UNIVERSITAT DE BARCELONA

Treball final de grau

DOBLE GRAU DE MATEMÀTIQUES I ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Transformers: Arquitectura i Comprensió Semàntica en Música

Autor: Héctor Balsells Roure

Director: Dr. Santiago Seguí Mesquida
Realitzat a: Departament Matemàtiques i Informàtica

Barcelona, 15 de gener de 2025

Abstract

Transformers are a new neural network architecture that has revolutionized the world of Artificial Intelligence. Chatbots such as OpenAI's ChatGPT or Google's Gemini have become part of the daily lives of many people. In this document we explore in detail how they work, and if they can be used to understand and extract qualities of song lyrics.

Resum

Els Transformers són una nova arquitectura de xarxa neuronal que han revolucionat el món de la Intel·ligència Artificial. Xatbots com ChatGPT d'OpenAI o Gemini de Google han passat a formar part del dia a dia de molta gent. En aquest treball explorem en detall com funcionen, i si poden ser utilitzats per entendre i extreure qualitats de lletres de cançons.

2020 Mathematics Subject Classification. 68T01, 68T50

Agraïments

Vull agrair a la meva família, als meus amics i a la Maria. Gràcies.

Índex

1 Introducció

El projecte

ChatGPT, Gemini i tots els altres xatbots que han sorgit i s'han fet populars aquests últims anys es basen en una arquitectura de Xarxa Neuronal innovadora anomenada Transformer (o transformador). Aquesta arquitectura, primerament proposada l'any 2017 en l'ara famós paper *Attention is all you need* [Vas+23], originalment estava enfocada per a tasques de traducció, tot i que ràpidament es van descobrir moltes noves aplicacions de l'arquitectura en altres camps, amb unes modificacions mínimes.

En aquest treball comentarem com funcionen els xatbots com ChatGPT i Gemini fixant-nos en l'arquitectura al seu darrere: Els Transformers. Explorem com transformen el text d'entrada, primer a números i aquests números a vectors, com processen aquests vectors amb tècniques pròpies d'àlgebra lineal, i com aconsegueixen generar text a partir d'aquesta entrada.

L'ús en massa d'aquest tipus de tecnologia per la societat general planteja problemes de seguretat. Al ser una arquitectura de xarxa neuronal entrenada amb dades generades per humans, és possible que aquests nous models no estiguin lliures de biaixos (per exemple, aquests models podrien presentar sexisme a causa del possible sexisme que presenten les dades d'entrenament), o podrien facilitar informació il·legal als usuaris d'aquestes eines. És per això que també comentarem les tècniques utilitzades per entrenar aquestes xarxes neuronals per aconseguir no només que siguin el més correctes possibles, sinó que també siguin segures.

Finalment, com a cas d'ús intentem donar resposta a la pregunta de si aquests models són capaços d'extreure qualitats de lletres de cançons que s'alineen amb el nostre criteri humà, i si són lliures de biaixos per llengua. Un altre cop, les dades utilitzades per entrenar el model en cada llengua són creades per humans, i per tant és possible que en generar text en una llengua concreta, el model mostri biaixos propis de la societat que parla aquell idioma en concret. Establim una metodologia per determinar aquestes preguntes, i finalment demostrem algunes aplicacions d'aquestes eines.

Estructura de la Memòria

El treball està dividit en dues parts: Una primera part teòrica on s'explica el funcionament i entrenament dels Transformers, i una segona part més pràctica on s'explora si eines com ChatGPT o Gemini són útils a l'hora d'entendre lletres de cançons.

En la part teòrica, primer s'explica en detall tots els passos que segueix un Transformer: Com a partir d'un text d'entrada és capaç de generar una resposta. Primerament, s'ensenya com el Transformer converteix el text d'entrada a vectors que codifiquen el contingut semàntic d'aquest text. Després, es demostra com es processen aquests vectors per incorporar informació del seu context utilitzant la

mecànica d'atenció, i finalment com s'utilitza aquest processament per generar text a partir del text d'entrada. Tot seguit, s'explica el procés d'entrenament d'aquesta xarxa, des de descens del gradient per modificar els paràmetres de la xarxa per a obtenir bons resultats, fins a tècniques de regularització i de Reinforcement Learning. També es presenta codi en Python que implementa aquesta arquitectura, demostrant com es pot entrenar per generar lletres de cançons.

En la part del cas d'ús, primer explorem quina entrada donar als xatbots per a poder obtenir de manera efectiva la sortida desitjada. Després, es fa una anàlisi de la utilitat d'aquestes eines per respondre les preguntes que ens plantegem sobre la seva capacitat d'anàlisi semàntic, comparant amb criteris humans i a més a més comprovant si estan lliures de biaixos segons llengua d'entrada. Per acabar, es mostren dues possibles aplicacions d'aquests models en aquest context.

2 Transformers i Xares Neuronals

2.1 Transformers

El camp d'aprenentatge automàtic compren tots els algorismes que aprenen a partir de noves dades, és a dir, que utilitzen dades anteriorment vistes per donar una millor sortida en noves dades. Un algorisme molt utilitzat dins d'aquest camp és el de xarxes neuronals FeedForward (veure secció 2.1.6), que es tracta d'un model que conté un conjunt de paràmetres entrenables que es modifiquen per a obtenir les sortides desitjades. A partir d'aquesta idea inicial, apareixen els Transformers, que incorporen nous mecanismes a les xarxes neuronals FeedForward per a obtenir una sortida més precisa.

Aquesta arquitectura de xarxa neuronal ha permès la creació dels xatbots com ara ChatGPT o Gemini. En el paper original [Vas+23] es presenta una arquitectura coneguda com a "encoder-decoder Transformer". Tot i això, els xatbots actuals com ChatGPT d'OpenAI, o Gemini de Google, els dos models que estudiarem durant aquest treball, funcionen utilitzant una versió lleugerament modificada coneguda com a "decoder-only Transformer". Primerament presentada en el paper del GPT-1 *Improving Language Understanding by Generative Pre-Training* [Rad+18], aquesta arquitectura presenta un mecanisme d'atenció conegut com a "self-attention", on un token només atén tokens anteriors (a diferència del mecanisme d'atenció del "encoder-decoder"), molt més adient per a tasques de generació de text. Per tant, d'ara endavant tractarem exclusivament aquest tipus de Transformers.

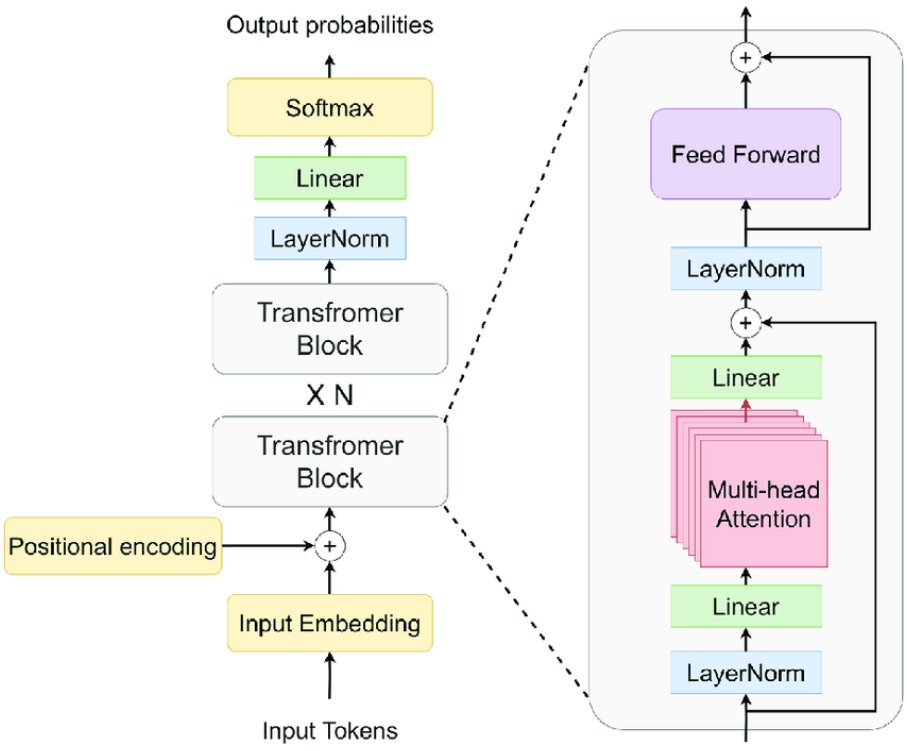


Figura 1: Estructura bàsica d'un decoder-only Transformer

En la figura 1 es presenta l'estructura bàsica d'un decoder-only transformer. Es poden distingir les següents parts: **Tokenització del input, Transformació de tokens a embeddings, codificació posicional, Multi-head Attention, normalització, FeedForward i Softmax.** Aquest procés permetrà convertir un text una seqüència de tokens d'entrada, i a partir d'aquesta obtenir una distribució de probabilitats sobre quin serà el pròxim token.

2.1.1 Tokenització del input

Per poder fer servir el Transformer, primer s'ha de fer un pas de preprocessament conegut com a tokenització. Com a input un Transformer rep una seqüència de paraules provinents d'un text, que pot contenir caràcters propis d'altres llengües o caràcters especials com ara emojis. Per poder operar amb aquest text, el Transformer primerament necessita identificar cada component que conté el text, per així poder tractar-lo individualment (on cada component és conegut com a token), i això es fa en el procés conegut com a tokenització. Tot i que la forma més directa (i conceptualment la que s'acostuma a utilitzar) de fer aquesta separació és tractar cada paraula o caràcter com a un token, a la pràctica el procés és més complex. Els algorismes tokenitzadors vam ser primerament introduïts en els models de llenguatge a la secció 2.2 del paper del GPT-2 [Rad+19], i ara tant Google com OpenAI tenen versions dels tokenitzadors que utilitzen disponibles en codi obert, i es poden consultar en els repositoris de SentencePiece [KR18] i de Tiktoken [Ope23], respectivament. Ambdós estan basats en l'algorisme byte-pair-encoding (BPE), tot i que SentencePiece també implementa l'algorisme unigram language model. Un cop identificats cada token en el text d'entrada, un número únic els hi és assignat, i a partir d'aquí ja es pot procedir a la seva transformació a embeddings (veure secció 2.1.2).

Definició 2.1.1.1. *Un vocabulari $\mathcal{V}$ és un conjunt finit de $N_{\mathcal{V}}$ elements tal que $\mathcal{V} := \{1, \ldots, N_{\mathcal{V}}\}$. Un token és un valor $t \in \mathcal{V}$. Denotem per $x \equiv x[1:l] \in \mathcal{V}^*$ una seqüència de l tokens.*

L'algorisme BPE sorgeix com a resposta al problema de la mida del vocabulari i la mida del context. Si implementéssim la versió simple on cada caràcter (o paraula) fos un token, tindríem un vocabulari molt gran (ara mateix, l'especificació Unicode té 154998 caràcters diferents), provocant que capes del Transformer com la lineal o la capa Softmax fossin molt costoses. A més a més, si es tingués tokens a nivell de caràcter, el text d'entrada estaria representat per molts tokens, fet que faria el mecanisme d'atenció menys eficient per culpa de la mida de context, com discutirem més endavant. És per això que aquest algorisme s'aplica per a comprimir el vocabulari i busca l'equilibri entre mida de vocabulari i mida de seqüències d'entrada.

Aquest algorisme, que comença amb el text tokenitzat de la forma simple (és a dir, assignant a cada caràcter un número identificatiu), es pot descriure de la següent manera:

Algorithm 1 Byte Pair Encoding (BPE)

Require: Peça de text com a llista d'enters (paraules separades per caràcters)
Require: Número d'operacions de substitució N

1: Inicialitza un vocabulari de caràcters únics del text
2: **for** $i = 1$ to N **do**
3: Compta la freqüència de cada parella adjacent de caràcters al text
4: Identifica la parella (a, b) més freqüent
5: Substitueix cada instància de (a, b) amb el símbol ab al text
6: Afegeix el nou símbol ab al vocabulari
7: **end for**
8: **return** Vocabulari actualitzat i text tokenitzat.

Com a exemple, considerem el text "$aaabdaaabac$". En un primer pas, s'identifica la parella "aa" com la més comuna, i per tant definim $Z = aa$ i obtenim el nou text "$ZabdZabac$". Podem seguir i ara veiem que la parella "ab" és la més comuna, la qual substituïm pel símbol $Y = ab$ i obtenim la seqüència "$ZYdZYac$". Ara, la parella a ser substituïda és ZY, i definim $X = ZY$, obtenint la seqüència "$XdXac$". Aquest seria un bon punt on aturar-se, ja que totes les parelles apareixen només un cop.

En el cas de la sèrie GPT (i probablement en el de Gemini, tot i que no es té el seu codi obert), es realitza un pas anterior per assegurar-se que certs caràcters mai puguin ajuntar-se en un mateix token (per exemple, caràcters de diferents paraules, o un final de paraula amb símbols com 's). Per fer-ho, es defineix un patró que serà buscat en el text d'entrada. Si en algun moment el patró coincideix amb alguna part del text, el text és separat en aquest punt. Un cop tot el text està correctament separat, cada part es processa utilitzant l'algorisme 1 de forma separada, així assegurant que no s'ajunten caràcters no desitjats. Els patrons específics usats per la sèrie de models GPT es pot trobar al repositori del TikToken.

A més a més, els tokenitzadors com TikToken incorporen la possibilitat d'afegir tokens especials que poden ser útils per a diversos motius, com per exemple per l'entrenament de la xarxa. Per exemple, la versió GPT-2 utilitza el token especial `<|endoftext|>` per indicar on acaba una peça de text, essencialment indicant a tokens posteriors que no poden atendre a cap token anterior d'aquest token especial.

Finalment, cal destacar la diferència entre la implementació d'aquest algorisme per TikToken i per SentencePiece. La principal diferència radica en quina tokenització inicial del text utilitzen cada una d'aquestes implementacions abans d'aplicar BPE. En el cas de SentencePiece, aquesta tokenització es fa fent servir directament els caràcters definits per Unicode. Com que és un vocabulari inicial molt extens, hi haurà tokens que no es veuran durant l'entrenament de BPE, i per tant aquests tokens no tindran una codificació pròpia (tècniques com el *token fallback* es poden utilitzar per mitigar aquesta desaventatja). A TikToken en canvi, la tokenització inicial es fa servint la codificació utf-8, que funciona assignant a cada caràcter entre 1 i 4 bytes per la seva representació. Per tant, el vocabulari inicial serà tan sols de 256 caràcters, ja que es processa cada byte de la codificació per separat.

2.1.2 Transformació de tokens a embeddings

Un cop acabat el procés de tokenització, s'ha aconseguit transformar una peça de text en una llista de números, on cada un representa un token del vocabulari. Tot i això, aquests números no donen cap informació sobre el text que codifiquen. És el token número 132 un verb, un nom o qualsevol altra cosa? És femení, masculí? A més a més, a causa de la mida del vocabulari, si volguéssim aplicar tècniques com ara la construcció de vectors "one-hot", on el vector és 0 en totes les posicions excepte en la posició que codifiquen, on tenen un 1 (es a dir, el vector one hot v del token i i mida de vocabulari n, seria $v_i = [\delta_{ij}]\, \forall j = 1, 2, \ldots, n$ amb $\delta_{ij} = 0$ si $j \neq i$ i $\delta_{ij} = 1$ si $j = i$) seria altament ineficient quant a memòria. És per tot això que els "vector embeddings" dels tokens són necessaris.

Un "vector embedding" o incrustació vectorial d'un token és un vector $v \in \mathbb{R}^{d_{model}}$, on d_{model} és la dimensió (sovint alta) de la incrustació, que representa amb molta més profunditat el significat del text que codifiquen, ja que el seu ús permet capturar el seu contingut semàntic, i representen una enorme reducció dimensional respecte a l'ús de vectors one-hot. Hi ha diversos algorismes que permeten extreure embeddings a partir d'una peça de text tokenitzada, com per exemple el **word2vec** [Mik+13], **BERT** [Dev+19] o el mateix algorisme dels **GPT**.

Posem com a exemple l'algorisme word2vec, desenvolupat a Google l'any 2013. Presenta dos models molt similars, "Continuous Bag-of-Words" o "CBOW", que intenta predir una paraula objectiu donat el context d'aquesta paraula i "Skip-gram", que intenta predir el context d'una paraula. Els embeddings finals són causats pels pesos de la xarxa que s'aprenen durant l'entrenament d'aquests models.

Més detalladament, el model "Skip-gram" funciona de la següent manera ("CBOW" funcionaria de manera molt similar): Per cada token d'un text, volem predir la probabilitat amb la qual un altre token aparegui en el seu context. És a dir, sigui $\mathcal{V}$ el vocabulari, i d una mida de context. Si el token $i \in \mathcal{V}$ essent estudiat apareix en el text i $Contexte_d(i)$ és el conjunt de tokens que es troben a d o menys posicions del token i, es vol predir amb quina probabilitat $j \in Contexte_d(i)$ per tot $j \in \mathcal{V}$. Per fer-ho, s'utilitza el següent model Feed-Forward:

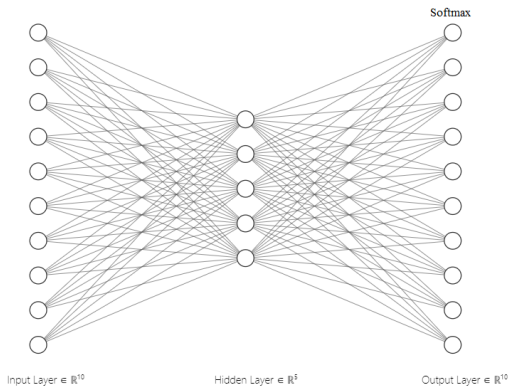


Figura 2: Estructura bàsica d'un model Skip-gram amb 10 tokens al vocabulari i dimensió embedding 5

Conceptualment, aquesta xarxa neuronal rep com a input el vector del token i codificat en one-hot, indicat com a v_i (tot i això, a la pràctica es pot implementar de forma sparse, utilitzant directament l'índex de token i per a fer-ho molt més eficient). Sigui $n =\mid \mathcal{V} \mid$ i d_{model} la dimensió desitjada del embedding, es disposa d'una capa lineal, és a dir, es multiplica el vector $v_i \in \mathbb{R}^n$ per una matriu que s'aprèn durant el procès d'entrenament, $W_1 \in \mathbb{R}^{n \times d_{model}}$, i definim $h = v_i \times W_1$ com l'estat amagat. Al resultat d'aquesta operació se li aplica una altra projecció lineal utilitzant una altra matriu apresa, $W_2 \in \mathbb{R}^{d_{model} \times n}$, obtenint així els logits $z = h \times W_2$. Finalment, s'aplica una última capa de Softmax (l'única capa no lineal). Softmax és una funció àmpliament usada per obtenir probabilitats a partir d'una llista de números. Es defineix de la següent manera:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{2.1}$$

on $i = 1, 2, \ldots, n$. Després de l'aplicació de la funció a tots els logits, obtenim com a sortida el vector $\hat{y} = (y_0, y_1, \ldots, y_n)$ on l'element j-èsim representa la probabilitat que el token j estigui en el context del token i, justament el que el model està dissenyat per a predir. Un cop entrenada aquesta xarxa, l'embedding del token i es defineix com a $e_i = v_i \times W_1$, és a dir, l'embedding és l'estat amagat del token en la xarxa.

L'entrenament d'aquesta xarxa es fa de forma supervisada, és a dir, s'utilitzen exemples dels que es coneix la solució per modificar les matrius W_1 i W_2, que contenen valors aleatoris en un principi. L'objectiu de l'entrenament és el de minimitzar la funció de pèrdua, una funció que mesura l'error entre els valors predits i els valors reals. La funció de pèrdua més utilitzada en aquest context és coneguda com la funció cross-entropy:

$$\text{L}(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{2.2}$$

Aquí, y representa el vector que conté l'etiqueta verdadera (és a dir, $y \in \text{R}^n$ és el vector amb tot zeros excepte en la posició del token que sí està en el context, on hi ha un 1) i $\hat{y}$ és el vector predit. És fàcil veure que si $y_i = \hat{y}_i$ (i suposant que 0 multiplicat pel logaritme de 0 dona 0), llavors $\text{L}(y, \hat{y}) = 0$, i es fa més gran quan més ens allunyem d'aquest resultat.

Per aconseguir minimitzar aquesta funció, s'aplica descens de gradient [BV04] al model, utilitzant el mètode conegut com a "backpropagation" [RHW86] per calcular el gradient de la funció de pèrdua respecte les matrius W_1 i W_2. Aquest algorisme utilitza repetidament la regla de la cadena per calcular el gradient respecte cada un dels paràmetres de la xarxa neuronal. En el cas del model skip-gram, la derivació és la següent:

$$\text{Tenim } L = -\sum_{i} y_i \log(\hat{y}_i) \tag{2.3}$$

$$\frac{\partial L}{\partial z_j} = \sum_i \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_j} \tag{2.4}$$

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i} \tag{2.5}$$

Utilitzant la següent definició podem representar el resultat:

$$\delta_{ij} = \begin{cases} 1, & \text{si } i = j, \\ 0, & \text{si } i \neq j. \end{cases} \tag{2.6}$$

Ho fem per casos, cas quan $i = j$:

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{e^{z_i} \cdot \sum_k e^{z_k} - e^{z_i} \cdot e^{z_i}}{(\sum_k e^{z_k})^2} = \hat{y}_i(1 - \hat{y}_i) \tag{2.7}$$

Cas quan $i \neq j$:

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{0 \cdot \sum_k e^{z_k} - e^{z_i} \cdot e^{z_j}}{(\sum_k e^{z_k})^2} = -\hat{y}_i \hat{y}_j \tag{2.8}$$

Per tant, ajuntant-ho tot obtenim:

$$\frac{\partial L}{\partial z_j} = \sum_i \left(-\frac{y_i}{\hat{y}_i} \right) \cdot (\hat{y}_i \left(\delta_{ij} - \hat{y}_j \right)) = \hat{y}_j - y_j \tag{2.9}$$

Finalment, tenim que:

$$\frac{\partial L}{\partial z} = \begin{bmatrix} \frac{\partial L}{\partial z_1} \\ \frac{\partial L}{\partial z_2} \\ \vdots \\ \frac{\partial L}{\partial z_n} \end{bmatrix} = \hat{y} - y \tag{2.10}$$

Ara, aplicant la regla de la cadena:

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial W_2} = h^T \cdot (\hat{y} - y) \tag{2.11}$$

$$\frac{\partial L}{\partial h} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial h} = (\hat{y} - y) \cdot W_2^T \tag{2.12}$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial h} \cdot \frac{\partial h}{\partial W_1} = v_i^T \cdot (\hat{y} - y) \cdot W_2^T \tag{2.13}$$

Un cop obtinguts els gradients de la funció de pèrdua respecte a tots els paràmetres de la xarxa, a cada pas d'entrenament, es poden optimitzar els pesos utilitzant

descens del gradient (sovint, es fa servir la versió estocàstica) utilitzant un hiperparàmetre ϵ (és a dir, un paràmetre escollit prèviament al procés d'entrenament) conegut com el "Learning rate" o Taxa d'aprenentatge seguint la següent fórmula:

$$W_1^{nou} = W_1^{antic} - \epsilon \cdot \frac{\partial L}{\partial W_1} \tag{2.14}$$

$$W_2^{nou} = W_2^{antic} - \epsilon \cdot \frac{\partial L}{\partial W_2} \tag{2.15}$$

L'algorisme d'aprenentatge d'aquesta xarxa neuronal quedaria, doncs, d'aquesta manera:

Algorithm 2 Entrenament del Skip-Gram

Require: Hiperparàmetres ϵ, N_{iter}
Require: Vectors d'entrada v_i, vectors context y_i, i matrius W_1, W_2 inicialitzades aleatòriament.
1: **for** $i = 1$ to N_{iter} **do**
2: Calcula els gradients $\frac{\partial L}{\partial W_1}$ i $\frac{\partial L}{\partial W_2}$
3: Actualitza els seus valors utilitzant les fórmules 2.14 i 2.15
4: **end for**
5: **return** Matrius W_1 i W_2 entrenades.

Tot i que aquest és l'algorisme bàsic, a la pràctica s'utilitzen "batches" i tècniques com el negative-sampling per fer tots els càlculs molt més eficients. Tot i que es tracta d'un model relativament simple, els embeddings obtinguts després de l'entrenament són robustos, i fins i tot es pot aplicar àlgebra amb aquests vectors. En un exemple donat pel paper original [Mik+13], l'operació $vector("King") - vector("Man") + vector("woman")$ dona com a resultat el vector que codifica la paraula "Queen".

Els models de ChatGPT (i segurament també els de Gemini, tot i que en ser completament propietari no es pot assegurar), utilitzen una capa d'embedding, similar a les capes lineals que s'utilitzen a Skip-Gram, que s'entrenen durant l'entrenament general del model i que un cop entrenades funcionen com a taules "look-up". Aquests embeddings tenen una dimensionalitat molt alta, que pot anar de 768 a més de 10000 depenen de la mida del model.

2.1.3 Codificació posicional

Després de completar l'últim pas, el Transformer ja té informació bàsica de cada token donada pel seu vector. Els Transformers estan dissenyats per processar múltiples tokens a la vegada (contràriament a altres models com ara Recurrent Neural Networks) per poder aplicar el mecanisme d'atenció. Tot i que els embeddings proporcionen molta informació, no diuen res sobre la posició del token en la seqüència d'entrada, i el Transformer no disposa d'aquesta dada crucial. És per

això que s'utilitza la codificació posicional per indicar en quina posició es troba cada token.

Hi ha diverses tècniques per fer-ho. En el paper original [Vas+23], els autors proposen l'ús d'ones sinusoidals de diferent freqüència per codificar la posició. Específicament, proposen les funcions següents:

$$PE_{(pos,2i)} = \sin(pos/1000^{2i/d_{model}}) \tag{2.16}$$

$$PE_{(pos,2i+1)} = \cos(pos/1000^{2i/d_{model}}) \tag{2.17}$$

On *pos* és la posició del embedding en la seqüència, $2i$, $2i+1$ són les dimensions del embedding parelles i senars, i d_{model} és la dimensió del embeddings. Per aplicar aquesta codificació als vectors d'entrada, simplement se sumen el vector i el vector corresponent a la seva posició, és a dir, si v és el vector en tercera posició, el vector codificat serà $u = v + \big(PE_{(3,0)}, PE_{(3,1)}, \ldots, PE_{(3,d_{model})}\big)$. D'aquesta manera s'aconsegueix que cada vector de l'entrada tingui un vector de posició diferent, i permet al Transformer distingir entre diferents posicions absolutes. També és important notar que aquesta modificació al vector no canvia massa el significat semàntic que codificava, ja que en ser un vector d'un espai amb dimensió molt alta, sumar un vector posicional mourà el vector molt poc en un espai tan gran. Els autors també indiquen que aquesta codificació ajuda al Transformer a entendre posicions relatives, pel fet que donat un desplaçament k fixat, PE_{pos+k} es pot representar com una funció lineal de PE_{pos}, ja que:

$$\begin{aligned}\sin\left(\frac{\text{pos}+k}{10000^{2i/d_{\text{model}}}}\right) = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)\cos\left(\frac{k}{10000^{2i/d_{\text{model}}}}\right) + \\ + \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)\sin\left(\frac{k}{10000^{2i/d_{\text{model}}}}\right)\end{aligned} \tag{2.18}$$

$$\begin{aligned}\cos\left(\frac{\text{pos}+k}{10000^{2i/d_{\text{model}}}}\right) = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)\cos\left(\frac{k}{10000^{2i/d_{\text{model}}}}\right) - \\ - \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)\sin\left(\frac{k}{10000^{2i/d_{\text{model}}}}\right)\end{aligned} \tag{2.19}$$

Un altre mètode, proposat per investigadors d'OpenAI en el paper [Su+23], proposa Rotary Position Embedding(RoPE) per a millorar les capacitats del Transformer per entendre posicionament relatiu, fet que ajuda a processar texts llargs. En comptes de sumar un vector posicional als embeddings d'entrada, aquest algorisme rota els vectors de "query" i "key" (veure secció 2.1.4) depenent de la seva posició. És a dir, donat un vector x, dividim l'espai en dos subespais de dimensió $d = d_{model}/2$, on un índex i d'aquest nou espai representa una parella de l'original, i després se li aplica una rotació de θ_{pos} graus a cada parella amb la matriu de rotació 2D:

$$\text{RoPE}_{pos,i}(x) = \begin{bmatrix} \cos(pos \cdot \theta_i) & -\sin(pos \cdot \theta_i) \\ \sin(pos \cdot \theta_i) & \cos(pos \cdot \theta_i) \end{bmatrix} x^T \tag{2.20}$$

$$\theta_i = 10000^{-2(i-1)/d} \tag{2.21}$$

Això permet fer que el producte vectorial de dos vectors rotats d'aquesta manera (i tenint en compte que l'angle θ_i està fixat) només depengui dels dos vectors i de la seva posició relativa, que és molt rellevant en el mecanisme d'atenció.

Proposició. *(Cas 2D) Siguin x_m i x_n els vectors a les posicions m i n, llavors $\langle RoPE(x_m), RoPE(x_n)\rangle = g(x_m, x_n, m-n)$*

Demostració.

$$\begin{aligned}
\langle \text{RoPE}(x_m), \text{RoPE}(x_n)\rangle &= \text{RoPE}(x_m)^T \cdot \text{RoPE}(x_n) = \\
&= \begin{bmatrix} x_m^{(1)} & x_m^{(2)} \end{bmatrix} \begin{bmatrix} \cos(m\cdot\theta_i) & \sin(m\cdot\theta_i) \\ -\sin(m\cdot\theta_i) & \cos(m\cdot\theta_i) \end{bmatrix} \\
&\quad \times \begin{bmatrix} \cos(n\cdot\theta_i) & -\sin(n\cdot\theta_i) \\ \sin(n\cdot\theta_i) & \cos(n\cdot\theta_i) \end{bmatrix} \begin{bmatrix} x_n^{(1)} \\ x_n^{(2)} \end{bmatrix} = \\
&= \left[x_m^{(1)}\cos(m) - x_m^{(2)}\sin(m)\right]\left[x_n^{(1)}\cos(n) - x_n^{(2)}\sin(n)\right] + \\
&+ \left[x_m^{(1)}\sin(m) + x_m^{(2)}\cos(m)\right]\left[x_n^{(1)}\sin(n) + x_n^{(2)}\cos(n)\right] = \\
&= \left(x_m^{(1)}x_n^{(1)} + x_m^{(2)}x_n^{(2)}\right)\cos(m-n) + \\
&+ \left(x_m^{(1)}x_n^{(2)} - x_m^{(2)}x_n^{(1)}\right)\sin(m-n) = g(x_m, x_n, m-n)
\end{aligned} \tag{2.22}$$

□

En el cas 2D és suficient utilitzar aquesta matriu 2×2 per fer la rotació, en casos de dimensionalitat més alta (com és sempre el cas), es pot utilitzar una matriu quadrada que té aquests blocs 2×2 a la diagonal:

$$R^d_{\Theta,m} = \begin{bmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{bmatrix}$$

On $\Theta = \{\theta_i, i \in [1, 2, \ldots, d/2]\}$. És fàcil veure que utilitzant aquesta matriu, el producte escalar continua depenent només de la posició relativa i els vectors. També s'ha de tenir en compte que la multiplicació explicita amb una matriu tan dispersa és computacionalment molt poc eficient, i per això s'usen mètodes numèrics per

fer-ho més efectiu. Per tant, aquests mecanismes (i altres, com ara la codificació de posició relativa) són suficients per proporcionar al Transformer amb la informació que li permeti distingir la posició de cada token.

2.1.4 Multi-head Attention

El mecanisme d'atenció és el "cor" del Transformer, la tecnologia innovadora que ha fet començar la revolució de la intel·ligència artificial que vivim en aquests moments. En la seva interpretació més bàsica, aquest mecanisme incorpora informació del context d'un embedding a l'embedding mateix, tenint en compte quines parts del context són més rellevants per cada embedding. Aquest enriquiment a partir del context és clau i presenta una gran millora de la IA en moltes tasques, com ara la generació de text.

En aquest punt del Transformer amb dimensió dels embeddings d_{model} i mida d'entrada L, es té una matriu $M \in \mathbb{R}^{L \times d_{model}}$, on cada fila representa l'embedding (codificat posicionalment si s'utilitza codificació de posició absoluta) d'un token d'entrada. Per demostrar el funcionament del mecanisme d'atenció, primer introduïm el cas de Single-head Attention, per després generalitzar a Multi-head Attention.

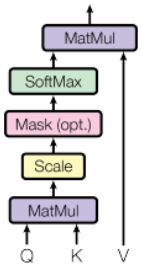


Figura 3: Procès del càlcul de l'atenció

Primerament, es calcula el vector de "query" q, el vector de "key" k i el vector de "value" v, amb $q, k \in \mathbb{R}^{d_k}$ i $v \in \mathbb{R}^{d_v}$, de cada vector d'embedding que rep el bloc d'atenció. Per fer-ho, simplement se li aplica tres capes lineals per separat (una per cada nou vector) a cada vector d'entrada. És a dir, donada la matriu d'entrada M, definim les matrius $W_q, W_k \in \mathbb{R}^{d_{model} \times d_k}$ i $W_v \in \mathbb{R}^{d_{model} \times d_v}$. Els valors d'aquestes matrius són paràmetres que s'aprenen durant el procés d'entrenament. Llavors, aconseguim els vectors desitjats amb les operacions $Q = MW_q$, $K = MW_k$ i $V = MW_v$, on cada fila d'aquestes matrius representa el vector de query, key o value corresponent. En el cas d'estar utilitzant codificació posicional RoPE, és en aquest punt on es rotarien tots els vectors de key i de query. Conceptualment, les matrius W_q i W_k són utilitzades pel Transformer per entendre quanta importància se li ha de donar a cada vector en el context de cada altre vector, és a dir, quan ha d'atendre un vector a un altre. Per fer-ho, podem construir la matriu QK^t. Normalment, aquest resultat és escalat pel factor $\frac{1}{\sqrt{d_k}}$, i finalment se li aplica la funció Softmax. Per tant, podem expressar el resultat d'aquesta operació com:

$$\text{AttentionPattern}(Q, K) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{2.23}$$

Aquí, fent abús de notació s'entén que la funció Softmax s'aplica a tots els elements de la matriu resultant del producte, fila per fila. L'aplicació d'aquesta funció és necessària per regularitzar el resultat, i aconseguir que cada columna contingui valors entre 0 i 1 que sumin 1. Per tant, el resultant és una matriu de dimensió $L \times L$, que es pot entendre com la matriu que indica quant d'important és un vector en el context d'un altre. Un valor proper a 1 a l'entrada corresponent a la matriu indica una gran importància, mentre que un valor proper a 0 indica poca importància.

També és important mencionar el perquè de l'ús del factor regularitzador $\frac{1}{\sqrt{d_k}}$. En el cas que els components de les matrius Q i K són variables aleatòries (veure definició 3.2.0.4) independents amb mitjana de zero i variància unitària ($\mathbb{E}_{q_i} = \mathbb{E}_{k_i} = 0, \mathbb{V}_{k_i}, \mathbb{V}_{k_i} = 1$), llavors al fer el producte escalar QK^T, sumem d_k terme per cada element de la matriu final. És a dir, cada element de la matriu resultant és el resultat de l'operació $\sum_{i=1}^{d_k} q_i k_i$. Això implica que la variància de la suma creix linealment amb d_k, és a dir, $\mathbb{V}\left[QK^T\right] \propto d_k$. Al escalar el resultat per $\frac{1}{\sqrt{d_k}}$, es normalitza aquesta variància. A més a més, en fer aquesta divisió, s'aconsegueix que les entrades a la funció de Softmax siguin més petites, i per tant la distribució resultant després del Softmax serà menys "punxeguda". Això permet evitar problemes durant l'entrenament, com els coneguts "vanishing gradient" o "exploding gradient", problemes molt presents en arquitectures precursores al Transformer, com les Recurrent Neuran Networks.

Una altra tècnica que s'utilitza per millorar el procés d'entrenament és coneguda com a "masking". Aquest procés evita que un token pugui atendre a tokens posteriors, que encara no han aparegut a la seqüència. D'aquesta manera, el procés d'entrenament pot utilitzar una sola seqüència d'entrada com si fossin tantes com tokens té, ja que pot intentar predir el següent token a cada posició sense que el token en qüestió "vegi" el token que està intentant predir. Per fer-ho, tots els elements de la part triangular superior del resultat $\frac{QK^T}{\sqrt{d_k}}$ es fiquen a $-\infty$ (o un valor molt petit). Així, després d'aplicar Softmax, tota la part triangular superior serà 0 i, per tant, un token no podrà atendre a un token posterior.

Un cop tenim el patró d'atenció emmascarat, podem obtenir el resultat de l'atenció aplicant la fórmula:

$$\text{Attention}(Q, K, V) = \text{AttentionPattern}(Q, K)V \tag{2.24}$$

Això donarà com a resultat una matriu de dimensió $L \times d_v$. Per transformar-la a la dimensió original del model, s'utilitza una altra matriu, coneguda com la matriu "Output" $W_O \in \mathbb{R}^{d_v \times d_{model}}$, i a cada vector se li suma el resultat d'aquesta operació per obtenir el resultat d'aquesta operació. Per tant, obtenim:

$$M_{nova} = M + \text{Attention}(Q, K, V)W_O \tag{2.25}$$

L'atenció "Multi-headed" és molt similar a la seva versió "Single-headed", però té unes quantes diferències clau que permeten aprofitar el paral·lelisme de hardware com les GPUs per accelerar enormement tots els càlculs. En comptes d'aplicar l'atenció només un cop per bloc, tenim h diferents caps ("heads") d'atenció que segueixen el mateix procés de forma concurrent, cada un amb les seves pròpies matrius apreses i el seu patró d'atenció.

En cadascun d'aquests caps, definim $d_k = d_q = d_v = d_{model}/h$ i apliquem les formules 2.23 i 2.24 de la mateixa forma. Finalment, els resultats de cada cap de l'equació 2.24 es concatenen un al costat de l'altre per formar una matriu de dimensió $L \times d_{model}$. Aquest resultat es multiplica per una matriu d'output $W_O \in \mathbb{R}^{d_{model} \times d_{model}}$, el resultat de concatenar les matrius d'output de cada cap d'atenció una sota de l'altra. Aquesta operació és equivalent a fer la projecció de d_v a d_{model} a cada cap i sumar-ho, però més eficient i compacte.

A la figura 1 s'observa que al finalitzar el procés d'atenció, se suma el resultat d'aquesta operació amb l'entrada del bloc, com també es representa a l'equació 2.25. Això és conegut com a connexió residual, i s'utilitza per mitigar impactes del vanishing gradient" o 'exploding gradient". També permet mantenir informació de capes anteriors durant el processament d'una nova, fet que pot resultar molt útil en models profunds com un Transformer.

2.1.5 Normalització

Un Transformer amb aquest mecanisme d'atenció i amb les components explicades en les següents seccions funcionaria sense problema. Tot i això, models com ChatGPT o Gemini tenen milers de milions de paràmetres a entrenar, i una gran quantitat d'exemples a la seva disposició per poder entrenar-se. Aquest procés pot arribar a ser extremadament costós, i per tant es requereixen mètodes per alleugerar i fer més ràpid l'entrenament.

LayerNorm [BKH16] és un algorisme utilitzat en aquest tipus de models que intenta millorar aquest problema. Es basa en un altre algorisme anterior anomenat BatchNorm [IS15]. LayerNorm normalitza cada fila de la matriu d'input $M \in \mathbb{R}^{L \times n_{model}}$ (recordant que cada fila representa un vector de token d'entrada). Per fer-ho, calculem per a cada fila $i \in [1, \ldots, L]$:

$$\mu_i = \frac{1}{n_{model}} \sum_{j=1}^{n_{model}} x_{i,j} \tag{2.26}$$

$$\sigma_i^2 = \frac{1}{n_{model}} \sum_{j=1}^{n_{model}} (x_{i,j} - \mu_i)^2 \tag{2.27}$$

Això ens dona la mitjana i la desviació típica de cada vector en el nostre context. Finalment, acabem la normalització aplicant aquesta fórmula a cada element:

$$\text{LayerNorm}(x_{i,j}) = \frac{x_{i,j} - \mu_{\text{i}}}{\sqrt{\sigma_{\text{i}}^2 + \epsilon}} \cdot \gamma_j + \beta_j \tag{2.28}$$

Aquí, γ_j i β_j són paràmetres que també s'aprenen durant el procés d'entrenament de la xarxa, i ϵ és una constant molt petita (sovint 10^{-5} o menys) per evitar divisions entre zero. Aquesta normalització normalment s'aplica tant a l'entrada del bloc d'atenció com a la sortida, i aconsegueix molta estabilitat en l'entrenament. Per exemple, fa que la sortida del LayerNorm sigui invariant a re-escalaments o desplaçaments dels paràmetres de la xarxa.

Proposició. *Siguin θ, θ' dos conjunts de paràmetres del model, que tenen com a matrius de pesos W i W' les quals difereixen per un factor d'escalat δ, i per un desplaçament per un vector α, és a dir, $W' = \delta W + 1\alpha$, on 1 representa un vector columna d'uns. Llavors, $LayerNorm_{\theta'}(W'x) = LayerNorm_{\theta}(Wx)$.*

Demostració. $\text{LayerNorm}_\theta(W'x) = \frac{W'x-\mu'}{\sqrt{\sigma'^2+\epsilon}} \cdot \gamma_j + \beta_j = \frac{(\delta W+1\alpha)x-\mu'}{\sqrt{\sigma'^2+\epsilon}} \cdot \gamma_j + \beta_j = \frac{Wx-\mu}{\sqrt{\sigma^2+\epsilon}} \cdot \gamma_j + \beta_j = \text{LayerNorm}_{\theta'}(Wx)$ □

Proposició. *Sigui x' una nova entrada obtinguda al escalar x per δ i w_i els pesos rellevants per l'entrada x_i. Llavors $LayerNorm(w_i^T x_i) = LayerNorm(w_i^T x_i')$*

Demostració. $\text{LayerNorm}(w_i^T x_i') = \frac{w_i^T x_i' - \mu_i}{\sqrt{\sigma_i^2+\epsilon}} \cdot \gamma_j + \beta_j = \frac{\delta w_i^T x_i - \delta\mu_i}{\sqrt{\delta^2\sigma_i^2+\epsilon}} \cdot \gamma_j + \beta_j = \text{LayerNorm}(w_i^T x_i)$
□

Com podem veure, aquesta normalització, a més a més de controlar l'esperança i la variància dels vectors en tot moment, també assegura una gran robustesa davant de re-escalaments i desplaçaments, fet que dota al model d'una gran estabilitat en l'entrenament.

2.1.6 FeedForward

En aquesta etapa del Transformer, disposem d'una matriu M on cada fila correspon a un token d'entrada. Els embeddings d'aquests tokens ja han estat actualitzats pel mecanisme d'atenció, integrant informació sobre el seu context en la seqüència. També s'han normalitzat per garantir estabilitat en els càlculs. Tot i això, fins ara només hem enriquit aquests vectors, sense extreure'n explícitament cap informació útil. És en aquest punt on entra en joc el bloc FeedForward, que s'encarrega d'analitzar i aprofitar tota aquesta informació latent.

El funcionament d'aquest bloc és més simple que el d'atenció. Donades dues matrius de pesos $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}, W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$ i dos vectors $b_1 \in \mathbb{R}^{d_{ff}}, b_2 \in \mathbb{R}^{d_{model}}$. Llavors, podem expressar la sortida d'aquest bloc com:

$$\text{FFN(M)} = M + \text{ReLU}(MW_1 + b_1)W_2 + b_2 \tag{2.29}$$

Aquí, ReLU és una funció no lineal definida com a $\text{ReLU}(x) = \max(0, x)$. Tot i ser una funció extremadament simple, és la primera (i única) no linearitat en el model, i permet expressar funcions molt complexes en combinació amb la resta del model. Tot i això, en models més moderns com les últimes versions de ChatGPT,

s'utilitza una altra funció d'activació, anomenada GELU (Gaussian Error Linear Unit), que es defineix de la següent forma:

$$\text{GELU}(x) = x \cdot \Phi(x) \tag{2.30}$$

$$\Phi(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{2.31}$$

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt \tag{2.32}$$

Per motius de complexitat computacional, normalment s'utilitza la següent aproximació:

$$\text{GELU}(x) \approx 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right) \tag{2.33}$$

Aquesta funció és molt semblant a una ReLU, però és més suau en la transició de positiu a negatiu, fet que comporta una millor derivabilitat (de fet, la funció ReLU no és derivable al 0). Això li dona millors propietats a l'hora de l'entrenament, com veurem més endavant. Podem veure la similitud de les dues funcions:

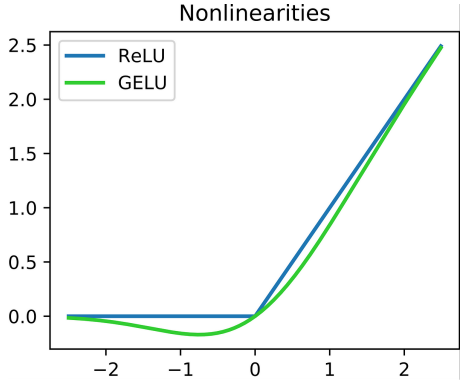


Figura 4: Gràfics per les funcions ReLU i GELU

Tot i que aquest bloc és molt més senzill que el bloc d'atenció, el comportament que emergeix en aquest FeedForward és increïblement ric. Una aparent limitació d'aquest model és que només pot representar tants atributs com dimensions té la seva capa amagada, és a dir, cada neurona s'activaria només si l'atribut que codifica està present en l'entrada. Si això fos cert, la xarxa neuronal només seria capaç de tenir en compte d_{ff} atributs, cadascun representat per una direcció ortogonal a les altres direccions en l'espai amagat.

Potser sorprenentment, hi ha papers [Elh+22] que indiquen que tot i només disposar d'un espai de dimensió d_{ff}, realment es poden representar molts més atributs seguint un procés conegut com a superposició. La idea principal és que, en un espai

de dimensió n, en comptes de codificar fins a n atributs com a direccions ortogonals, codifiquem els atributs com a direccions "quasi-ortogonals".

Definició 2.1.6.1. *Dos vectors v i w són quasi-ortogonals si i només si, donat $0 < \epsilon < 1$, es compleix que $|\langle v, w\rangle| < \epsilon$*

El següent lema demostra un resultat clau per aquests tipus de vectors.

Teorema 2.1.6.1. *(Johnson–Lindenstrauss) Donat $0 < \epsilon < 1$, un conjunt X que conté m vectors a $\mathbb{R}^N$, i un enter $n > 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \log m$. Existeix una projecció lineal $f : \mathbb{R}^N \hookrightarrow \mathbb{R}^n$ tal que*

$$(1-\epsilon)\|v_i - v_j\|^2 \leq \|f(v_i) - f(v_j)\|^2 \leq (1+\epsilon)\|v_i - v_j\|^2. \tag{2.34}$$

A més a més, $m = \exp(O(n))$

Per la demostració d'aquest teorema, necessitem un lema previ. Siguin $V_1, \ldots, V_N$ N variables aleatòries independents que segueixen una distribució $N(0,1)$, i sigui $Y = \frac{1}{\|V\|}(V_1, \ldots, V_N)$. És fàcil veure que Y és un punt escollit uniformement i aleatòriament de la superfície de l'esfera S^{N-1}. Sigui $Z \in \mathbb{R}^n$ la projecció de les n primeres coordenades de Y, i sigui $L\|Z\|^2$. Clarament, l'esperança de la llargada al quadrat de Z és $\mu = \mathbf{E}[L] = n/N$. El següent lema explica que L està concentrant al voltant de μ:

Lema 2.1.6.1. *Sigui $n < N$. Llavors*

- *a. Si $\beta \leq N$, llavors*

$$P\left[L \leq \frac{\beta n}{N}\right] \leq \exp\left(\frac{n}{2}(1 - \beta + \log\beta)\right) \tag{2.35}$$

- *b. Si $\beta > N$, llavors*

$$P\left[L \geq \frac{\beta n}{N}\right] \leq \exp\left(\frac{n}{2}(1 - \beta + \log\beta)\right) \tag{2.36}$$

La demostració d'aquest lema és tècnica i es pot trobar en el paper "An Elementary Proof of a Theorem of Johnson and Lindenstrauss" [DG03]. Continuem amb la demostració del teorema 2.1.6.1:

Demostració. Sid $N \leq n$ llavors el teorema és trivial. Si no, sigui S un subespai de dimensió n, i sigui v_i' la projecció del punt $v_i \in X$ a S. Llavors, agafant $L = \|v_i' - v_j'\|^2$ i $\mu = (n/N)\|v_i - v_j\|^2$, i aplicant el lema 2.1.6.1, obtenim:

$$\begin{aligned} P[L \leq (1-\epsilon)] &\leq \exp\left(\frac{n}{2}(1 - (1-\epsilon) + \log(1-\epsilon))\right) \\ &\leq \exp\left(\frac{n}{2}\left(\epsilon - \left(\epsilon + \frac{\epsilon^2}{2}\right)\right)\right) = \exp\left(-\frac{n\epsilon^2}{4}\right) \\ &\leq \exp(-2\log m) = 1/m^2 \end{aligned} \tag{2.37}$$

On a la segona línia hem utilitzat la desigualtat $\log(1-x) \leq -x-x^2/2$, valida per a tot $0 \leq x < 1$. Aplicant l'apartat b. del lema 2.1.6.1, la desigualtat $\log(1+x) \leq x - x^2/2 + x^3/3$ que és vàlida per qualsevol $x \geq 0$ obtenim:

$$\begin{aligned} P\left[L \leq (1-\epsilon)\right] &\leq \exp\left(\frac{n}{2}\left(1+(1-\epsilon)+\log\left(1+\epsilon\right)\right)\right) \\ &\leq \exp\left(\frac{n}{2}\left(-\epsilon+\left(\epsilon-\frac{\epsilon^2}{2}+\frac{\epsilon^3}{3}\right)\right)\right) = \exp\left(-\frac{n\left(\epsilon^2/2-\epsilon^3/3\right)}{2}\right) \\ &\leq \exp(-2\log m) = 1/m^2 \end{aligned} \tag{2.38}$$

Ara, podem definir l'aplicació $f(v_i) = (\sqrt{N/n}v_i')$. Pels càlculs fets, tenim que per una parella fixada i, j, la probabilitat que la distorsió $\|f(v_i)-f(v_j)\|^2/\|v_i-v_j\|^2$ no està a l'interval $[1-\epsilon, 1+\epsilon]$ és com a màxim $2/n^2$. Per tant, tenint en compte que tenim $\binom{m}{2}$ possibles parelles i utilitzant la cota trivial de la unió, obtenim que la probabilitat que alguna parella de punts sofreixi una distorsió gran és com a màxim $\binom{m}{2} \times 2/m^2 = 1 - 1/m$. Per tant, f té les propietats que busquem amb una probabilitat de com a mínim $1/m$. Repetint aquesta projecció $O(m)$ cops pot pujar les probabilitats d'èxit fins a la constant desitjada, i obtenim un algorisme que demostra l'enunciat del teorema. □

Per tant, aquest teorema indica que el nombre de vectors quasi-ortogonals que poden coexistir en un espai de dimensió n és de l'ordre de $exp(n)$ (ja que com $\|v_i - v_j\|^2 = \|v_i\|^2 + \|v_j\|^2 - 2\langle v_i, v_j\rangle$, el teorema també implica que el producte escalar i per tant l'angle entre dos vectors es conservarà), que en un espai en moltes dimensions és molt més significatiu que els n vectors ortogonals que podem tenir en aquest espai. En el paper [Elh+22], demostren experimentalment diverses proves que semblen implicar que efectivament les FFN aprenen aquest comportament per aconseguir representar molts més atributs amb un cost d'interferència molt baix, i que efectivament aquestes xarxes emulen xarxes neuronals de moltes més dimensions, projectant els vectors ortogonals de la xarxa amb més dimensions en vectors quasi-ortogonals de la xarxa implementada.

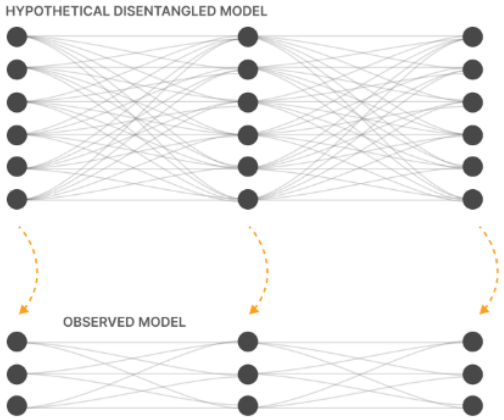


Figura 5: Projecció de una xarxa a una altra amb menys dimensions.

Aquest bloc, junt amb les capes de LayerNorm i d'atenció, són repetides N vegades, on N és un paràmetre que depèn de l'arquitectura específica del Transformer, així aconseguint enriquir el màxim possible els vectors. També s'ha de tenir en compte que en aquest bloc també es té una connexió residual, tal com s'indica en l'equació 2.29.

2.1.7 Softmax

La capa final, que ens donarà la sortida del Transformer, és una capa lineal seguida per una funció Softmax. La capa lineal ve definida per una nova matriu de pesos $W_U \in \mathbb{R}^{d_{model} \times n}$, on recordem que n representa la mida del nostre vocabulari. Es pot pensar com el contrari de la matriu d'embedding descrita en seccions anteriors. Quan la xarxa no s'està entrenant, només és necessari multiplicar l'últim vector de la matriu M (és a dir, l'última fila) per obtenir els logits de predicció sobre el següent token. Ara, només fa falta passar aquests logits per la funció Softmax 2.1 per obtenir una distribució sobre la possibilitat del pròxim token. Un cop obtingut un nou token, podem aplicar el mateix procés amb el nou token també com a entrada al Transformer per així generar seqüències de llargada arbitrària.

2.1.8 Algorisme complet

Finalment, presentem l'algorisme complet per un "forward-pass" d'un Decoder-only Transformer [PH22]. Aquí θ representa el conjunt de tots els paràmetres entrenables del Transformer (per exemple, les matrius de pesos). També presentem l'algorisme d'inferència [PH22], és a dir, l'algorisme que donat una sol·licitud i un model entrenat, dona com a sortida la resposta del model. L'hiperparàmetre τ s'utilitza per seleccionar si es vol escollir sempre el pròxim token més probable ($\tau = 0$), es vol seleccionar tenint en compte les probabilitats de sortida ($\tau = 1$) o es vol fer de forma completament uniforme ($\tau = \infty$).

Algorithm 3 $\mathbf{P} \leftarrow \text{Transformer}(x|\theta)$

/* Decoder-only forward pass */

Input: $x \in \mathcal{V}^\star$, seqüencia de tokens generades per l'algorisme 1.
Output: $P \in (0,1)^{n \times L}$, on la columna t de P representa $\hat{P}_\theta(x[t+1]|x[1:t])$.
Hiperparàmetres: $N, d_{model}, d_{ff} \in \mathbb{N}$.
Parameters: θ inclou tots aquests paràmetres:
    $W_e \in \mathbb{R}^{d_{model} \times n}, W_p \in \mathbb{R}^{d_{model} \times d_{model}}$, les matrius d'embedding i de posició.
    For $l \in [N]$:
        $\mathcal{W}_l$, paràmetres d'atenció multi-head, que inclou les matrius descrites a 2.1.4,
        $\gamma_1^l, \beta_1^l, \gamma_2^l, \beta_2^l \in \mathbb{R}^{d_{model}}$, two dos conjunts de paràmetres per LayerNorm,
        $W^l_{\text{FFN1}} \in \mathbb{R}^{d_{model} \times d_{ff}}, b^l_{\text{FFN1}} \in \mathbb{R}^{d_{ff}}$,
        $W^l_{\text{FFN2}} \in \mathbb{R}^{d_{ff} \times d_{model}}, b^l_{\text{FFN2}} \in \mathbb{R}^{d_{model}}$, paràmetres de la Feed-Forward Neural Network.

$\gamma, \beta \in \mathbb{R}^{d_{model}}$, paràmetres de la normalització final.
$W_u \in \mathbb{R}^{n \times d_{model}}$, la matriu d'unembedding.
for $t \in [L]$ **do**
$e_t \leftarrow W_e[:, x[t]]$
end for
$M \leftarrow W_p \times [e_1, e_2, \ldots, e_L]$
for $l = 1, 2, \ldots, L$ **do**
for $t \in [L]$ **do**
$\tilde{M}[:, t] \leftarrow \text{LayerNorm}(M[:, t] \mid \gamma_1^l, \beta_1^l)$
$M \leftarrow M + \text{MHAttention}(\tilde{M}^l \mathcal{W}_l, \text{Mask}[t, t'] = [[t \leq t']])$
$\tilde{M}[:, t] \leftarrow \text{LayerNorm}(M[:, t] \mid \gamma_2^l, \beta_2^l)$
$M \leftarrow M + W^l_{\text{FFN2}} \text{GELU}(W^l_{\text{FFN1}} \tilde{M} + b^l_{\text{FFN1}} 1^T) + b^l_{\text{FFN2}} 1^T$
end for
end for
for $t \in [L]$ **do**
$M[:, t] \leftarrow \text{LayerNorm}(M[:, t] \mid \gamma, \beta)$
end for
return $P = \text{softmax}(W_u M)$

Algorithm 4 $\mathbf{y} \leftarrow \text{Inference}(x, \hat{\theta})$

/* Sol·licitant al model i utilitzant-ho per predicció. */

Input: Paràmetres entrenats $\hat{\theta}$.
Input: *sol*, una sol·licitud.
Output: $y \in \mathcal{V}^\star$, la continuació de la sol·licitud.
Hiperparàmetres: $\ell_{\text{gen}} \in \mathbb{N}, \tau \in (0, \infty)$
$x \leftarrow \text{BPE}(sol)$
for $i = 1, 2, \ldots, \ell_{\text{gen}}$ **do**
$P \leftarrow \text{Transformer}(x \mid \hat{\theta})$
$p \leftarrow P[:, L + i - 1]$
Selecciona un token y de $q \propto p^{1/\tau}$
$x \leftarrow [x, y]$
end for
return $y = x[\ell + 1 : \ell + \ell_{\text{gen}}]$

Observem com, en l'algorisme d'inferència, es van seleccionant el següent token predit donat una entrada, i aquest nou token s'ajunta amb la resta de tokens d'entrada per formar l'entrada de la nova passada. D'aquesta manera obtenim una seqüència de tokens que es pot interpretar com una frase, completament generada a partir de la distribució de probabilitats que dona el Transformer.

2.1.9 ChatGPT i Gemini

Fins ara s'ha descrit una arquitectura de Transformer general tal com estan descrites en el paper original [Vas+23], apuntant quan les arquitectures modernes com ChatGPT o Gemini hi difereixen. Però una gran raó per la qual aquests models moderns són tan poderosos ve donada per la gran quantitat de paràmetres que contenen. En el cas de la versió GPT-3 de ChatGPT, aquí es resumeixen quins paràmetres han utilitzat, que donen un total de cent setanta-cinc mil milions de paràmetres entrenables [Bro+20]:

Dimension	**GPT-3**
d_{model}	12288
Número de caps d'atenció	96
Número de blocs d'atenció	96
Dimensió de la capa amagada	49152
d_{key}	128
d_{query}	128
d_{value}	128
Mida del vocabulari	50257
Mida del context	2048

Taula 1: Dimensions del Transformer per a GPT3

Desafortunadament, els paràmetres utilitzats per GPT4 o Gemini no han sigut fets públics. A més a més del disseny de l'arquitectura, és aquesta gran quantitat de paràmetres el que permet a ChatGPT i Gemini funcionar correctament. En el paper "Scaling Laws for Neural Language Models" [Kap+20] s'investiga com el rendiment d'aquests models escala segons tres factors: la mida del model (quants paràmetres té), la mida de les dades en les quals s'entrena, i el poder computacional del qual es disposa. Els autors troben que el rendiment (mesurat amb una funció de pèrdua cross-entropy, de la qual en parlarem en la següent secció) segueix una relació exponencial predictible amb aquestes variables. Específicament, sent L la funció de pèrdua, obtenim de forma aproximada:

$$L \propto N^{-\alpha} + D^{-\beta} + C^{-\gamma} \tag{2.39}$$

On N és el nombre de paràmetres, D és la mida de les dades d'entrenament, C el poder computacional, i α, β, γ són exponents escaladors que s'han determinat de forma empírica, amb un valor aproximat de $\alpha \approx 0.076$, $\beta \approx 0.095$ i $\gamma \approx 0.050$. Demostren com augmentar la mida del model millora el rendiment sempre que es disposi d'una mida de les dades suficients per evitar "over-fitting" (veure 2.2.2).

La conclusió d'aquest paper és que un augment en aquests tres paràmetres (sempre buscant l'equilibri entre ells) és molt més important pel rendiment general del model que no pas determinar hiperparàmetres com ara quantes capes tindrà el nostre model, o quantes dimensions tindrà la capa amagada del FFNN.

2.2 Entrenament del Transformer

2.2.1 Entrenament Supervisat

Com s'ha explicat en l'anterior secció, un decoder-only Transformer dona com a sortida una distribució de probabilitats per a què, donada una seqüència de tokens d'entrada, predir quin serà el pròxim token, i l'arquitectura possibilita generar text de forma coherent. Fins ara s'ha donat per suposat que el model disposa d'un conjunt de paràmetres θ ja entrenats. El procés per aconseguir aquests paràmetres és conegut com a "l'entrenament" del model.

En general, es parla de tres mètodes per entrenar un model: **Entrenament supervisat, entrenament no supervisat, i reinforcement learning (RL).** La majoría del entrenament del Transformer es fa seguint entrenament supervisat. Aquest mètode d'entrenament es caracteritza per la manera com el model aprèn a partir d'un conjunt de dades (sovint conegut com el data set) etiquetades, és a dir, un conjunt de dades d'entrada que també conté la solució desitjada. El model doncs aprèn a partir d'aquestes dades a predir quina és la solució desitjada donat un conjunt de dades que no tenen l'etiqueta.

En el cas del decoder-only Transformer, recordem que donada una seqüència de tokens d'entrada $x = [x_1, x_2, \ldots, x_L]$, el model prediu la probabilitat del pròxim token t, és a dir, si tenim un vocabulari $\mathcal{V}$, dona com a sortida $P(x_t = v \mid x_1, x_2, \ldots, x_{t-1})$ per a tot $v \in \mathcal{V}$. Després, es defineix una funció de pèrdua (Loss). Aquesta funció mesura la diferència entre la sortida donada pel model i la sortida correcta, i per tant es vol minimitzar la funció. És per això que aquest tipus d'entrenament és supervisat, ja que necessitem un conjunt de dades amb el resultat correcte per poder calcular la funció de pèrdua. La funció de pèrdua més utilitzada en Transformers és coneguda com across-entropy, i el problema d'entrenament de la xarxa es redueix a un problema d'optimització per trobar el mínim d'aquesta funció:

$$\text{Loss}(p_\theta) = -\frac{1}{L}\sum_{t=1}^{L}\sum_{i=1}^{|\mathcal{V}|} y_i \log\left[p_{t,\theta}\left(x_{t+1} = v_i \mid x_{\leq t}\right)\right] \tag{2.40}$$

Aquí, $y \in \mathbb{R}^{|\mathcal{V}|}$ és un vector que conté la probabilitat correcta extreta des del conjunt de dades amb etiquetes, i recordem que θ és el conjunt de tots els paràmetres entrenables de la xarxa. Usualment, el vector y es defineix com un vector "one-hot" que només conté un 1 en la posició del token correcte, que es sap que va després per l'etiqueta. Tot i això, a vegades s'utilitza el vector $y' = (1-\epsilon)\cdot y_i + \frac{\epsilon}{|\mathcal{V}|}$ [Kum24] per substituir el vector "one-hot" on ϵ és un número positiu petit. Això és conegut com a "Label smoothing", per millorar problemes d'overfitting (veure 2.2.2).

Per minimitzar aquesta funció, es realitzen els següents passos de forma repetida:

1. **Forward pass:** Primer de tot, es fa un forward-pass utilitzant l'algorisme 3. D'aquesta forma s'obté la distribució de probabilitats predita. Cal destacar la utilitat de la màscara descrita anteriorment en aquest pas: Com que els

tokens no són capaços d'atendre a tokens que venen després, amb una única seqüència es poden obtenir tantes seqüències d'entrenament com llarga sigui la seqüència original, ja que podem utilitzar tots els tokens anteriors per intentar predir, a cada pas, el següent. Per tant, amb una única seqüència de tokens $x = [x_1, x_2, \ldots, x_L]$, obtindrem com a sortida un vector de distribució de probabilitats $p_\theta = [p_1, p_2, \ldots, p_L]$, on l'element p_i d'aquest vector representa la distribució de probabilitats de quin serà el token $i + 1$.

2. **Càlcul de la pèrdua:** Ara, ja podem utilitzar el vector p per calcular la pèrdua amb la fórmula 2.40. És fàcil veure que aquesta funció és mínima quan p_i coincideix amb y_i, és a dir, quan la probabilitat predita és la mateixa que la probabilitat real, obtinguda des de les etiquetes.

3. **Backpropagation:** Per tant, es vol obtenir els paràmetres θ que fan mínima la funció de pèrdua, ja que llavors obtindrem una distribució de probabilitats de sortida que s'assembla el màxim possible a la sortida verdadera. Per ferho, s'aplica descens del gradient, i per calcular el gradient de la funció Loss respecte tots els paràmetres θ, s'utilitza l'algorisme conegut com a "backpropagation". Com s'ha explicat breument en la secció 2.1.2, aquest algorisme consisteix en aplicar repetidament la regla de la cadena per obtenir els gradients de la funció respecte els diferents paràmetres de la xarxa. En el cas del Transformer, backpropagation s'aplicaria de forma molt similar a la descrita a 2.1.2 per skip-gram, però tenint en compte l'estructura més complexa de la xarxa.

4. **Descens de gradient:** Un cop obtinguts els gradients de la funció de pèrdua respecte tots els paràmetres, finalment podem aplicar descens del gradient. La idea bàsica darrere descens del gradient és moure els paràmetres una quantitat petita en la direcció contrària del gradient en l'espai on cada paràmetre diferent representa una dimensió. Conceptualment, el gradient indica la direcció en aquest espai en la qual la funció creix més ràpidament. Per tant, si es canvien els paràmetres en la direcció contrària del gradient, els paràmetres seguiran el "camí" que fa que la funció de pèrdua decreixi tan ràpidament com es pugui. L'hiperparàmetre $\eta \in \mathbb{R}_+$, conegut com el learning rate, s'encarrega de controlar quant de grans són aquests canvis. Per tant, es canvien els paràmetres seguint aquesta norma:

$$\theta_{t+1} \leftarrow \theta_t - \eta \cdot \nabla \text{loss}(\theta_t) \tag{2.41}$$

A més a més, tenim que sota algunes condicions de la funció de pèrdua (funció convexa i gradient Liptschitz-continu) en podem garantir la convergència a un mínim local (que al ser la funció convexa, també serà un mínim global).

Definició 2.2.1.1. *Sigui $f : \mathbb{R}^n \to \mathbb{R}$ una funció. Diem que el seu gradient és Liptschitz-continu amb una constant $L > 0$ si i només si $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$ per qualsevol $x, y \in \mathbb{R}^n$*

Teorema 2.2.1.1. *(Convergència del descens del gradient) Sigui $f : \mathbb{R}^n \to \mathbb{R}$ una funció convexa i diferenciable, i que té gradient és Liptschitz-continu amb constant $L > 0$. Llavors, si apliquem descens del gradient per k iteracions amb un learning rate fixat $\eta \leq 1/L$, s'obtindrà una solució $f^{(k)}$ que satisfà:*

$$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|_2^2}{2k\eta} \tag{2.42}$$

on $f(x^*)$ és el valor òptim i $x^{(0)}$ és el valor inicial. Intuïtivament, això significa que descens del gradient convergeix sempre, amb una velocitat de $O(1/k)$.

La demostració d'aquest teorema es pot trobar completa a [Tib13].

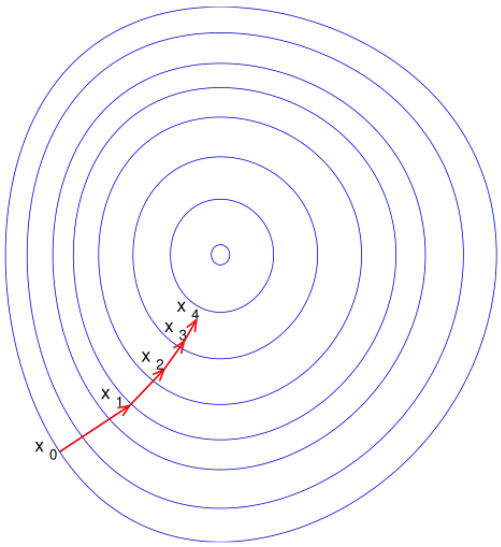


Figura 6: Descens del gradient en un conjunt de nivells.

Tot i que el descens del gradient és molt útil, acostuma a ser molt lent i costós de calcular, especialment per models molt grans com ChatGPT o Gemini. És per això que s'han proposat nous mètodes que són més eficients a l'hora de trobar el mínim de la funció. El més utilitzat és l'Adam (Adaptive Moment Estimation) [KB17]. Aquest algorisme adapta el learning rate de forma dinàmica per convergir encara més ràpid, i també soluciona el problema del tradicional descens del gradient on, si el learning rate és massa gran, es "salta" el mínim. Per fer-ho, té en compte el primer moment m_t per mesurar la quantitat de moviment (inspirat pel terme físic) i així mantenir una memòria dels gradients passats, i el segon moment v_t, que mesura la variabilitat i s'utilitza per fer actualitzacions petites a paràmetres amb un gradient molt gran i actualitzacions més agressives per paràmetres amb gradient petit. Com que aquests vectors s'inicialitzen a 0, tenen un biaix inicial cap al 0. És per això que l'algorisme incorpora una correcció d'aquest biaix. L'algorisme d'Adam és el següent:

Algorithm 5 $\theta \leftarrow \text{Adam}(f(\theta))$

Input: η: Learning rate
Input: $\beta_1, \beta_2 \in [0, 1)$: Velocitats de decreixement exponencial per a les estimacions dels moments
Input: $f(\theta)$: Funció objectiu amb paràmetres θ
Input: θ_0: Conjunt de paràmetres inicial
Input: ϵ: Constant petita per evitar divisions per zero

$m_0 \leftarrow 0$ (Inicialitzar el vector del primer moment)
$v_0 \leftarrow 0$ (Inicialitzar el vector del segon moment)
$t \leftarrow 0$ (Inicialitzar el temps)
while θ_t no ha convergit **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \text{Backprop(f}(\theta))$ (Calcular gradients)
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Actualitzar l'estimació del primer moment amb biaix
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Actualitzar l'estimació del segon moment amb biaix)
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Calcular l'estimació del primer moment sense biaix)
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Calcular l'estimació del segon moment sense biaix)
    $\theta_t \leftarrow \theta_{t-1} - \eta \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Actualitzar paràmetres)
end while
return θ_t (Paràmetres resultants)

Finalment, aquest seria l'algorisme complet per entrenar un decoder-only Transformer, utilitzant descens de gradient com a optimitzador (si es volgués usar Adam, s'hauria de substituir l'actualització dels paràmetres i seguir l'algorisme anterior):

Algorithm 6 $\hat{\theta} \leftarrow \text{Training}(x_{1:N_{\text{data}}}, \theta)$

/* Entrenant la predicció del següent token */
Input: $\{x_n\}_{n=1}^{N_{\text{dades}}}$, un conjunt de dades d'entrenament.
Input: θ, paràmetres inicials.
Output: $\hat{\theta}$, paràmetres entrenats.
Hiperparàmetres: $N_{\text{èpoques}} \in \mathbb{N}, \eta \in (0, \infty)$

for $i = 1, 2, \ldots, N_{\text{èpoques}}$ **do**
    for $n = 1, 2, \ldots, N_{\text{dades}}$ **do**
        $P(\theta) \leftarrow \text{Transformer}(x_n \mid \theta)$
        $\text{loss}(\theta) = -\frac{1}{L} \sum_{t=1}^{L} \sum_{i=1}^{|\mathcal{V}|} y_i \log \left[p_{t,\theta} \left(x_{t+1} = v_i \mid x_{\leq t} \right) \right]$
        $\nabla \text{loss}(\theta) \leftarrow \text{Backprop}(\text{loss}(\theta))$
        $\theta \leftarrow \theta - \eta \cdot \nabla \text{loss}(\theta)$
    end for
end for
return $\hat{\theta} = \theta$

En el cas de GPT-3 [Bro+20], s'utilitza l'algorisme Adam amb $\beta_1 = 0,9$, $\beta_2 =$

$0,95$ i $\eta = 10^{-8}$. També, per millorar l'eficiència de l'entrenament, de vegades a cada pas d'optimització s'utilitza un subconjunt (batch) aleatori de les dades totals d'entrenament per fer els càlculs més ràpids, la mida del qual creix gradualment durant l'entrenament. Aquest subconjunt s'escull sense repetició fins a arribar un nombre d'èpoques específic, a partir del qual ja es tenen en compte repeticions.

2.2.2 Regularització i estabilitat numèrica

Tot l'explicat a l'anterior secció cobreix l'entrenament general d'un Transformer, però aquest tipus d'entrenament encara presenta alguns problemes d'estabilitat numèrica. Un problema comú durant l'entrenament dels Transformers és conegut com a "exploding gradient", quan el gradient de la funció de pèrdua és massa gran. Per evitar aquest problema, a GPT-3 s'aplica una tècnica coneguda Global Gradient clipping. El global gradient indica quant de gran és el gradient i es calcula com la norma L^2 del vector que conté els gradients respecte els paràmetres de la xarxa:

$$\text{global\_norm} = \sqrt{\sum_i \|\nabla_i \text{Loss}\|^2} \tag{2.43}$$

El vector de gradients es reescala abans d'actualitzar els paràmetres de tal forma que aquesta norma no arriba a un màxim establert (en el cas de GPT-3, aquest màxim és 1):

$$\nabla_i \text{Loss} \leftarrow \frac{\nabla_i \text{Loss}}{\text{global\_norm}} \times \text{màxim} \tag{2.44}$$

D'aquesta forma s'aconsegueix evitar el "exploding gradient". Com hem discutit breument anteriorment, si el learning rate és massa gran, pot passar per sobre del mínim que busca l'algorisme de descens del gradient, i no trobar-lo. És per això que és preferible tenir un learning rate gran al principi de l'entrenament per millorar la funció de pèrdua ràpidament, i al final que sigui més petit, per no "saltar-se" la solució òptima. Per fer-ho, s'utilitza una tècnica coneguda com a Cosine Learning Rate Decay [LH17]. Aquesta tècnica consisteix en variar el learning rate seguint la fórmula:

$$\eta_t = \frac{1}{2}(\eta_{max} + \eta_{min}) + \frac{1}{2}(\eta_{max} - \eta_{min}) \cos\left(\frac{t\pi}{T}\right) \tag{2.45}$$

On t és el pas d'entrenament actual i T és el nombre de passos d'entrenament totals. D'aquesta forma s'aconsegueix un learning rate que varia en el temps de la forma desitjada. En el cas de GPT-3, $\eta_{min} = 0.1\eta_{max}$. Tot i això, de vegades en comptes de començar amb $\eta = \eta_{max}$, es comença amb un cert η_{ini} que s'incrementa fins al màxim seguin un procés conegut com a LR warmup. Per fer-ho, se segueix la següent fórmula:

$$\eta_t = \eta_{ini} + t\frac{\eta_{max} - \eta_{ini}}{T_{warmup}} \tag{2.46}$$

En el cas de GPT-3, aquest procès es fa durant els primers 375 milions de tokens d'entrenament.

Un altre problema comú és conegut com a "overfitting" (o sobreajustament). Aquest problema succeeix quan un model és entrenat de manera que aprèn tan bé les dades d'entrenament que després no és capaç de generalitzar correctament. Més precisament, "overfitting" succeeix quan el model aprèn de les dades d'entrenament, incloent-hi el soroll i les dades poc comunes, fins a un punt que en dades noves que no ha vist anteriorment actua de forma errònia.

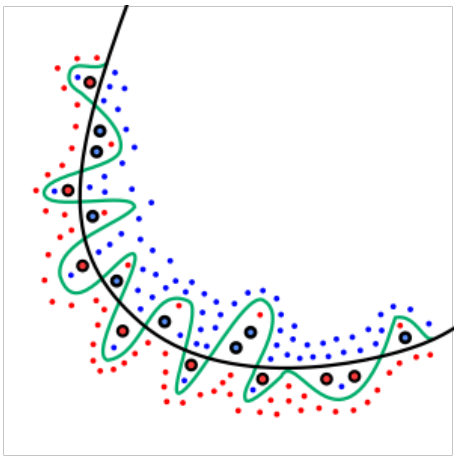


Figura 7: La línea verda representa un model amb overfitting, i la negra un model regularitzat en una tasca de classificació.

Per evitar aquest problema, s'utilitzen diferents tècniques de regularització, essent les més comunes en transformes les conegudes com a **dropout** i **Regularització** L^2.

En la regularització L^2, s'afegeix un terme $\Omega(\theta) = \frac{1}{2}\|\theta\|_2^2 = \frac{1}{2}\theta\theta^T$ (entenent θ com un vector que conté tots els paràmetres de la xarxa) a la funció de pèrdua. Afegir aquesta penalització a la funció de pèrdua dona com a resultat uns pesos més propers a l'origen, i per tant fa que les actualitzacions als pesos siguin més petites per evitar "overfitting". Per veure-ho amb més detall, estudiem el gradient de la pèrdua amb el factor de regularització [GBC16]:

$$\widetilde{\text{Loss}} = \frac{\alpha}{2}\theta\theta^T + \text{Loss} \tag{2.47}$$

$$\nabla_\theta\widetilde{\text{Loss}} = \alpha\theta + \nabla_\theta\text{Loss} \tag{2.48}$$

Aquí, $\alpha > 0$ representa un hiperparàmetre de regularització. Per actualitzar els pesos fem el pas de descens de gradient:

$$\theta \leftarrow (1 - \eta\alpha)\theta - \eta\nabla_\theta \text{Loss} \tag{2.49}$$

Com podem veure, l'addició d'aquest terme penalitzador modifica el pas de descens del gradient de manera que els paràmetres s'apropen més a l'origen, ja que els en comptes de restar el gradient de la pèrdua a θ, ho restem a $(1 - \eta\alpha)$ que amb l'elecció d'hiperparàmetres adients sempre serà un valor entre 0 i 1, aconseguint així l'efecte desitjat.

En canvi, dropout és una tècnica que es pot veure com una manera molt eficient d'entrenar un conjunt molt gran de xarxes a la vegada, i que en el moment de la inferència s'utilitzaran totes a la vegada per produir el resultat desitjat. Més específicament, utilitzant dropout durant l'entrenament s'entrenen totes les subxarxes de la xarxa que volem entrenar, obtingudes de desactivar neurones que no són de sortida.

Per aconseguir-ho, per a cada paràmetre θ_i de la xarxa, es defineix una variable aleatòria $r_i \sim \text{Bernoulli}(p)$ (veure definició 3.2.0.4), on $0 \leq p \leq 1$ és un hiperparàmetre (sovint 0,1 o similar). Aquesta variable aleatòria es multiplica per la sortida de la neurona parametritzada pel paràmetre θ_i, i d'aquesta forma s'aconsegueix "desactivar" la sortida de la neurona amb una probabilitat p, i s'entrena la subxarxa resultant que no conté aquesta neurona en específic.

Emprant aquesta tècnica, no només s'aconsegueix que la sortida del model sigui menys dependent de la sortida d'una neurona en específic (i per tant solucionant part del problema d'overfitting), sinó que a més fa la xarxa més robusta davant possible soroll en l'entrada.

2.2.3 Fine Tuning

A més de tot el que s'ha discutit anteriorment, empreses com OpenAI o Google han de poder assegurar un cert grau de seguretat i certesa en les seves aplicacions. Els models que presenten al públic no contesten a sol·licituds com "Descriu el procés de creació d'una bomba". Per aconseguir-ho, els processos d'entrenament incorporen un últim pas: **Reinforcement-Learning from Human Feedback** (RLHF) o aprenentatge de reforç a partir de la retroalimentació humana [Chr+23]. Aquest pas no només assegura una major seguretat a l'ús dels models, sinó que també aconsegueix dotar-los d'una major habilitat per entendre la intenció de l'usuari.

Un cop entrenat el model, es fa un entrenament supervisat conegut com a fine-tuning, per acabar de preparar els models per contestar als usuaris de la forma usual sol·licitud-resposta. Aquest procés consisteix en escollir una mostra del conjunt de sol·licituds preparades pel model. Després, un humà escriu la sortida desitjada, i aquesta sortida s'utilitza amb un mètode tradicional de descens del gradient per acabar d'entrenar el model prèviament entrenat. En el cas que es vulgui usar un model de llenguatge per alguna tasca específica, es pot fer servir aquest procés per donar un entrenament dissenyat per la tasca en concret. Desgraciadament, és molt costós generar les etiquetes per aquest fine-tuning, ja que es necessita un conjunt de sol·licituds prou grans, i els humans han d'escriure manualment etiquetes detallades.

En canvi, RLHF utilitza retroalimentació per humans per acabar d'entrenar el model, que és molt més senzill d'obtenir que les etiquetes del fine-tuning anterior. L'estructura bàsica d'un problema de Reinforcement-Learning és el següent [Ber23]:

- **Espai d'estats:** L'espai d'estats inclou tota la informació disponible sobre la tasca en qüestió que és rellevant per a les decisions que l'agent d'IA pugui prendre, incloent-hi variables conegudes i desconegudes. Normalment, l'espai d'estats canvia amb cada decisió que pren l'agent.
- **Espai d'accions:** L'espai d'accions conté totes les decisions que l'agent d'IA podria prendre. En el context d'un joc de tauler, per exemple, l'espai d'accions és discret i ben definit: consisteix en tots els moviments legals disponibles per al jugador d'IA en un moment determinat. En el context de la generació de text, l'espai d'accions és immens, ja que inclou tot el "vocabulari" de tokens disponibles per a un model de llenguatge.
- **Funció de recompensa:** La recompensa és la mesura de l'èxit o del progrés que incentiva l'agent d'IA. En alguns casos, com els jocs de tauler, definir l'èxit —en aquest cas, guanyar el joc— és objectiu i senzill. Però quan la definició d'"èxit" és ambigua, dissenyar una funció de recompensa efectiva pot ser un gran desafiament. En un marc matemàtic, aquest feedback s'ha de traduir en un senyal de recompensa: una quantificació escalar del feedback positiu (o negatiu). L'objectiu de l'agent és maximitzar la recompensa rebuda a llarg termini.
- **Política:** Una política (policy) és, essencialment, l'estratègia o el "procés de pensament" que guia el comportament d'un agent d'IA. En termes matemàtics simples, una política ("π") és una funció que pren un estat ("s") com a entrada i retorna una acció ("a"): $\pi(s) \rightarrow a$. L'objectiu final serà optimitzar la política per obtenir una recompensa màxima.

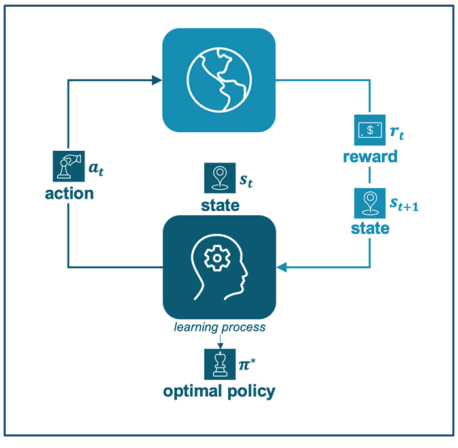


Figura 8: Reinforcement-Learning

En el cas dels models de llenguatge, podem pensar la xarxa com la política, que donat la seqüència de tokens d'entrada (estat) escull el pròxim token de la seqüència

(pren una acció). RLHF ofereix una forma molt còmoda i ràpida per dissenyar el senyal de recompensa: Tenir humans com a avaluadors als quals se'ls presenta diverses sortides possibles, i se'ls demana que determinin l'ordre segons quina és millor (i si no hi ha alguna d'inapropiada). Aquest feedback es pot utilitzar com la recompensa desitjada, i es poden actualitzar els paràmetres per maximitzar aquesta recompensa.

Tot i això, fent servir ascens del gradient (ascens ja que es busca maximitzar la recompensa) els canvis que es fan als paràmetres preentrenats són excessivament grans. És per això que s'utilitza un algorisme d'entrenament conegut com a Proximal Policy Optimization (PPO) [Sch+17]. Aquest algorisme presenta un objectiu a maximitzar alternatiu per assegurar que les actualitzacions sigui relativament petites:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{2.50}$$

Aquí, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ és una proporció que controla quant de diferents són els paràmetres un cop actualitzats respecte als paràmetres anteriors, $\epsilon \in (0, 1)$ és el valor que limita el canvi dels paràmetres, i $\hat{A}_t$ és el que es coneix com la funció d'avantatja, que estima quan de bo és prendre una acció en un estat en concret en comparació a les altres possibles accions. La part més important de l'equació és l'addició de la funció *clip*, que treu qualsevol incentiu al model de fer una actualització que provoqui la proporció r_t surti fora de l'interval $(0, 1)$, aconseguint així l'efecte desitjat.

Després d'aquest pas, el model ja està entrenat i preparat per ser utilitzat pel públic general, amb un grau de seguretat assegurat i amb el fine-tuning adequat.

2.2.4 Dades d'entrenament

Per fer l'entrenament, es necessita una gran quantitat de dades que serveixin per aplicar el procés descrit en les últimes seccions. Com ChatGPT i Gemini són models amb una gran quantitat de paràmetres, és necessari obtenir moltes dades. En el cas de GPT-3 [Bro+20], aquestes dades són principalment extretes del conjunt de dades CommonCrawl [Raf+23]. Aquestes dades són filtrades i ajuntades amb altres dades per obtenir un conjunt de dades d'entrenament de la millor qualitat possible. En la següent taula es poden veure la mida d'aquest conjunt d'entrenament:

Dataset	**Tokens**	**Presencia**
Common Crawl (filtrat)	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

Taula 2: Estadístiques del conjunt d'entrenament

2.3 Implementació d'un Transformer

Al repositori de Github https://github.com/hecmar007/SongRatings es pot trobar una implementació [Kar24] de l'arquitectura descrita fins ara a l'arxiu "SongGPT.ipynb". S'han fet algunes simplificacions en la tokenització, i s'utilitza la tokenització senzilla a nivell de caràcter. A més a més, la codificació posicional s'ha fet sumant un embedding posicional, i no seguint RoPE.

A part d'aquests petits canvis, el codi demostra una implementació de l'arquitectura del Transformer amb la llibreria `pytorch` de Python. El model presentat és un model amb multi-head attention, i està entrenat amb lletres de cançons en anglès extretes d'una pàgina de Kaggle. Per tant, el model és entrenat per generar cançons, però pot adaptar-se canviant les dades d'entrenament. Aquest model, entrenat amb els paràmetres que es poden veure en el repositori i després d'una hora d'entrenament en una màquina convencional, genera com a sortida una cançó com la següent:

```
But you ain't standin' off tha back

Life, old emberates (Never said that)

RUCH, old emberrate a bills (T-CLE Hahaha)

When you ain't tribbin' for jamas (T-C-M-O)

I can't stop that your boys ain't change, ain't change

And make emberate pop'll make emberate that your lot

(Never said, you can't change, ain't change, ain't change)

Tell me your boy your boy man just trippin' that your socked and have a little
```

Tot i que la lletra no té sentit, s'assembla bastant al lèxic anglès, deixant entreveure el poder d'aquesta arquitectura tenint en compte el poc temps d'entrenament i els pocs recursos utilitzats per produir aquest text.

3 Cas d'ús

Després de veure la detallada explicació de l'arquitectura del Transformer, explorem la possibilitat d'aquest tipus de model per acomplir certes tasques. Ens preguntem si ChatGPT o Gemini és idoni per l'anàlisi de sentiments en lletres de cançons, tant populars com de menys conegudes, i ho comparem amb criteris humans. En particular, ens interessa saber si aquestes eines són capaces de detectar si hi ha racisme, sexisme o altres temes delicats presents a la cançó, o fins i tot habilitats cognitives, com ara l'empatia que transmet la lletra. A més a més, també ens interessa investigar si aquests models tenen en compte la llengua a l'hora d'emetre aquestes valoracions. És a dir, si demanen les puntuacions d'una cançó en castellà, i després en anglès, les puntuarà de forma diferent tenint en compte que la societat castellanoparlant valora de forma diferent aquests ítems comparat amb la societat angloparlant? Establim una possible metodologia a seguir. Com a casos d'ús interessants, analitzem dues artistes fent servir aquestes eines per fàcilment comparar el contingut de les seves lletres, i finalment, desenvolupem una aplicació que pot determinar tots aquests paràmetres i afegir etiquetes a qualsevol cançó. Tot i que aquest particular projecte se centra en la lletra de les cançons, un cop les eines estiguin disponibles al públic, aquest experiment és fàcilment aplicable a videoclips de cançons, on també resultaria molt interessant. Tots els experiments en aquesta secció s'han fet utilitzant la versió 4-o de ChatGPT i la versió 1.5 Flash de Gemini.

Per fer els experiments, farem servir 5 cançons del TOP 100 cançons a Espanya i Estats Units, i 5 cançons amb 500.000 reproduccions o menys a YouTube. Un llistat complet de les cançons utilitzades es pot trobar a l'apèndix A.1.

3.1 Objectiu i sol·licituds

Primerament, definim quins són exactament els punts que volem analitzar a cada cançó. Hi ha molts possibles, però en aquest treball ens centrarem en els següents deu punts:

- **Habilitats cognitives:**
  - Empatia
  - Creativitat
  - Presa de decisions
  - Relacions interpersonals saludables
  - Resiliència
- **Sensibilitat:**
  - Racisme
  - Sexisme
  - Abús de substàncies

- Violència
- Contingut sexual

Aquests ítems poden resultar una mica ambigus en la seva interpretació (per exemple, que vol dir exactament la creativitat en aquest context?), i pot resultar interessant comprovar si la interpretació humana (i la qualificació que els doni) és similar a la interpretació dels xatbots. Un cop decidits aquests punts, entra en joc el "prompt engineering" o l'enginyeria de sol·licituds, que explora com podem demanar exactament a ChatGPT i Gemini que duguin a terme la tasca que volem completar, i com ho podem fer de la manera més òptima. Tot i que aquests models estan entrenats per poder entendre instruccions en qualsevol mena de context, hi ha algunes tècniques que es poden utilitzar per intentar millorar al màxim la qualitat de les respostes.

En el paper "Prompt Engineering with ChatGPT: A Guide for Academic Writers" [Gir23] l'autor identifica quatre parts diferenciades per fer una sol·licitud efectiva:

- **Instrucció:** Una tasca o un seguit d'instruccions que guien el comportament del model i el guia al resultat desitjat. Un exemple rellevant en el nostre cas podria ser *Qualifica si la següent cançó conté algun d'aquests temes: Empatia, creativitat, ...*
- **Context:** Informació externa o context addicional que dona coneixement extern al model, ajudant a generar respostes més rellevants. *Explora aquests temes segons la rellevància que podrien tenir en un possible nou escoltador de la cançó.*
- **Dades d'entrada:** L'entrada que volem que el model processi i que ens doni una resposta. És la part més important de la sol·licitud perquè el model entengui la tasca. *Lletra de la cançó.*
- **Indicador de sortida:** Especifica quin format volem com a sortida. *Per cadascun dels ítems, dona una puntuació del 0 al 10 per quant de present està en la lletra. Indica exactament quina o quines línies de la lletra has utilitzat per justificar la puntuació.*

El paper també identifica diferents erros que cal evitar a l'hora d'escriure una sol·licitud, com ara ambigüitat, falta de context, etc. En el nostre cas, el punt més rellevant és el de "reforçament de biaix". Aquest fenomen es dona quan es proporciona com a entrada al model una sol·licitud amb un biaix pressuposat, fals, i que pot perpetuar idees que mantenim a priori o biaixos perjudicials presents en la societat. Per exemple, s'ha d'evitar entrades com per exemple "Dona una puntuació sobre les relacions interpersonals saludables tenint en compte els llaços familiars que es mostren". Aquí, estem mostrant el biaix de valorar molt positivament les relacions familiars dins les relacions interpersonals.

Tenint tot això en compte, ja podem formular diferents models de "prompt" per indicar al model la tasca que es vol dur a terme. Tenim dos objectius principals que volem que el model ens resolgui: Donada una lletra de cançó, es vol obtenir la puntuació de cada un dels punts definits anteriorment segons quant presents a la lletra, i exactament quina part de la lletra justifica aquesta puntuació. També es podria demanar que el model etiqueti automàticament la cançó amb cadascun d'aquests ítems, però també es podrien afegir aquestes etiquetes directament de les puntuacions obtingudes. Un altre factor important a considerar és el de l'idioma. Com que models com ChatGPT i Gemini han estat entrenats principalment amb un corpus anglès, és possible que la qualitat de les respostes sigui superior si la sol·licitud també es fa en anglès. Per tant, es provarà en diferents idiomes per avaluar com és millor. En conseqüència, proposo les següents sol·licituds (les traduccions de les sol·licituds es poden trobar a l'apèndix A.2):

1. Estic explorant les lletres de diferents cançons, i m'agradaria que qualifiquessis si la següent cançó conté aquests temes: Empatia, Creativitat, Presa de decisions, Relacions interpersonals saludables, Resiliència, Racisme, Sexisme, Abús de substàncies, Violència, Contingut sexual. Explora aquests temes com si fossis un usuari que escolta la cançó per primer cop.

   Lletra de la cançó.

   Per cadascun dels temes, dona una puntuació del 0 al 10 segons quant de present i rellevant és el tema en la lletra. Per cada puntuació, indica exactament quina o quines línies de la lletra has utilitzat per justificar-la. Finalment, dona una llista com a resum de totes aquestes puntuacions. Exemple: Empatia: 5, Creativitat: 3, Presa de decisions: 6, etc.

2. Vull que analitzis la lletra següent d'una cançó. La teva tasca és determinar fins a quin punt aquesta lletra tracta els temes següents:

   Empatia, Creativitat, Presa de decisions, Relacions interpersonals saludables, Resiliència, Racisme, Sexisme, Abús de substàncies, Violència, Contingut sexual.

   Per a cada tema, assigna una puntuació del 0 al 10, on 0 significa que no hi ha cap menció o al·lusió al tema, i 10 significa que el tema és central i es tracta amb gran detall o profunditat. Després, explica breument per què has assignat aquesta puntuació, citant les línies exactes de la cançó que t'han portat a aquesta conclusió.

   Exemple de resposta esperada per a un tema:

   Resiliència (6/10): "He caigut mil vegades, però sempre m'aixeco." Aquesta línia transmet un missatge de superació personal, però no desenvolupa més aquest aspecte en altres parts de la lletra, motiu pel qual no assigno una puntuació més alta.

   Lletra de la cançó

3. Utilitzar les sol·licituds anteriors, però demanant cada ítem en el qual estem interessant separat, un per un, per així obtenir (en principi) una resposta més detallada.

Després de provar totes aquestes possibles entrades, les dues primeres donen una sortida molt similar, i per tant es poden utilitzar indiferentment. A partir d'ara preferim la primera en ser més curta. Per tant, per obtenir les dades preguntarem a ChatGPT o Gemini sobre les cançons utilitzant aquesta sol·licitud, com veurem en la següent secció i a l'apèndix A.4. Provant la tercera entrada, els models ens donen una resposta molt més detallada (10-15 línies de justificació en comptes de les 2-3 línies amb les altres entrades). Això pot resultar molt interessant si s'està interessat en analitzar profundament el raonament justificat dels models, però en el nostre cas d'ús és més interessant rebre totes les puntuacions juntes.

3.2 Avaluació de sortida

Ja hem determinat com podem estructurar l'entrada per aconseguir que el model respongui amb la informació en la qual estem interessats. Però encara no podem avaluar si la resposta donada per aquests models té qualitat comparat amb el nostre criteri com humans, o si canvia en les seves valoracions tenint en compte llengua. Per fer-ho, seguiré els següents passos:

1. **Obtenció de dades:** Obtenir dades tant qualitatives com quantitatives dels models utilitzant les sol·licituds explorades anteriorment (les puntuacions de cada categoria serien dades quantitatives, i les justificacions d'aquestes puntuacions, qualitatives). Després, obtenir dades humanes, on es demanarà que puntuïn les mateixes categories. D'aquesta forma, haurem obtingut les mateixes dades donades per humans i per models. També obtenir dades de sortida variant l'idioma de la sol·licitud per la mateixa cançó.

2. **Anàlisi preliminar:** Calcular la mitjana en les dades humanes serà útil per tenir en compte la variabilitat dels criteris humans, i tenir un únic número amb què comparar criteri humà i model. També és útil el càlcul d'aquests paràmetres per comparar entre les sortides segons els diferents idiomes. En el cas que demanéssim al model la puntuació d'una mateixa cançó diverses vegades (per minimitzar la variabilitat entre diferents sortides) també es podrien calcular aquestes mètriques.

3. **Comparació:** Podem utilitzar estimadors (veure definició 3.2.3.1) com Root Mean Squared Error per comparar les dades quantitatives del model i les humanes. Aquest estimador donarà una primera aproximació a la resposta de la pregunta de quin dels dos models s'ajusta més a criteris humans.

4. **Conclusió:** Un cop tinguem totes les mesures calculades, finalment es podrà determinar si aquests models de llenguatge són adients per extreure les categories en les quals estem interessats, i si es veuen influenciades per la llengua de la sol·licitud o lletra.

També resultarà útil tenir en compte les següents definicions estadístiques:

Definició 3.2.0.1. *Un espai de probabilitat és la tripleta $(\Omega, \mathcal{A}, P)$, on Ω, l'espai mostral, és el conjunt de tots els resultats possibles, $\mathcal{A}$, un conjunt de subconjunts de l'espai mostral, i $P : \mathcal{A} \to [0,1]$ és la funció de probabilitat que compleix les següents condicions:*

1. $P(\Omega) = 1$
2. *Donada una successió $\{A_n\}_{n\in\mathbb{N}}$ d'elements de $\mathcal{A}$ disjunts dos a dos, aleshores $P\left(\bigcup_{r=1}^{\infty} A_r\right) = \sum_{r=1}^{\infty} P\left(A_r\right)$*

Definició 3.2.0.2. *Un model estadístic paramètric és un conjunt $(\Omega, \mathcal{A}, \{P_\theta, \theta \in \Theta\}), \Theta \subseteq \mathbb{R}^d$, on el paràmetre θ és desconegut.*

Definició 3.2.0.3. *Donat un espai mostral Ω, una col·lecció $\mathcal{E} \subset \mathcal{P}(\Omega)$ de subconjunts de Ω és una σ-àlgebra si:*

- $\Omega \in \mathcal{E}$
- *Si $A \in \mathcal{E}$ llavors $A^c \in \mathcal{E}$*
- *Si $A = \{A_n, n \in \mathbb{N}\} \subset \mathcal{E}$ llavors $\bigcup_{n\in\mathbb{N}} A_n \in \mathcal{E}$*

Definició 3.2.0.4. *Una variable aleatòria $\mathbf{X}$ sobre un espai de probabilitat $(\Omega, \mathcal{A}, P)$ és una funció $\mathbf{X} : (\Omega, \mathcal{A}, P) \to (\mathbb{R}, \mathcal{B}(\mathbb{R}))$ mesurable, és a dir, $\mathbf{X}^{-1}(B) \in \mathcal{A}\, \forall B \in \mathcal{B}(\mathbb{R})$, on $\mathcal{B}(\mathbb{R})$ denota la σ-àlgebra de Borel als reals, la σ-àlgebra formada pels oberts dels reals.*

3.2.1 Obtenció de dades

Per obtenir les dades dels models, simplement utilitzem les sol·licituds determinades com a entrada amb cada cançó. Les puntuacions aconseguides es troben a les taules de la següent pàgina. És important notar que per una cançó, a causa del contingut explícit de la lletra, Gemini no ha donat sortida, dient que "No puedo abordar este tema. Puede que no sea seguro o apropiado responderte a eso. Cambiemos de tema." És per aquest motiu que les entrades corresponents a la cançó Q U E V A S H A C E R H O Y tenen un -. També és important notar que mentre que ChatGPT sempre respon amb les línies explícites de la cançó per justificar la seva qualificació (tal com es demana a l'entrada), Gemini de vegades no ho fa i simplement dona un raonament més general. Per exemple, aquestes són les cinc primeres puntuacions i justificacions per la cançó "La pasión de Beefie" dels dos models:

- **ChatGPT:** Empatía: 9/10 Líneas: "Perdónales porque no saben lo que hacen." "Perdóname al ladrón, perdona al asesino." "Perdona a las ratchets que no pueden cuidar a sus hijos." "Perdona a estas calles por tener tanta ruina." Justificación: Hay un fuerte tono de comprensión y perdón hacia las acciones de otros, reconociendo las circunstancias difíciles que los llevan a actuar de cierta manera.

Creatividad: 6/10 Líneas: La estructura de las metáforas y el juego lírico, como “Estoy enderezando la esquina con marihuana, cocaína.” La manera de relatar experiencias callejeras de forma poética y simbólica. Justificación: La creatividad está presente en cómo el artista expresa una visión compleja de su entorno, aunque no sea el tema central.

Toma de decisiones: 7/10 Líneas: “Perdóname ma’re por no poder ayudarte, estoy trabajando pa’ poder compensarte.” “Estoy enderezando la esquina con marihuana, cocaína.” Justificación: Se refleja en la lucha entre decisiones difíciles, como entrar en actividades ilícitas para sobrevivir y compensar a los seres queridos.

Relaciones interpersonales saludables: 6/10 Líneas: “Te echo de menos y tengo ganas abrazarte.” “No quiero que llores, dame una sonrisa, pronto va’ a estar mejor.” Justificación: Aunque las relaciones no son saludables en todos los casos, hay una búsqueda de conexión emocional con la madre y otros cercanos.

Resiliencia: 8/10 Líneas: “No la tomes con ellos que no saben lo que hacen.” “Tú ni te imaginas que esmaye.” “Pa’ forgive, ’cuz I’m for real, papi Yung Beef.” Justificación: La letra muestra resistencia frente a las adversidades, tanto personales como sociales, manteniendo una actitud de perdón y superación.

- **Gemini:** Tema: Empatía Puntuación: 8 Justificación: Las líneas “Perdónales porque no saben lo que hacen”, “Perdóname ma’re, por no poder ayudarte”, “No voy a dejar que nadie venga a agobiarte” y otras similares demuestran una profunda empatía hacia los demás, incluso hacia aquellos que le han hecho daño.

  Tema: Creatividad Puntuación: 6 Justificación: La canción utiliza un lenguaje muy visual y crea imágenes vívidas. Sin embargo, no hay una exploración profunda de procesos creativos.

  Tema: Toma de decisiones Puntuación: 7 Justificación: La letra habla sobre las difíciles decisiones que ha tenido que tomar en su vida, como “Estoy enderezando la esquina con marihuana, cocaína”.

  Tema: Relaciones interpersonales saludables Puntuación: 5 Justificación: Aunque hay expresiones de amor hacia su madre y sus amigos, la mayoría de las relaciones descritas son conflictivas o marcadas por la violencia.

  Tema: Resiliencia Puntuación: 9 Justificación: La canción es un testimonio de la capacidad de superar adversidades y seguir adelante a pesar de las dificultades.

Per tant, sembla ser que ChatGPT és més precís a l’hora de justificar exactament quines línies de la lletra l’ha portat a una puntuació. A continuació es troben les taules de puntuacions per les 15 cançons (on cada sol·licitud s’ha fet en l’idioma amb el qual estan escrites), i a l’apèndix B.2 també una taula amb les diferents puntuacions de la mateixa cançó en català, castellà i anglès.

Títol	Empat	Crea	P.d.D.	R. I. S.	Res	Racism	Sexism	A.d.S	Violèn	C. S.
1. DEGENERE	3	4	5	2	4	0	6	8	0	7
2. Si Antes Te Hubiera Conocido	2	3	4	1	3	0	7	0	0	8
3. Q U E V A S H A C E R H O Y	2	6	3	3	1	0	7	2	0	10
4. IA	2	6	7	1	5	0	8	2	3	7
5. GRAN VÍA	2	5	6	2	0	0	4	3	0	9
6. Sticky	1	8	3	2	7	0	5	2	8	9
7. 25	7	6	6	7	5	0	0	3	0	2
8. Love Sombody	8	5	6	7	6	0	0	3	0	4
9. tv off	2	8	7	3	9	5	6	1	8	4
10. Timeless	2	6	3	2	4	0	6	8	7	6
11. La Pasion de Beefie	9	6	7	6	8	1	4	7	6	3
12. ianomvui druga	4	3	8	6	7	0	6	7	0	7
13. Cyberpunk	4	7	6	3	2	0	1	8	2	1
14. Calgary 88	7	9	8	10	6	0	0	0	0	0
15. Heavenly	5	6	2	4	3	0	0	0	0	1

Taula 3: Puntiacions ChatGPT

Títol	Empat	Crea	P.d.D.	R. I. S.	Res	Racism	Sexism	A.d.S	Violèn	C. S.
1. DEGENERE	0	3	2	1	0	0	8	7	2	10
2. Si Antes Te Hubiera Conocido	2	3	4	2	0	0	3	0	0	8
3. Q U E V A S H A C E R H O Y	-	-	-	-	-	-	-	-	-	-
4. IA	0	0	6	0	0	0	3	1	2	5
5. GRAN VÍA	2	4	6	1	0	0	3	0	0	10
6. Sticky	0	0	0	0	0	0	10	0	8	10
7. 25	2	2	3	4	3	0	0	2	0	2

8. Love Sombody	0	0	0	3	2	0	0	3	0	1
9. tv off	0	6	3	0	5	3	1	0	7	0
10. Timeless	0	8	3	1	6	0	3	7	5	6
11. La Pasion de Beefie	8	6	7	5	9	2	4	8	8	5
12. ianomvui druga	6	5	3	3	2	0	0	8	2	5
13. Cyberpunk	3	7	6	2	4	0	0	9	3	3
14. Calgary 88	8	9	6	10	8	0	0	0	0	0
15. Heavenly	7	2	1	3	4	0	0	0	0	0

Taula 4: Puntuacions Gemini

A més a més, recollim puntuacions proporcionades per tres humans de diferents edats i sexes (per intentar reduir la variabilitat humana) per les mateixes cançons i mateixos temes, per així poder investigar quan s'adeqüen les respostes donades pels models a les donades per un humà. Per fer-ho, primer s'ha donat un exemple del tipus de puntuacions buscades, i després se'ls hi ha mostrat la lletra a puntuar, una a una. Aquestes puntuacions es poden trobar a l'apèndix B.1.

3.2.2 Anàlisi preliminar

Durant la fase d'obtenció de dades, s'han recollit les puntuacions donades per tres humans a les mateixes cançons demanades als models. En demanar-ho a tres humans, baixem la variabilitat en les respostes, i treballem amb les mitjanes d'aquestes puntuacions per fer les comparacions amb les respostes donades per ChatGPT i Gemini. La següent taula mostra aquestes mitges (l'ordre de les cançons és el mateix que l'utilitzat fins ara i l'ordre mostrar a l'apèndix).

Cançó	**Emp**	**Crea**	**P. D.**	**R. S.**	**Res**	**Rac**	**Sexis**	**A.d.S**	**Vio**	**C. S.**
1	1.67	2.00	2.67	2.33	1.33	0.00	6.00	6.33	2.33	6.33
2	2.67	2.33	3.00	3.67	2.00	0.00	3.67	0.00	0.67	4.67
3	1.33	2.67	2.33	3.00	1.67	0.00	4.67	1.00	0.67	10.00
4	0.00	2.33	2.00	0.00	0.33	0.00	8.67	0.00	5.33	6.67
5	3.67	2.67	4.00	3.00	2.33	0.00	3.33	2.67	0.00	7.67
6	0.00	3.00	1.67	0.00	2.00	7.00	9.33	9.33	9.33	9.00
7	5.67	4.33	5.00	6.33	7.33	0.00	1.33	1.00	0.33	0.00
8	2.00	1.67	1.00	1.33	1.33	0.00	3.33	2.00	0.00	1.67
9	2.67	6.00	5.67	2.33	3.33	3.67	5.00	0.00	9.33	2.67
10	1.67	3.00	4.33	1.33	2.67	0.00	8.33	6.33	5.67	7.33
11	5.00	3.00	4.33	2.33	5.33	2.33	2.67	7.67	2.67	2.67
12	2.00	4.33	6.00	3.33	5.67	0.00	4.00	7.67	0.00	4.33
13	2.67	5.00	2.00	5.67	3.33	0.00	0.00	3.00	0.00	0.00
14	6.67	5.33	7.00	9.33	7.00	0.00	0.33	0.00	0.00	0.00
15	3.00	3.33	2.00	6.00	1.67	0.00	2.00	0.00	0.00	1.33

Taula 5: Mitjana de puntuacions per humans

A l'hora de comparar amb la resposta donada pels models, s'ha de tenir en compte que aquestes mitjanes s'han obtingut a partir de dades amb bastant variància, i que, per tant, poden no ser completament fiables. Tot i això, són suficients per als nostres objectius, ja que busquem obtenir una primera aproximació de la resposta a la pregunta de si la sortida dels models és similar a la donada per humans.

També ens interessa determinar si la llengua utilitzada per fer la sol·licitud afecta la sortida dels models. Analitzem doncs les puntuacions obtingudes al demanar l'anàlisi de la mateixa lletra d'una cançó ("La pasión de Beefie") però canviant l'idioma de la sol·licitud. En el cas de ChatGPT, les dades recollides es resumeixen en la següent taula:

Idioma	**Empatia**	**Creativitat**	**P.d.D.**	**R. I. S.**	**Resiliència**
Cast	8,6 ± 0,8	6,4 ± 0,5	7 ± 0,6	5 ± 0,6	8,8 ± 0,4

Cat	8,6 ± 0,8	5,8 ± 0,4	7,4 ± 0,5	5,6 ± 1,2	9,2 ± 0,4
Ang	9 ± 0,6	5 ± 1,4	6,8 ± 0,7	5,4 ± 0,5	8 ± 0,6

Taula 6: Mitjanes i errors estàndard de "La pasión de Beefie" per ChatGPT en diferents idiomes (primera part), arrodonides al primer decimal

Idioma	**Racisme**	**Sexisme**	**A.d.S.**	**Violència**	**C. S.**
Cast	1,6 ± 0,5	4,2 ± 0,4	7,8 ± 0,4	6,2 ± 0,4	3,4 ± 0,5
Cat	0,4 ± 0,5	3,4 ± 0,8	7,2 ± 0,7	5,88 ± 0,7	3,4 ± 1,0
Ang	0,8 ± 1,0	4 ± 0,9	7,6 ± 0,8	5,8 ± 0,4	3,8 ± 0,4

Taula 7: Mitjanes i errors estàndard de "La pasión de Beefie" per ChatGPT en diferents idiomes (segona part), arrodonides al primer decimal

Per a Gemini, tenim les següents dades:

Idioma	**Empatia**	**Creativitat**	**P.d.D.**	**R. I. S.**	**Resiliència**
Cast	7,7 ± 0,5	6,3 ± 0,5	7 ± 1,6	4,3 ± 0,5	9 ± 0,8
Cat	-	-	-	-	-
Ang	8,8 ± 0,8	5,8 ± 1	7 ± 0,4	2,8 ± 1,2	8,6 ± 0,5

Taula 8: Mitjanes i errors estàndard de "La pasión de Beefie" per Gemini en diferents idiomes (primera part), arrodonides al primer decimal

Idioma	**Racisme**	**Sexisme**	**A.d.S.**	**Violència**	**C. S.**
Cast	1 ± 0,8	3 ± 0,8	8 ± 0,8	7,3 ± 0,9	4,7 ± 0,5
Cat	-	-	-	-	-
Ang	0,6 ± 1,2	1,4 ± 0,8	8,2 ± 0,4	7,4 ± 1,2	2,6 ± 0,5

Taula 9: Mitjanes i errors estàndard de "La pasión de Beefie" per Gemini en diferents idiomes (segona part), arrodonides al primer decimal

Donant una ullada ràpida a aquestes taules, es veu com ChatGPT és bastant consistent entre idiomes, i manté una variància relativament baixa entre tots ells. A Gemini les puntuacions donades són similars, però són menys consistents i tenen una variància més alta.

Un altre fet molt interessant és que, en fer la sol·licitud en català, Gemini es negava a donar una resposta (d'aquí surten els guions a la taula). Curiosament, en fer la sol·licitud en castellà donava sortida la meitat de les vegades, mentre que en anglès donava sortida sempre, tot això amb la mateixa lletra d'entrada. Per tant, tot i que a priori sembla que la sortida és independent de l'idioma amb què es fa la sol·licitud, és en moments com aquest on sorgeixen les diferències.

Pel que fa a les justificacions, els dos models donen justificacions ajustades al que es podria esperar per les puntuacions donades. Tot i això, ChatGPT acostumar a ser molt més específic a l'hora de dir quina és la línia exacta que està utilitzant per justificar la puntuació, mentre que Gemini dona justificacions menys precises.

3.2.3 Comparació

Per tractar estadísticament aquests models, és útil la següent definició:

Definició 3.2.3.1. *Sigui $g : \Theta \to \mathbb{R}^k$ una funció i $(\Omega, \mathcal{A}, \{P_\theta, \theta \in \Theta\}), \Theta \subseteq \mathbb{R}^d$ un model estadístic paramètric. Un estadístic és una funció $T : \Omega \to g(\Theta)$. Qualsevol estadístic que donades n observacions d'una variable aleatòria amb llei P_θ intenta estimar els valors de $g(\theta)$ és conegut com un estimador.*

Un model de llenguatge es pot veure com un estimador que intenta estimar els paràmetres entrenables de la xarxa. En el nostre cas, si volem veure quant de bé aquests models s'apropen als criteris humans a l'hora de puntuar lletres, es pot veure que el model estima els paràmetres de manera que el resultat de la xarxa donarà uns resultats el més propers a les puntuacions donats per humans.

Definició 3.2.3.2. *Una funció de pèrdua és una funció mesurable $W : \mathbb{R}^k \times \Theta \to \mathbb{R}^+$ que representa l'error que cometem en proposar t com a valor estimat de $g(\theta)$, i que compleix $W(t, \theta) = 0 \leftrightarrow t = g(\theta)$*

Per comparar les dades recollides utilitzant els models de llenguatge amb les dades recollides amb humans, estadísticament podem utilitzar una funció de pèrdua. Una de les més senzilles és calcular l'arrel de l'error quadràtic mitjà (RMSE per les seves inicials en anglès) entre la sortida del model de llenguatge i els valors humans. En aquest cas, que es vol comparar dues matrius 15×10, utilitzem la fórmula:

$$\text{RMSE} = \sqrt{\frac{1}{150} \sum_{i=1}^{15} \sum_{j=1}^{10} (a_{i,j} - b_{i,j})^2} \tag{3.1}$$

Fent aquest càlcul entre les taules del ChatGPT i els humans, obtenim un valor de **2,2922**. En canvi, fent el càlcul amb les puntuacions donades per Gemini i humans s'obté un **2,37** (tenint en compte que com Gemini no va puntuar la cançó Q U E V A S H A H A C E R H O Y, s'ha eliminat la fila corresponent). Un altre estimador possible és l'error absolut mitjà (MAE), que es calcula amb la següent fórmula:

$$\text{MAE} = \frac{1}{150} \sum_{i=1}^{15} \sum_{j=1}^{10} \mid a_{i,j} - b_{i,j} \mid \tag{3.2}$$

Aquest estadístic dona una menor importància a diferències molt grans, que amb el RMSE es veuen molt més penalitzades. Ara s'obtenen els valors **1,29** pel ChatGPT, i **1,31** per Gemini. Tot i que és consistent amb els resultats obtinguts amb RMSE, aquí la diferència és més petita.

3.2.4 Conclusió

Aquests models donen unes puntuacions i justificacions raonades i que mirant les dades i les lletres de les cançons, s'ajusten a l'esperat. Tot i que sembla que amb les dades recollides ChatGPT és més precís que Gemini, s'ha de tenir en compte que les dades recollides tenen molta variància, i que, per tant, s'hauria de recollir més per fer una anàlisi amb més profunditat. Tot i això, es pot dir amb relativa certesa que aquestes eines resulten útils per qualificar quant de presents es troben els temes demanats a diferents lletres de cançons, en diferents idiomes i de diferents gèneres i popularitat.

Un resultat més sorprenent és causat per la comprovació de biaixos de llengua. S'ha demostrat empíricament que depenent de l'idioma d'entrada, el xatbot pot decidir no respondre, citant preocupacions pels temes que es demana comentar, però en altres idiomes respon sense cap problema, demostrant que en alguns casos, l'idioma en què es fa la sol·licitud és rellevant.

Tot i això, aquest experiment ha servit sobretot per proposar i provar una possible metodologia a seguir en futurs experiments. Aquests experiments s'han fet amb poques dades, i per tant les conclusions que se'n poden extreure són poc significants, però estableix una possible metodologia efectiva a seguir posteriorment.

3.3 Aplicacions

3.3.1 Comparació artistes

Un cop analitzat la utilitat dels models de llenguatge com a eines per determinar diferents tipus de continguts en les lletres de cançons, presentem dues possibles aplicacions bàsiques d'aquests models que podrien resultar útils.

Una primera aplicació podria ser la comparació de dues artistes. Aquesta comparació podria mostrar ràpidament com són les cançons d'un artista en concret, i així poder tenir una primera idea del contingut d'aquestes. Com a exemple, comparem els continguts de les lletres del top 20 de cançons més populars de la Shakira i la Rosalia. Amb l'ajuda d'un mapa de calor és fàcil comparar d'una ullada quins temes són més prominents depenent de l'artista que es vol estudiar. També és possible generar gràfics de radar amb temes que, per exemple, podrien permetre a figures parentals determinar ràpidament si un artista és adequat pels seus fills. A continuació trobem els mapes de calor de les dues artistes:

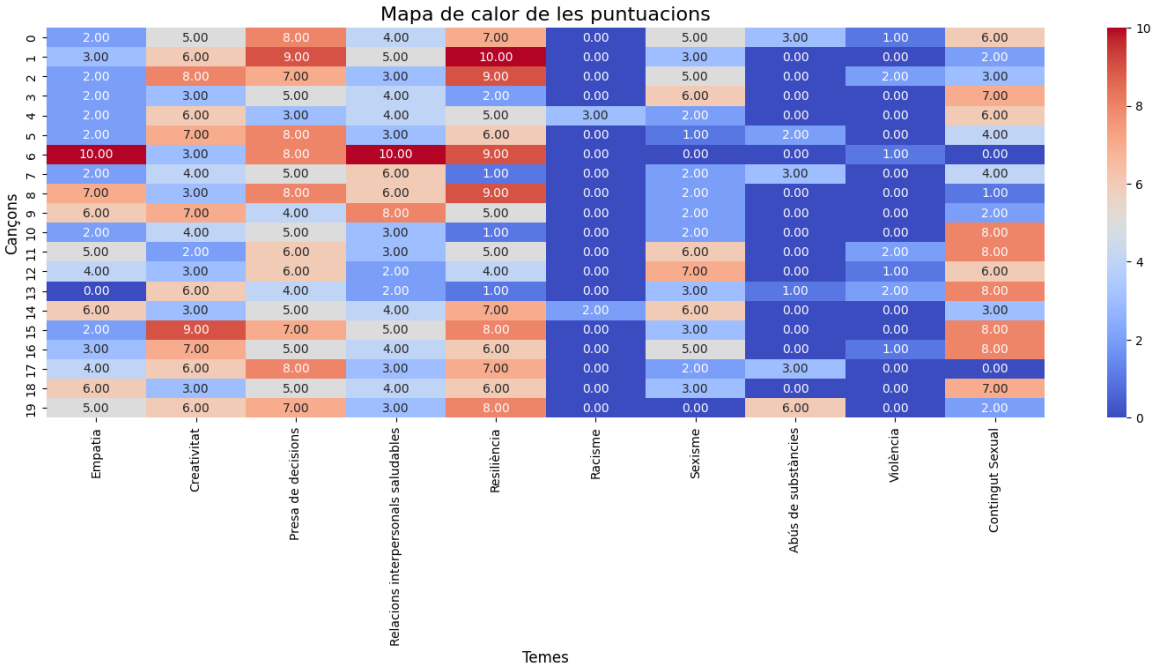


Figura 9: Mapa de calor de les puntuacions obtingudes pel top 20 cançons de Shakira

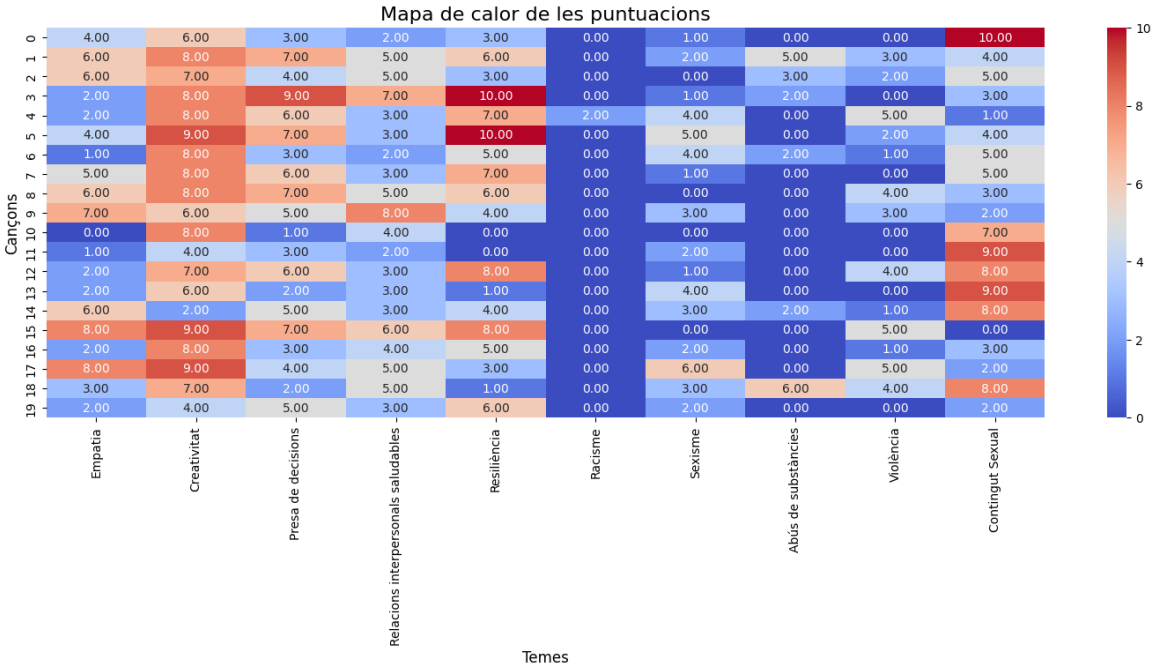


Figura 10: Mapa de calor de les puntuacions obtingudes pel top 20 cançons de Rosalia

També és fàcil calcular i comparar les mitjanes aritmètiques d'aquestes puntua-

cions, i la seva variància:

Artist	Emp	Crea	P.D.	R. S.	Res	Rac	Sexis	A.d.S	Vio	C. S.
Shakira	3,75	5,05	6,15	4,3	5,8	0,25	3,25	0,9	0,5	4,65
Rosalia	3,85	7	4,75	4,05	4,85	0,1	2,2	1	2	4,9

Taula 10: Mitjanes

Artist	Emp	Crea	P.d.D.	R. S.	Res	Rac	Sexis	A.d.S	Vio	C. S.
Shakira	5,39	3,85	2,72	3,71	7,56	0,59	4,09	2,59	0,55	7,83
Rosalia	5,87	3,3	4,29	2,65	8,73	0,19	2,96	3,1	3,6	8,49

Taula 11: Variàncies

Amb aquestes dades es pot ràpidament arribar a conclusions com que les dues artistes, tot i fer música diferent, acostumen a tractar temes similars (sent les lletres de la Shakira una mica més sexistes i les de la Rosalia més violentes), i que totes dues tenen una variabilitat en els temes de les lletres de les seves cançons relativament alta, especialment en el referent al contingut sexual. Visualitzem els seus gràfics de radar:

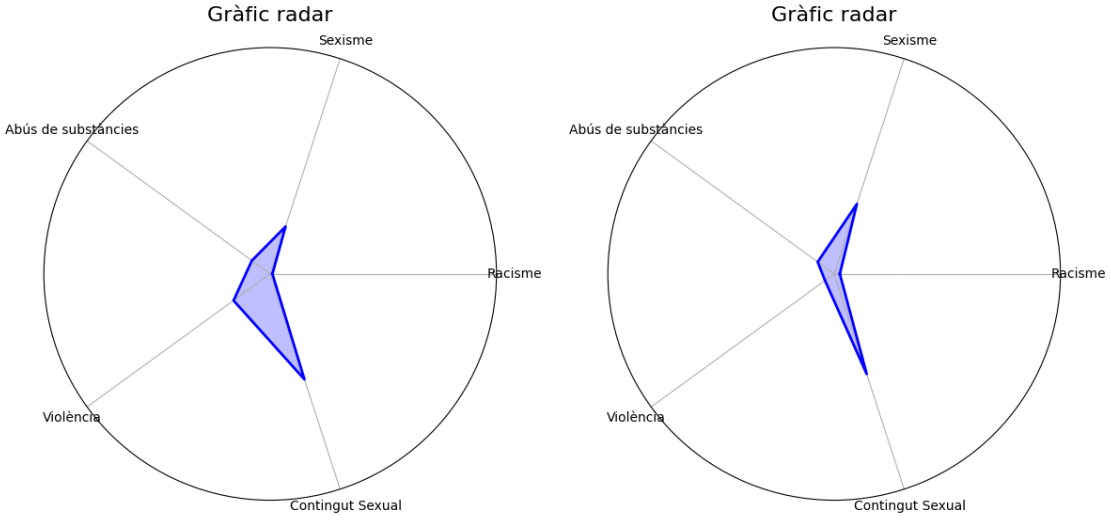


Figura 11: Gràfic radar de les mitjanes del top 20 cançons més populars de la de la Rosalía (esquerra) i la Shakira (dreta)

3.3.2 Qualificació automàtica

Una altra possible aplicació és fer una aplicació que, donada una cançó d'entrada, directament faci la sol·licitud amb la lletra apropiada i retorni una llista amb les puntuacions buscades. Així, es pot automatitzar ràpidament la recollida d'aquest

tipus de dades. Un codi amb Python que exemplifica com es pot fer es troba al repositori de GitHub https://github.com/hecmar007/SongRatings.

Aquest codi fa servir les APIs públiques que ofereixen ChatGPT i Gemini (i també l'API de Genius per aconseguir les lletres de les cançons). Aquestes APIs permeten fer crides als models preentrenats d'OpenAI i Gemini des de codi, i permet fàcilment desenvolupar aplicacions utilitzant tot el poder d'aquests models. Donat el nom de la cançó desitjada, el codi aconsegueix la lletra de la cançó de la base de dades de Genius (una plataforma de lletres de cançons en línia). Després, una sol·licitud molt similar a la usada anteriorment (però lleugerament modificada per fer que la sortida sigui més fàcil d'interpretar) s'envia a Gemini o ChatGPT utilitzant la seva API, que retorna la resposta del model. Finalment, s'utilitza la llibreria `re` per aconseguir la llista de Python amb totes les qualificacions.

4 Conclusions

Aquesta memòria explora l'arquitectura de xarxa neuronal coneguda com a Transformer des del seu funcionament fins al seu entrenament, tenint en compte tota l'àlgebra lineal que forma part de tot aquest procés, específicament en els casos dels xatbots. A més a més, s'explora quan d'adequades són aquests xatbots per analitzar i puntuar lletres de cançons en diferents temes rellevants.

L'anàlisi del funcionament dels Transformers ens ha portat pel camí de veure com l'àlgebra lineal és capaç de crear un model molt complet, i que fent ús del mecanisme d'atenció és capaç d'incorporar informació del context a cada token per així aconseguir una generació de text molt robusta. També hem vist com es poden entrenar aquests models, i com es poden utilitzar tècniques més enllà del descens del gradient per regularitzar-los i fer-los més segurs.

En el cas d'ús hem vist com aquestes eines es poden utilitzar per a l'anàlisi de cançons, i s'han demostrat dues possibles aplicacions. S'ha proposat una possible metodologia a seguir per a futur experiments, i s'ha demostrat la seva utilitat. Amb les dades recollides, em demostrat que les sortides donades pel ChatGPT s'apropen més a els criteris humans, i en principi està lliure de biaixos per llengua. També s'ha vist que Gemini sí que té un biaix a l'hora de decidir quan respondre a una sol·licitud. Tot i això, les dades recollides són poques i la diferència en el RMSE dels dos models és molt petit, i no permet donar una conclusió amb seguretat sobre quin dels dos models seria millor per les tasques plantejades. En una feina futura s'haurien de recollir més dades per poder extreure'n unes conclusions més sòlides. Una altra via per explorar és com es poden aplicar els Transformers a àudio o vídeo en comptes de text, i si es podrien utilitzar per analitzar directament la música o els videoclips d'algunes cançons.

A Cançons i sol·licituds utilitzades

A.1 Cançons

En aquest apèndix es presenten totes es cançons utilitzades i la seva lletra:

- **DEGENERE** per *Myke Towers*
- **Si Antes Te Hubiera Conocido** per *KAROL G*
- **Q U E V A S H A C E R H O Y** per *Omar Courtz i De La Rose*
- **IA** per *Clarent*
- **GRAN VÍA** per *Quevedo i Aitana*
- **Sticky** per *Tyler, the creator*
- **25** per *Rod Wave*
- **Love Sombody** per *Morgan Wallen*
- **tv off** per *Kendrick Lamar*
- **Timeless** per *The Weeknd i Playboy Carti*
- **La Pasión de Beefie** per *Yung Beef*
- **ianomvui druga** per *Rojuu*
- **Cyberpunk** per *Faxu*
- **Calgary 88** per *Antonia Font*
- **Heavenly** per *Judeline*

A.2 Sol·licituds en anglès:

1. I am exploring the lyrics of different songs, and I would like you to assess whether the following song contains these themes: empathy, creativity, decision-making, healthy interpersonal relationships, resilience, racism, sexism, substance abuse, violence, sexual content. Explore these themes as if you were a listener hearing the song for the first time.

   Song lyrics.

   For each theme, provide a score from 0 to 10 based on how present and relevant the theme is in the lyrics. For each score, specify exactly which line or lines from the lyrics you used to justify it.

2. I want you to analyze the following song lyrics. Your task is to determine to what extent these lyrics address the following topics: empathy, creativity, decision-making, healthy interpersonal relationships, resilience, racism, sexism, substance abuse, violence, sexual content.

   For each topic, assign a score from 0 to 10, where 0 means there is no mention or allusion to the topic, and 10 means the topic is central and is addressed in great detail or depth. Then, briefly explain why you assigned this score, quoting the exact lines from the song that led you to this conclusion.

   Example of the expected response for a topic:

   Resilience (6/10):

   "I've fallen a thousand times, but I always get back up." This line conveys a message of personal perseverance, but the song does not further develop this aspect elsewhere in the lyrics, which is why I did not assign a higher score.

   Song lyrics

A.3 Sol·licituds en castellà:

1. Estoy explorando las letras de diferentes canciones, y me gustaría que evaluaras si la siguiente canción contiene estos temas: empatía, creatividad, toma de decisiones, relaciones interpersonales saludables, resiliencia, racismo, sexismo, abuso de sustancias, violencia, contenido sexual. Explora estos temas como si fueras un oyente escuchando la canción por primera vez.

   Letra de la canción

   Para cada tema, da una puntuación del 0 al 10 según cuán presente y relevante sea el tema en la letra. Para cada puntuación, especifica exactamente qué línea o líneas de la letra has utilizado para justificarla.

2. Quiero que analices la siguiente letra de una canción. Tu tarea es determinar en qué medida esta letra aborda los siguientes temas: empatía, creatividad, toma de decisiones, relaciones interpersonales saludables, resiliencia, racismo, sexismo, abuso de sustancias, violencia, contenido sexual.

   Para cada tema, asigna una puntuación del 0 al 10, donde 0 significa que no hay ninguna mención o alusión al tema, y 10 significa que el tema es central y se trata con gran detalle o profundidad. Después, explica brevemente por qué has asignado esta puntuación, citando las líneas exactas de la canción que te llevaron a esta conclusión.

   Ejemplo de la respuesta esperada para un tema:

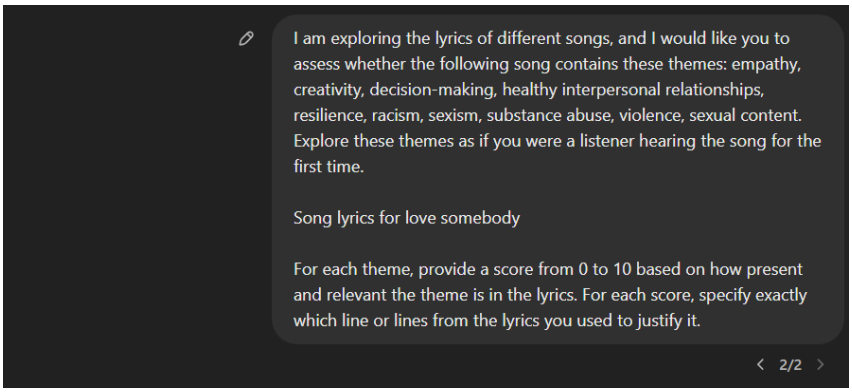
   Resiliencia (6/10):

   "He caído mil veces, pero siempre me levanto" Esta línea transmite un mensaje de superación personal, pero no desarrolla más este aspecto en otras partes de la letra, por lo que no asigno una puntuación más alta.

   Letra de la canción.

A.4 Sol·licitud i resposta

Aquí es presenten les captures d'una sol·licitud completa al ChatGPT.

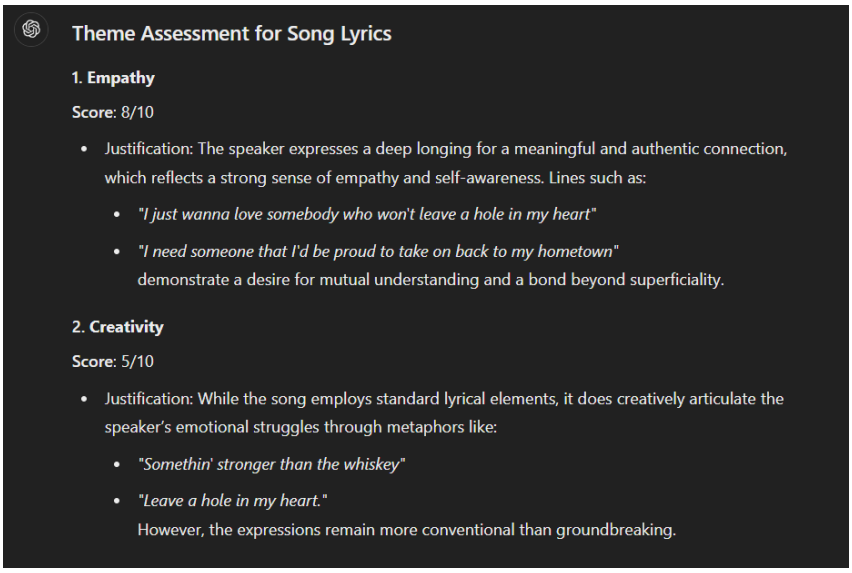


I am exploring the lyrics of different songs, and I would like you to assess whether the following song contains these themes: empathy, creativity, decision-making, healthy interpersonal relationships, resilience, racism, sexism, substance abuse, violence, sexual content. Explore these themes as if you were a listener hearing the song for the first time.

Song lyrics for love somebody

For each theme, provide a score from 0 to 10 based on how present and relevant the theme is in the lyrics. For each score, specify exactly which line or lines from the lyrics you used to justify it.

2/2



Theme Assessment for Song Lyrics

1. Empathy

Score: 8/10

- Justification: The speaker expresses a deep longing for a meaningful and authentic connection, which reflects a strong sense of empathy and self-awareness. Lines such as:
  - *"I just wanna love somebody who won't leave a hole in my heart"*
  - *"I need someone that I'd be proud to take on back to my hometown"*

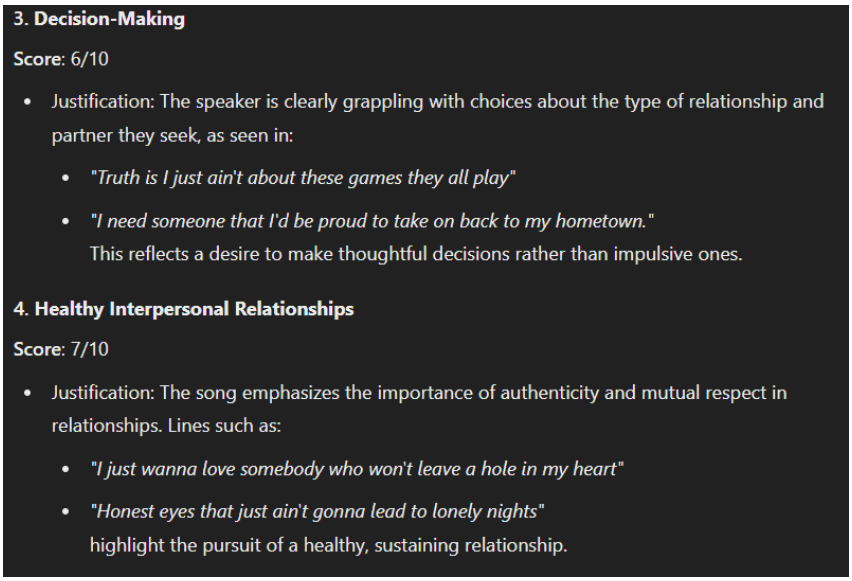
  demonstrate a desire for mutual understanding and a bond beyond superficiality.

2. Creativity

Score: 5/10

- Justification: While the song employs standard lyrical elements, it does creatively articulate the speaker's emotional struggles through metaphors like:
  - *"Somethin' stronger than the whiskey"*
  - *"Leave a hole in my heart."*

  However, the expressions remain more conventional than groundbreaking.



3. Decision-Making

Score: 6/10

- Justification: The speaker is clearly grappling with choices about the type of relationship and partner they seek, as seen in:
  - *"Truth is I just ain't about these games they all play"*
  - *"I need someone that I'd be proud to take on back to my hometown."*

  This reflects a desire to make thoughtful decisions rather than impulsive ones.

4. Healthy Interpersonal Relationships

Score: 7/10

- Justification: The song emphasizes the importance of authenticity and mutual respect in relationships. Lines such as:
  - *"I just wanna love somebody who won't leave a hole in my heart"*
  - *"Honest eyes that just ain't gonna lead to lonely nights"*

  highlight the pursuit of a healthy, sustaining relationship.

5. Resilience

Score: 6/10

- Justification: The speaker's persistence in searching for love despite repeated disappointments shows resilience. This is evident in:
  - *"And oh, I've tried, but every time I feel her kiss me, I keep comin' up empty."*
    However, the tone remains wistful and longing, with less focus on overcoming challenges fully.

6. Racism

Score: 0/10

- Justification: The lyrics do not address or allude to racism.

7. Sexism

Score: 0/10

- Justification: The lyrics do not contain language or themes that critique or perpetuate sexism.

8. Substance Abuse

Score: 3/10

- Justification: The mention of whiskey in:
  - *"Somethin' stronger than the whiskey"*
    alludes to the use of alcohol as a coping mechanism, but the theme is not deeply explored.

9. Violence

Score: 0/10

- Justification: The lyrics do not mention or imply violence.

10. Sexual Content

Score: 4/10

- Justification: There is mild implication of intimacy in lines like:
  - *"Even when she's up against me, I don't know why it feels like there's nobody with me."*
    However, the focus is more on emotional connection than physical.

Summary Table:

Theme	Score	Justification
Empathy	8	Desire for genuine connection and understanding.
Creativity	5	Metaphors present but conventional in style.
Decision-Making	6	Reflects thoughtful choices about relationships.
Healthy Relationships	7	Emphasis on authenticity and mutual respect.
Resilience	6	Persistence despite repeated disappointments.
Racism	0	No mention of racism.
Sexism	0	No mention or perpetuation of sexism.
Substance Abuse	3	Brief mention of whiskey as a coping mechanism.
Violence	0	No mention or implication of violence.
Sexual Content	4	Mild implication of intimacy but focus is emotional.

B Puntuacions

B.1 Puntuacions per humans

Emp	Crea	P.d.D.	R. I. S.	Res	Rac	Sexisme	A.d.S	Vio	C. S.
0	1	0	0	0	0	9	7	5	7
5	1	4	4	0	0	7	0	2	8
0	1	0	0	0	0	10	0	2	10
0	1	0	0	0	0	10	0	5	10
5	1	5	4	0	0	5	1	0	10
0	7	0	0	5	9	10	10	10	10
7	7	8	7	8	0	0	0	0	0
0	2	0	0	0	0	6	0	0	0
8	9	9	7	9	0	0	0	9	0
4	6	7	2	5	0	9	2	0	7
6	5	5	5	8	3	0	5	0	2
3	8	8	8	8	0	3	7	0	2
5	7	3	7	6	0	0	5	0	0
10	9	8	9	9	0	0	0	0	0
2	3	4	7	0	0	4	0	0	0

Taula 12: Puntuacions Home 52 anys

Emp	Crea	P.d.D.	R. I. S.	Res	Rac	Sexisme	A.d.S	Vio	C. S.
5	3	6	1	4	0	7	7	2	6
3	5	4	6	6	0	1	0	0	2
4	4	7	8	5	0	4	3	0	10
0	6	6	0	1	0	9	0	6	4
4	6	7	5	7	0	5	2	0	6
0	2	5	0	0	8	10	10	10	10
3	3	5	6	8	0	0	0	1	0
2	3	0	2	4	0	0	1	0	1
0	4	8	0	1	7	9	0	10	8
1	3	6	2	3	0	8	9	8	7
7	4	5	2	2	0	6	9	2	4
3	3	2	0	2	0	8	9	0	9
3	5	3	3	4	0	0	3	0	0
10	7	9	10	8	0	1	0	0	0
7	4	2	8	5	0	0	0	0	4

Taula 13: Puntuacions Home 23 anys

Emp	Crea	P.d.D.	R. I. S.	Res	Rac	Sexisme	A.d.S	Vio	C. S.

0	2	2	6	0	0	2	5	0	6
0	1	1	1	0	0	3	0	0	4
0	3	0	1	0	0	0	0	0	10
0	0	0	0	0	0	7	0	5	6
2	1	0	0	0	0	0	5	0	7
0	0	0	0	1	3	8	8	8	9
7	3	2	6	6	0	4	3	0	0
4	0	3	2	0	0	4	5	0	4
0	5	0	0	0	4	6	0	9	0
0	0	0	0	0	0	8	8	9	8
6	3	3	0	6	0	2	9	6	4
0	2	8	2	7	0	4	7	0	4
0	3	0	7	0	0	0	6	0	0
0	0	4	9	4	0	0	0	0	0
0	3	0	3	0	0	0	0	0	0

Taula 14: Puntuacions Dona 22 anys

B.2 Puntuacions per idiomes

Idioma	Empatia	Crea	P.d.D.	R. I. S.	Res	Racisme	Sexisme	A.d.S	Violència	C. S.
Castellà	9	6	7	6	8	1	4	7	6	3
Castellà	10	7	8	5	9	2	4	8	6	4
Castellà	8	6	7	5	9	2	4	8	6	3
Castellà	8	7	6	5	9	1	4	8	7	3
Castellà	8	6	7	4	9	2	5	8	6	4
Català	10	5	8	7	9	1	4	6	5	3
Català	8	6	7	4	9	1	3	7	6	4
Català	8	6	7	5	9	0	2	7	5	3
Català	9	6	8	7	10	0	4	8	7	5
Català	8	6	7	5	9	0	4	8	6	2
Anglès	9	3	6	5	8	0	4	7	6	4
Anglès	8	4	6	5	7	2	3	7	6	3
Anglès	10	7	8	6	9	2	5	8	6	4
Anglès	9	5	7	6	8	0	3	7	5	4
Anglès	9	6	7	5	8	0	5	9	6	4

Taula 15: Puntuacions de “La pasión de Beefie per ChatGPT en diferents idiomes

Idioma	Empatia	Crea	P.d.D.	R. I. S.	Res	Racisme	Sexisme	A.d.S	Violència	C. S.
Castellà	8	6	7	5	9	2	4	8	8	5
Castellà	8	7	9	4	10	1	2	9	8	5
Castellà	-	-	-	-	-	-	-	-	-	-
Castellà	-	-	-	-	-	-	-	-	-	-

Castellà	7	6	5	4	8	0	3	7	6	4
Català	-	-	-	-	-	-	-	-	-	-
Català	-	-	-	-	-	-	-	-	-	-
Català	-	-	-	-	-	-	-	-	-	-
Català	-	-	-	-	-	-	-	-	-	-
Català	-	-	-	-	-	-	-	-	-	-
Anglès	9	7	8	5	9	3	2	9	8	3
Anglès	8	6	7	2	9	0	2	8	7	3
Anglès	8	6	7	2	9	0	1	8	9	3
Anglès	9	6	7	3	8	0	2	8	7	2
Anglès	10	4	7	2	8	0	0	8	6	2

Taula 16: Puntuacions de "La pasión de Beefie per Gemini en diferents idiomes

B.3 Puntuacions d'artistes

Títol	**Empatia**	**Crea**	**P.d.D.**	**R. I. S.**	**Res**	**Racisme**	**Sexisme**	**A.d.S**	**Violència**	**C. S.**
LA NOCHE DE ANOCHE	4	6	3	2	3	0	1	0	0	10
Omega	6	8	7	5	6	0	2	5	3	4
BESO	6	7	4	5	3	0	0	3	2	5
Despechá	2	8	9	7	10	0	1	2	0	3
MALAMENTE	2	8	6	3	7	2	4	0	5	1
New Woman	4	9	7	3	10	0	5	0	2	4
COON ALTURA	1	8	3	2	5	0	4	2	1	5
LLYLM	5	8	6	3	7	0	1	0	0	5
LA FAMA	6	8	7	5	6	0	0	0	4	3

Yo x Ti, Tu x Mi	7	6	5	8	4	0	3	0	3	2
DI MI NOMBRE	0	8	1	4	0	0	0	0	0	7
TUYA	1	4	3	2	0	0	2	0	0	9
Antes de morirme	2	7	6	3	8	0	1	0	4	8
Besos moja2	2	6	2	3	1	0	4	0	0	9
EL PAÑUELO	6	2	5	3	4	0	3	2	1	8
COMO UN G	8	9	7	6	8	0	0	0	5	0
LA COMBI VERSACHE	2	8	3	4	5	0	2	0	1	3
PIENSO EN TU MIRÁ	8	9	4	5	3	0	6	0	5	2
Linda	3	7	2	5	1	0	3	6	4	8
CANDY	2	4	5	3	6	0	2	0	0	2

Taula 17: Puntuacions ChatGPT del top 20 cançons de Rosalia

Títol	**Empatia**	**Crea**	**P.d.D.**	**R. I. S.**	**Res**	**Racisme**	**Sexisme**	**A.d.S**	**Violència**	**C. S.**
Soltera	2	5	8	4	7	0	5	3	1	6
TQG	3	6	9	5	10	0	3	0	0	2
Shakira: Bzrp Musci Sessions	2	8	7	3	9	0	5	0	2	3
Perro Fiel	2	3	5	4	2	0	6	0	0	7
Hips don't lie	2	6	3	4	5	3	2	0	0	6
Te felicito	2	7	8	3	6	0	1	2	0	4
Acróstico	10	3	8	10	9	0	0	0	1	0
Me enamoré	2	4	5	6	1	0	2	3	0	4
Monotonía	7	3	8	6	9	0	2	0	0	1
La bicicleta	6	7	4	8	5	0	2	0	0	2
Clandestino	2	4	5	3	1	0	2	0	0	8

Me gusta	5	2	6	3	5	0	6	0	2	8
Chantaje	4	3	6	2	4	0	7	0	1	6
Puntería	0	6	4	2	1	0	3	1	2	8
La tortura	6	3	5	4	7	2	6	0	0	3
Loba	2	9	7	5	8	0	3	0	0	8
Las de la intuición	3	7	5	4	6	0	5	0	1	8
Deja vu	4	6	8	3	7	0	2	3	0	0
Copa vacía	6	3	5	4	6	0	3	0	0	7
DON'T YOU WORRY	5	6	7	3	8	0	0	6	0	2

Taula 18: Puntuacions ChatGPT del top 20 cançons de Shakira

Referències

[Ber23] Dave Bergmann. *What is reinforcement learning from human feedback (RLHF)?* Accessed: 2024-12-30. Nov. de 2023. URL: https://www.ibm.com/think/topics/rlhf.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros i Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML]. URL: https://arxiv.org/abs/1607.06450.

[Bro+20] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[BV04] Stephen Boyd i Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: 9780521833783. URL: http://web.stanford.edu/~boyd/cvxbook/.

[Chr+23] Paul Christiano et al. *Deep reinforcement learning from human preferences*. 2023. arXiv: 1706.03741 [stat.ML]. URL: https://arxiv.org/abs/1706.03741.

[Dev+19] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[DG03] Sanjoy Dasgupta i Anupam Gupta. "An Elementary Proof of a Theorem of Johnson and Lindenstrauss". A: *Random Structures & Algorithms* 22.1 (2003), pàg. 60 - 65. DOI: 10.1002/rsa.10073.

[Elh+22] Nelson Elhage et al. "Toy Models of Superposition". A: *Transformer Circuits Thread* (2022). URL: https://transformer-circuits.pub/2022/toy_model/index.html.

[GBC16] Ian Goodfellow, Yoshua Bengio i Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Gir23] L. Giray. "Prompt Engineering with ChatGPT: A Guide for Academic Writers". A: *Annals of Biomedical Engineering* 51 (2023), pàg. 2629 - 2633. DOI: 10.1007/s10439-023-03272-4.

[IS15] Sergey Ioffe i Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[JMR21] Olga Julià, David Márquez-Carrera i Carles Rovira. *Estadística: problemes i més problemes*. Vol. 431. Textos docents. Barcelona: Edicions Universitat de Barcelona, 2021, pàg. 269. ISBN: 9788491686453.

[Kap+20] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

[Kar24] Andrej Karpathy. *Let's build GPT: from scratch, in code, spelled out.* https://www.youtube.com/watch?v=kCc8FmEb1nY&t=4s. Accessed: 2025-01-05. 2024.

[KB17] Diederik P. Kingma i Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[KR18] Taku Kudo i John Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Network-based text generation*. Accessed: 2024-11-03. 2018. URL: https://github.com/google/sentencepiece.

[Kum24] Praveen Kumar. *Decoding the Transformer Model Architecture, Loss Function, and Inference from the Attention Is All You Need Paper*. Accessed: 2024-12-17. 2024. URL: https://praveenkumar2909.medium.com/decoding-the-transformer-model-architecture-loss-function-and-inference-from-the-attention-is-717b98d183b3.

[LH17] Ilya Loshchilov i Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. arXiv: 1608.03983 [cs.LG]. URL: https://arxiv.org/abs/1608.03983.

[Mik+13] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[Ope23] OpenAI. *Tiktoken: A fast BPE tokeniser for use with OpenAI's models*. Accessed: 2024-11-03. 2023. URL: https://github.com/openai/tiktoken.

[PH22] Mary Phuong i Marcus Hutter. *Formal Algorithms for Transformers*. 2022. arXiv: 2207.09238 [cs.LG]. URL: https://arxiv.org/abs/2207.09238.

[Rad+18] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". A: (2018). OpenAI preprint. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[Rad+19] Alec Radford et al. *Language Models are Unsupervised Multitask Learners*. OpenAI preprint. 2019. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[Raf+23] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG]. URL: https://arxiv.org/abs/1910.10683.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton i Ronald J. Williams. "Learning representations by back-propagating errors". A: *Nature* 323.6088 (1986), pàg. 533 - 536.

[Sch+17] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[Su+23] Jianlin Su et al. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. arXiv: 2104.09864 [cs.CL]. URL: https://arxiv.org/abs/2104.09864.

[Tib13] Ryan Tibshirani. *Lecture 6: September 12*. Lecture notes for Convex Optimization (10-725), Carnegie Mellon University. 2013. URL: https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec6.pdf.

[Vas+23] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.