h_0, que puede ser un vector de ceros o una inicialización aprendida. Para cada elemento en la secuencia de entrada, la red actualiza su estado oculto y produce una salida. Así, en el momento de tiempo t, la entrada x_t y el estado oculto previo h_{t-1} se combinan para generar el nuevo estado oculto h_t. Este se calcula utilizando la fórmula:

$$h_t = f(W_h h_{t-1} + W_x x_t + b_h) \tag{2.4}$$

Donde W_h y W_x son las matrices de pesos asociadas al estado oculto y a las entradas respectivamente, f es una función no lineal y b_h es un vector de sesgos. La salida y_t se obtiene a partir del estado oculto h_t:

$$y_t = g(W_y h_t + b_y) \tag{2.5}$$

Donde W_y es una matriz de pesos, g es la función de activación y b_y es un vector de sesgos.

El proceso se repite para cada elemento de la secuencia de entrada, propagando así la información relevante anterior a través de los estados ocultos.

Uno de los principales problemas de las RNN es la dificultad para recordar información a largo plazo, pues se ha comprobado que, si el modelo es entrenado con secuencias de entrada largas, la importancia de las primeras palabras que son procesadas tiende a ir perdiendose a medida que se procesa el resto de la secuencia. Esto limita la capacidad de este tipo de redes para capturar dependencias a largo plazo en secuencias largas.

Redes neuronales convolucionales (CNN)

El funcionamiento de las redes neuronales convolucionales (*convolutional neural networks (CNN)* se basa en el calculo de una función de un vecindario local para cada una de las entradas utilizando pesos compartidos. Son una buena alternativa a las redes neuronales recurrentes, pues son sustancialmente más sencillas de entrenar, ya que no necesitan mantener el estado oculto a largo plazo. Se pueden utilizar en tareas de clasificación y de generación de texto.

2.2.3. Arquitectura *encoder-decoder*

Los modelos *encoder-decoder*, también conocidos como modelos *seq2seq* (*sequence-to-sequence*) son modelos capaces de generar secuencias de salida contextualmente apropiadas y de longitud arbitraria a partir de una secuencia de entrada [5]. Se utilizan en problemas donde la secuencia de salida generada a partir de la secuencia de entrada no tiene la misma longitud. Estos modelos son ampliamente utilizados en tareas como el resumen de textos, la respuesta a preguntas, el diálogo y la traducción automática.

Este tipo de modelos están formados por tres componentes principales:

- El codificador o *encoder*, que acepta las secuencias de entrada y genera la secuencia correspondiente de representaciones contextualizadas. Esta secuencia captura la información relevante de cada elemento de la secuencia de entrada y sus contextos.
- El vector de contexto c, que es una función de la secuencia de representaciones generada por el *encoder*. Este vector sintetiza la información necesaria de la secuencia de entrada para que el decodificador pueda generar la secuencia de salida.
- El decodificador o *decoder*, que toma el vector de contexto c como entrada y genera una secuencia de estados ocultos de longitud arbitraria. A partir de esta, se obtiene la secuencia correspondiente de estados de salida.

Esta arquitectura se puede implementar utilizando diversos tipos de redes neuronales, incluyendo los transformers, como veremos en la sección 2.3, o las redes neuronales recurrentes.

2.2.4. Atención

Tal y como se ha explicado en las secciones anteriores, en las redes neuronales clásicas, el cálculo de cada capa se realiza mediante una combinación lineal de los vectores de entrada y los correspondientes pesos, seguida de la aplicación de una función de activación. Esta operación se representa matemáticamente como $Z = \phi(XW)$, donde X es el vector de entrada, W es el vector de pesos, ϕ es la función de activación y Z son las salidas producidas en las capas. [9]

Sin embargo, los modelos de redes neuronales pueden volverse aún más flexibles y potentes si permitimos que los pesos dependan de los *inputs*. Esta interacción multiplicativa, donde los pesos son dinámicos y dependen de las entradas, se conoce como mecanismo de atención. Formalmente, puede expresarse como $Z = \phi(XW(X))$.

De manera más general, el mecanismo de atención puede describirse mediante la fórmula $Z = \phi(VW(Q, K))$. En este contexto:

- Q (*queries*) es un conjunto derivado de X que describe qué es lo que busca cada palabra, es decir, con qué otro tipo de palabras puede estar relacionada.
- K (*keys*) es otro conjunto derivado de X que se usa para describir cuáles son las propiedades de las palabras.
- V (*values*) es un conjunto también derivado de X que describe cómo cada entrada debe ser transmitida hasta la salida.

Los vectores de *queries*, *keys* y valores se obtienen procesando las palabras mediante tres redes neuronales independientes.

Cuando se utiliza la atención para calcular una salida z_i, se utiliza la *query* q_i correspondiente y se compara con cada una de las claves k_j de todas las otras palabras de la secuencia, calculando cuál es su nivel de relación con cada una. Esto se realiza mediante el producto escalar entre el vector *query* y los vectores *key* de las otras palabras de la secuencia, de la siguiente manera:

$$\alpha_{ij} = softmax(q_i \cdot k_j) \tag{2.6}$$

Se utiliza la función *softmax* para normalizar los resultados y poder así crear el correspondiente vector de pesos. Así, el resultado se representa como α_{ij} y debe cumplir las siguientes condiciones:

$$0 \leq \alpha_{ij} \leq 1 \tag{2.7}$$

$$\sum_j \alpha_{ij} = 1 \tag{2.8}$$

El coeficiente $\alpha_i j$ determina cuánto peso se debe dar a cada valor v_j en la combinación final, y representa el nivel de importancia que tiene cada palabra de la secuencia sobre la palabra i. La salida z_i se calcula entonces como una suma ponderada de los valores v_j, donde los pesos son los coeficientes calculados:

$$z_i = \sum_j \alpha_{ij} v_j \tag{2.9}$$

Este enfoque permite que los *outputs* del modelo sean una combinación dinámica ponderada de los *inputs*, lo que hace que este tipo de sistemas sean mucho más efectivos para una amplia gama de tareas, como la traducción automática, el resumen de textos o la generación de texto entre otros. [9, 5]

2.3 Transformers

En esta sección presentaremos la arquitectura de transformers.

Los modelos basados en transformers emplean una arquitectura *encoder-decoder* que utiliza el mecanismo de atención, descrito en la sección anterior. La arquitectura de los transformers fue introducida en el artículo "*Attention is All You Need*" [11], publicado en 2017. Esta supuso una gran revolución en diversas áreas del aprendizaje automático por su capacidad para manejar de manera efectiva secuencias largas y complejas [4]. Actualmente, hay una gran cantidad de modelos basados en el que se utilizan en el área del procesamiento de lenguaje natural.

2.3.1. Estructura del Transformer

Tal y como se ha mencionado anteriormente, la arquitectura de transformers utiliza el modelo *encoder-decoder* introducido en la sección 2.2.3. En este caso, se utilizan un total de seis bloques de *encoder* y seis de *decoder*, cada uno de los cuales contiene varias capas y mecanismos que facilitan el procesamiento eficiente de las secuencias de datos.

En general, los transformers utilizan tres tipos de capas: capas lineales simples, capas de *multi-head attention* y capas donde encontramos redes neuronales de tipo *feed forward*.

En la figura 2.3 encontramos el esquema de la estructura de los *encoders* y *decoders*. Tal y como se puede observar, el *encoder* está formado por una primera capa de *multi-head attention* y, a continuación, una red neuronal de tipo *feedforward* completamente conectada, además de las capas de normalización que encontramos después de estas. Por lo que respecta al *decoder*, lo conforman dos capas de autoatención y una red neuronal, también con capas de normalización después de estas. [5, 2]

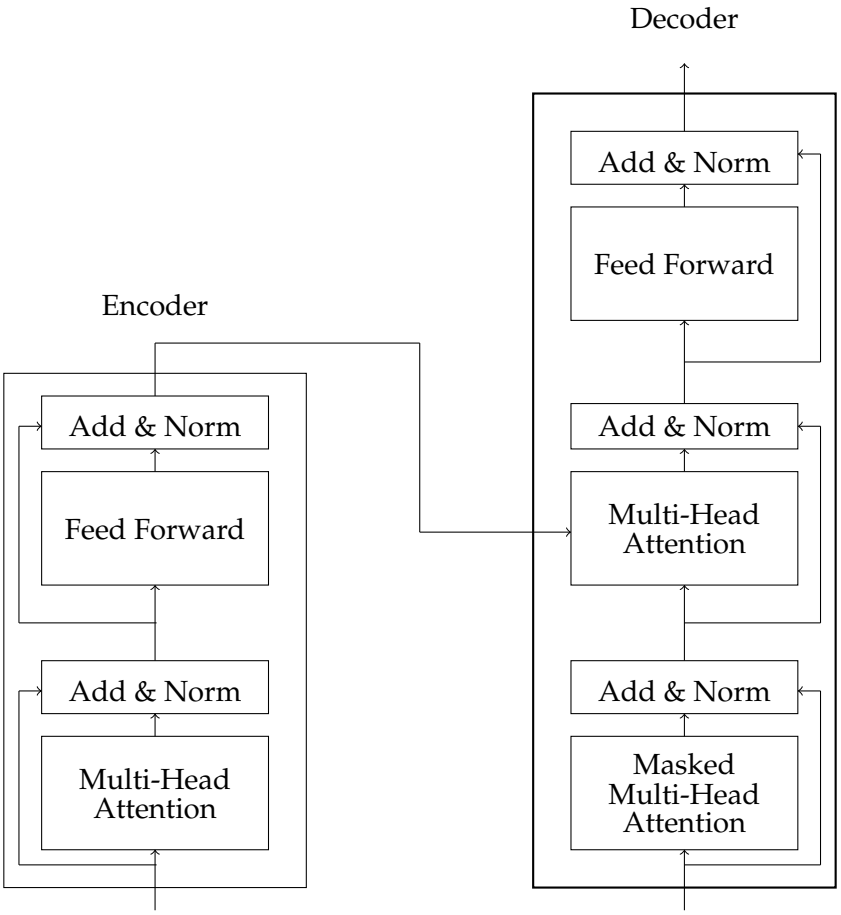

Figura 2.3: Esquema de la arquitectura de los transformers

La ide principal del transformer es, a lo largo de esta serie de capas, poder construir representaciones contextualizadas cada vez más acertadas de los significados de las palabras o *tokens* de las secuencias de entrada. Así, en las distintas capas del transformer, para obtener la representación de una palabra, se combina la información de la representación obtenida en la capa anterior con la información de las representaciones de las palabras vecinas. El objetivo es producir la versión contextualizada de cada palabra, representando lo que significa esta en el contexto particular en que se encuentra. [5]

A continuación se explica con más detalle los procesos que ocurren dentro de cada capa.

2.3.2. Capas de *multi-head attention*

Las capas de *multi-head attention* suponen la verdadera innovación de la arquitectura de transformers [5], pues es en estas donde el modelo aplica el mecansimo de atención. En el caso de los transformers, se utiliza una variación del mecanismo de atención explicado en la sección 2.2.4. Esta variación se conoce como *scaled dot-product attention*. Igual que en el mecanismo general de atención, se utilizan tres matrices: la matriz de *queries* Q, la de

claves K (por su nombre en inglés, *keys*) y la de valores V (*values*). En este caso, el valor de atención se calcula utilizando la siguiente fórmula:

$$A(Q, K, V) = softmax(\frac{QK^t}{\sqrt{d_k}}) \tag{2.10}$$

Donde d_k es la dimensión del estado oculto de la fuente.

Como vemos, el cambio respecto al cálculo de atención descrito en la sección 2.2.4 reside en que en este caso se añade el factor $\frac{1}{\sqrt{d_k}}$, que se utiliza para escalar el resultado obtenido, y conseguir así que las gradientes sean más estables, ya que si la secuencia de entrada fuera muy larga, la función *softmax* puede devolver gradientes extremadamente pequeños, cosa que podria dificultar al modelo realizar un aprendizaje eficiente. [2]

Aunque esta forma de cálculo de la atención es empleada por los transformers, estos no se quedan solamente en este concepto, sino que van un paso más allá y utilizan un tipo de atención algo más complejo conocido como *multi-head attention*. Este proceso consiste en calcular distintas atenciones sobre las mismas *queries* y pares clave-valor. Se utiliza este tipo de atención ya que las distintas palabras dentro de la secuencia de texto pueden estar relacionadas entre sí de muchas maneras distintas de manera simultánea. Capturar todas estas relaciones entre palabras es muy complicado si se utiliza un solo valor de atención.

Así, en las capas de *multi-head attention* encontramos distintas capas de atención conocidas como *heads* (cabezas), que calculan valores de atención de manera paralela. Cada una de estas tiene un conjunto distinto de parámetros que se aprende durante el entrenamiento, con los cuales hará el cálculo de atención entre las palabras de entrada. Utilizando parámetros distintos en cada cabeza se consigue que cada una se centre en aspectos distintos de las relaciones entre las palabras. El cálculo de atención para una cabeza i se realiza mediante la siguiente fórmula:

$$H_i = A(QW_i^Q, KW_i^K, VW_i^V) \tag{2.11}$$

Donde W_i^Q, W_i^K y W_i^V son las matrices de parámtetros asociadas a la *query*, clave y valor respectivamente de la cabeza i. El valor de *multi-head attention* se obtiene mediante la concatenación de los resultados de las cabezas. Para un total de h cabezas de atención:

$$MultiHead(Q, K, V) = (H_1 \oplus H_2 \oplus ... \oplus H_h)W_O \tag{2.12}$$

La matriz de parámetros W_O proyecta la concatenación de los resultados de las h cabezas de vuelta al subespacio original. [3, 2, 5]

2.3.3. Caps de redes *Feed Forward*

Las capas de redes *feed forward* en la arquitectura de transformers están compuestas por N redes neuronales independientes de tipo *feed forward* (en español, redes de propagación hacia adelante).

Una red neuronal de tipo *feed forward* es una red multicapa en la que, a diferencia de las redes neuronales recurrentes (RNN) explicadas en la sección 2.2.2, las conexiones entre neuronas no pueden formar ciclos. Esto implica que las señales siempre se propagan en una única dirección: desde la capa de entrada hacia la capa de salida, pasando por las capas ocultas. En la arquitectura de transformers, estas redes están completamente conectadas, lo que significa que las neuronas de una capa reciben las salidas de todas

las neuronas de la capa anterior y envían sus salidas a todas las neuronas de la capa siguiente. Además, típicamente contienen solamente una capa oculta.

Este tipo de red neuronal es similar al modelo de perceptró multicapa explicado en la sección 2.2.1, pero no deben confundirse, pues en el caso de las redes neuronales *feed forward*, las unidades de cómputo no son perceptrones, sino unidades más complejas. [5]

En los transformers, estas capas complementan las salidas de la capa de atención. Mientras que la capa de atención procesa cada palabra de la secuencia relacionandola con las demás palabras, las capas *feef forward* procesan cada palabra de manera independiente. Este procesamiento es crucial para mejorar la representación de las características extraídas de la capa de atención.

2.3.4. Capas de normalización

Las capas de normalización se encuentran detrás de tanto las capas de atención como las de redes *forward*. En ellas se aplica el proceso de normalización de capas (en inglés, *layer normalization* o *layer norm* [1]). Este tipo de normalización se utiliza para mejorar el rendimiento del entrenamiento de redes neuronales profundas, manteniendo los valores de la capa oculta dentro de un rango que facilita el entrenamiento basado en gradientes.

El proceso de normalización de capas toma como entrada un vector para cada palabra de la secuencia, y devuelve este mismo vector normalizado. El primer paso en el proceso es calcular la media μ y la desviación típica σ del vector, de la siguiente manera:

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} x_i^l \tag{2.13}$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(x_i^l - \mu^l)^2} \tag{2.14}$$

Donde H es el número de unidades ocultas en la capa. A continuación, los componentes del vector se normalizan restando la media y desviación típica calculadas [5]:

$$\hat{x} = \frac{(x - \mu)}{\sigma} \tag{2.15}$$

Finalmente, se introducen parámetros de *bias* b y *gain* g:

$$LayerNorm = g\hat{x} + b \tag{2.16}$$

2.4 Modelos de lenguaje grandes (GPT, Gemma, Llama-3)

Los modelos de lenguaje grandes (en inglés, *Large Language Models* o LLM) son modelos de aprendizaje profundo basados en redes neuronales con miles de millones de parámetros, que utilizan la arquitectura de transformers. Estos modelos pueden ser entrenados con grandes cantidades de texto, perimitiéndoles realizar una gran cantidad de tareas en el ámbito del procesamiento de lenguaje natural.

Inicialmente, los LLM operaban de manera secuencial, basando sus predicciones en las distribuciones de probabilidad de las palabras dentro de un texto. Sin embargo, este enfoque tenía la limitación de no considerar el contexto más amplio en el que aparecen

las palabras, así como sus distintos significados y asociaciones. Los avances en la arquitectura de las redes neuronales y, especialmente, la aparición de los transformers, han representado una gran evolución hacia modelos de lenguaje grandes mucho más avanzados, capaces de procesar simultáneamente enormes cantidades de texto y permitiendo establecer relaciones más sólidas entre las palabras y el contexto en que aparecen.

2.4.1. GPT

2.4.2. Gemma

2.4.3. Llama-3

CAPÍTULO 3

Adaptación de modelos de lenguaje grandes (PEFT: LoRA)

3.1 Low-Rank Adaptation (LoRA

CAPÍTULO 4
Resultados experimentales

4.1 Conjunto de datos

4.2 Medidas de evaluación

4.3 Resultados experimentales

????? ????????????? ????????????? ????????????? ????????????? ?????????????

CAPÍTULO 5
Conclusions

????? ????????????? ????????????? ????????????? ????????????? ?????????????

Bibliografía

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[2] Bohdan Bilonoh and Sergii Mashtalir. Parallel multi-head dot product attention for video summarization. In *2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP)*, pages 158–162, 2020.

[3] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. Multi-head attention: Collaborate instead of concatenate, 2021.

[4] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[5] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Stanford University and University of Colorado at Boulder, third edition draft edition, 2023. Draft version.

[6] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[7] Seyedeh Leili Mirtaheri and Reza Shahbazian. *Machine Learning*. CRC Press, 2022.

[8] Tom Mitchell. *Machine Learningl*. McGraw-Hil, 1997.

[9] Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts; London, England, 2022.

[10] Claude Sammut and Geoffrey I. Webb, editors. *Samuel's Checkers Player*, pages 881–881. Springer US, Boston, MA, 2010.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2023.

APÉNDICE A

Configuració del sistema

????? ????????????? ????????????? ????????????? ????????????? ?????????????

A.1 Fase d'inicialització

????? ????????????? ????????????? ????????????? ????????????? ?????????????

A.2 Identificació de dispositius

????? ????????????? ????????????? ????????????? ????????????? ?????????????

APÉNDICE B
??? ????????????? ????

????? ????????????? ????????????? ????????????? ????????????? ?????????????