

Treball de Recerca: Creació d'un xatbot aplicat en l'institut

Em pots explicar de què tracta aquest treball de recerca?

Aquest treball de recerca té com a objectiu crear un Xat Bot per a l'Institut Doctor Puigvert.

Quin és l'objectiu d'aquest treball de recerca?

L'objectiu és crear un Xat Bot que modernitzi la pàgina web de l'institut i faci més fàcil l'obtenció d'informació per als visitants.

Per què va sorgir la idea de crear un Xat Bot per a l'institut?

La idea va sorgir per la necessitat de modernitzar la pàgina web i oferir una manera més fàcil d'accedir a la informació del centre.

En què se centra aquest treball?

El treball se centra en combinar les tecnologies més recents per desenvolupar un Xat Bot que pugui respondre preguntes de manera eficient.

Com funciona el Xat Bot desenvolupat en aquest projecte?

Aquest Xat Bot utilitza un sistema que recull informació específica de l'institut per oferir les millors respostes possibles als que el facin servir.

Escriu aquí el teu missatge

NAME

2023 - 2024

TUTOR NAME

1r - 2n Batxillerat

HIDDEN

Índex

0.	Resum / Abstract	4
1.	Introducció	5
2.	Objectius	6
3.	Acrònims	7
4.	Marc teòric	8
4.1.	Què és la intel·ligència artificial?.....	8
4.2.	Que és un LLM?	12
4.3.	<i>Prompt engineering?</i>	15
4.4.	Explicació del funcionament dels <i>embeddings</i>	17
4.5.	Models de LLM: GPT, Gemini i Llama	20
4.6.	MoE, barreja d'experts?	21
4.7.	Què és una API?	22
4.8.	Virtualització i contenidors.	22
5.	Part pràctica	24
5.1.	Entorn de treball i estructura de tot el sistema	24
5.2.	Executar proveïdor de LLM	25
5.3.	Extreure les dades de la pàgina web	25
5.4.	Creació del servidor RAG	26
5.5.	Creació del servidor API.....	27
5.6.	Creació de la pàgina web / interfície d'usuari	28
5.7.	Desplegament	29
5.8.	Prova del xatbot	31
5.8.1.	Exemples de xats dels usuaris	34
6.	Conclusions	37
7.	Continuació del TdR.....	38
8.	Bibliografia web.....	39

Agraïments

Vull donar les gràcies als meus pares per ajudar-me a provar el xatbot i a donar-me idees per millorar el document. També vull donar les gràcies al meu tutor del TdR, TUTOR NAME, per donar-me tota la llibertat en la creació del TdR, d'aquesta manera he tingut via lliure per fer el TdR, i he pogut aprofitar molt més el temps per fer el TdR. Per últim, vull donar les gràcies a HELPER per ajudar-me a explicar millor els temes complexos que tracto en el meu document i a totes les persones que d'una manera o altres han col·laborat en el meu treball.

0. Resum / Abstract

Català:

Aquest treball de recerca té com a objectiu la creació d'un xatbot per a l'Institut HIDDEN. Vaig decidir fer aquest treball de recerca per modernitzar la pàgina web de l'institut i facilitar als visitants l'accés a informació sobre l'institut. El treball consisteix en unir les últimes tecnologies per poder crear un xatbot funcional per l'institut, aquest està basat en models LLM que mitjançant la utilització d'un RAG fet específicament per l'institut on obté informació per respondre als usuaris amb la millor resposta possible que permet la tecnologia actual.

El document està estructurat en dues parts: una teòrica i una pràctica. La part teòrica explica que és la intel·ligència artificial, el funcionament dels LLM, i el desenvolupament d'APIs, mentre que la part pràctica descriu el procés de programació i implementació del xatbot, incloent-hi la creació de servidors per a la gestió de dades, la interacció amb el model LLM, i la creació d'una interfície d'usuari.

English:

This research work aims to create a chatbot for the HIDDEN Institute. I decided to do this research work to modernize the website of the institute and make it easier for visitors to access information about the institute. The work consists of combining the latest technologies to be able to create a functional chatbot for the institute, this is based on LLM models that through the use of a RAG made specifically for the institute where it obtains information to respond to users with the best possible answer that current technology allows.

The document is structured in two parts: a theory and a practice. The theoretical part explains that it is artificial intelligence, the operation of LLMs, and the development of APIs, while the practical part describes the process of programming and implementing the chatbot, including the creation of servers for the management of data, the interaction with the LLM model, and the creation of a user interface.

1. Introducció

El treball de recerca o TdR és una activitat de recerca que han de fer tots els estudiants de Batxillerat a Catalunya. Consta d'una part escrita i una presentació oral. Aquest treball representa el 10% de la nota final del batxillerat.

El TdR està tutoritzat per un professor o professora, que guia i assessora l'estudiant a cada fase del procés de recerca a través de reunions periòdiques.

Cada alumne escull lliurement el seu TdR, i en el meu cas la idea és crear un xatbot tipus ChatGPT. He escollit fer aquest TdR perquè podria donar una visió més moderna a la pàgina de l'Institut i causaria un impacte positiu als visitants de la web, a més tindran una ajuda a l'hora de cercar informació sobre l'Institut.

Aquest serà una petita finestra dins la pàgina de l'HIDDEN, en la qual es veurà el xatbot a on es podrà preguntar dubtes relacionats amb l'Institut. A més, aquest bot estarà sincronitzat amb la web de l'Institut, per tant, podrà respondre amb la informació actualitzada.

2. Objectius

El meu objectiu en aquest TdR és millorar l'accessibilitat i l'experiència d'usuari en l'ús de la pàgina web de l'institut. Com que la Intel·ligència Artificial (IA) és notícia actualment, volia saber si era possible crear un xatbot des de casa i a més aplicar-lo a l'institut, per comprovar si d'aquesta manera es poden complir els objectius de millorar l'accessibilitat i l'experiència d'usuari de la pàgina web de l'institut.

Aquest TdR s'alinea amb 3 objectius dels 17 de l'agenda 2030. Aquests objectius són:

- 4.** Educació de qualitat
(<https://www.un.org/sustainabledevelopment/es/education/>):
Facilitar a les persones sense coneixements necessaris a informar-se sobre com accedir a una educació i a entendre com és el procés en aquest institut, per conèixer-lo i obtenir ajuda. També es pot aprofitar els xats per poder conèixer quins dubtes tenen principalment les persones, i d'aquesta manera es podrà millorar la informació i l'educació a futur. Per exemple, si s'observa que en el xat es pregunta molt "que faig si el meu fill es baralla molt", això permetrà als professors millorar l'educació en la relació entre les persones per millorar la convivència.
- 9.** Indústria, innovació i infraestructures
(<https://www.un.org/sustainabledevelopment/es/infrastructure/>):
La mateixa creació del xatbot i el sistema de cerca amb IA, fan servir una tecnologia molt nova la qual promou la innovació en nous models d'intel·ligència artificial i en el progrés tecnològic.
- 10.** Reducció de les desigualtats
(<https://www.un.org/sustainabledevelopment/es/inequality/>):
Persones amb menys recursos, coneixements o dificultats per fer servir equips electrònics, podrien parlar amb el xatbot per obtenir la informació requerida.



(Imatge dels objectius agenda 2030 descrits a sobre. Imatges tretes de <https://www.un.org/sustainabledevelopment/es/2015/09/la-asamblea-general-adopta-la-agenda-2030-para-el-desarrollo-sostenible/>)

3. Acrònims

IA	Intel·ligència Artificial
LLM	Model de llenguatge extens
Token	Identificador d'un conjunt de paraules
Prompt	Text on estan les instruccions, també se li diu al text que dona l'usuari quan parla amb un LLM
Embedding	Representació en forma de vector d'un <i>token</i> dins de les primeres capes de processament d'un LLM.
MoE	<i>Mixture of Experts</i> (Barreja d'experts): Ajuntar/unir diversos LLM per tal que treballin com un de sol.
CPU	<i>Central Processing Unit</i> (Unitat central de processament), és el component principal d'un ordinador que executa instruccions i fa càlculs, controlant i coordinant el funcionament de tot el sistema.
GPU	<i>Graphics processing unit</i> (Unitat de procés gràfic), és un component especialitzat a processar i renderitzar gràfics, accelerant càlculs complexos, especialment en tasques visuals i de computació paral·lela, com ara gràfics 3D i aprenentatge automàtic.
API	<i>Application Programming Interface</i> (Interfície de programació d'aplicacions), conjunt de regles i definicions que permeten que diferents programes o aplicacions interactuïn entre si.
IP	Adreça d'una màquina a internet o en una xarxa.

4. Marc teòric

En tot l'apartat del marc teòric s'explicarà a detall les tecnologies que farà servir en la creació de tot el sistema del xatbot començant per conceptes més teòrics i definicions acabant amb programes i estructures pel sistema.

4.1. Què és la intel·ligència artificial?

Definició

“La intel·ligència artificial (abreujat: IA) és una part de la informàtica, dedicada al desenvolupament d'algorismes que permet a una màquina (habitualment un computador) prendre decisions intel·ligents o, si més no, comportar-se com si tingués una intel·ligència semblant a la humana.”

(Intel·ligència Artificial. Wikipedia, 27/8/2024. Enllaç a la Bibliografia)

És a dir, la IA busca que els ordinadors puguin aprendre, raonar, resoldre problemes i fins i tot ser creatives, igual que nosaltres els humans.

Historia

La idea que les màquines poguessin pensar com els humans es va començar a investigar des de els anys 50, quan Alan Turing va proposar la famosa Prova de Turing per mesurar la capacitat d'una màquina per mostrar un comportament intel·ligent, indistingible del d'un humà. Aquest moment és considerat un dels punts de partida en la història de la intel·ligència artificial.

La IA ha passat per diverses fases, avui dia, gràcies als avenços en el processament de dades i l'aprenentatge automàtic s'ha començat a veure el gran potencial que tenen.

Us de la IA avui dia

Avui dia, la IA fa anys que es fa servir, els casos més coneguts són els següents:

- Medicina: La IA s'utilitza per diagnosticar malalties mitjançant la interpretació d'imatges mèdiques, com ara radiografies o ressonàncies magnètiques.
- Vehicles autònoms: Els automòbils que es condueixen sols, com els desenvolupats per empreses com Tesla o Waymo, depenen de la IA per analitzar l'entorn, prendre decisions de conducció i evitar obstacles en temps real.
- Assistents virtuals: Assistents com Siri, Alexa o Google Assistant utilitzen IA per entendre el llenguatge natural, respondre preguntes i realitzar

tasques, des de posar una alarma fins a controlar dispositius en una llar intel·ligent.

- Entreteniment: Plataformes com YouTube, Netflix, Spotify, entre altres, utilitzen algoritmes d'IA per personalitzar les recomanacions de vídeos, pel·lícules, sèries i música, analitzant el comportament dels usuaris per oferir-los contingut que els pugui interessar.

Un exemple pràctic per entendre el funcionament de l'IA: distingir entre gossos i gats

Imaginem que volem desenvolupar un sistema d'intel·ligència artificial capaç de distingir entre gossos i gats a partir d'imatges. Per entendre com es dissenya un sistema d'aquest tipus, podem començar pensant en com ho faria una persona. Com li explicaríem a una persona quines són les diferències entre un gos i un gat? A simple vista, tots dos animals tenen característiques molt similars:

Gos:

- Té dues orelles.
- Té un nas.
- Camina a quatre potes.
- Són característiques comunes, però no suficients per a diferenciar-lo del gat.

Gat:

- També té dues orelles.
- Té un nas.
- Camina a quatre potes.
- De nou, no són suficients per distingir-lo del gos amb precisió.

Si ens basem només en aquestes característiques generals, ens adonem que resulta complicat descriure les diferències amb exactitud, ja que hi ha molts trets en comú. No obstant això, com a éssers humans, som capaços de diferenciar un gos d'un gat a primera vista, fins i tot si no podem verbalitzar de manera clara totes les diferències. Com ho aconseguim? La resposta és que ho fem a partir de l'experiència: hem vist molts gossos i gats al llarg de la nostra vida, i hem après a reconèixer patrons específics en la seva aparença, comportament i altres detalls subtils.

La intel·ligència artificial ve a solucionar aquest problema amb un enfocament similar. En lloc de proporcionar una llista exhaustiva de regles que defineixen les diferències entre gossos i gats, el que fem és entrenar el sistema amb un

gran volum d'imatges d'ambdós animals. Aquest procés es coneix com a *aprenentatge automàtic* (*machine learning*).

Durant la fase d'entrenament, a la IA se li proporcionen milers (o fins i tot milions) d'imatges, cadascuna etiquetada com a "gos" o "gat". El sistema d'intel·ligència artificial no rep instruccions explícites sobre quines són les diferències; en canvi, l'algoritme comença a identificar patrons en les imatges per si mateix. A mesura que processa més dades, l'algoritme millora en la seva capacitat de detectar trets característics: potser nota que els gossos solen tenir musells més amples o que els gats tenen una postura corporal diferent.

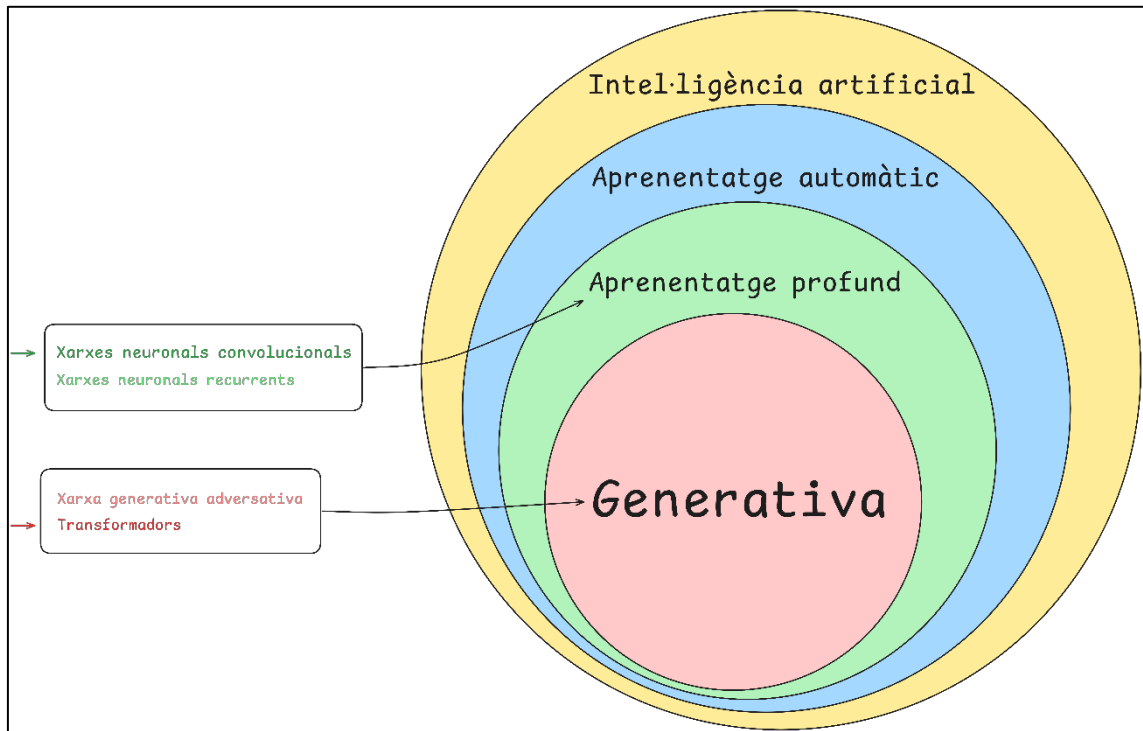
A partir d'aquests patrons, el model "aprèn" a classificar noves imatges amb una gran precisió, sense la necessitat de seguir regles predefinides. Aquest procés d'aprenentatge és conegut com a *aprenentatge supervisat*, ja que les imatges entrenades inclouen etiquetes que indiquen a la IA si l'animal és un gos o un gat.

Branques de la intel·ligència artificial

Aquesta branca de la intel·ligència artificial es coneix com a *aprenentatge automàtic* o *machine learning* en anglès. A la vegada està subdividida en moltes més branques, ja que per cada *aprenentatge automàtic* es fa servir un mètode que estigui optimitzat per aquesta tasca. En el cas d'identificar imatges, se solen fer servir les Xarxes Neuronals Convolucionals (CNNs). En el cas d'aquest TdR s'entra encara més endins, es fan servir els transformadors (*transformers*) que es troben dins de la branca de l'*aprenentatge profund* (*deep learning*).

Gràcies a la creació dels transformadors avui dia existeix ChatGPT entre altres models. El que s'ha trobat en els últims mesos és que els transformadors són tan bons que a l'hora de mesclar-los amb altres entrades i sortides superen la qualitat que tenen models específics en una tasca, vol dir que si fas servir un model fet amb transformadors i fas servir un fet amb CNNs, encara que l'arquitectura dels CNNs estigui optimitzada per reconèixer patrons en imatges, els transformadors seran més precisos i més detallistes en reconèixer el contingut d'una imatge, aquest gran poder ve donat per la propietat que com més gran és un model i més dades se li donen, les seves capacitats augmenten, aquesta propietat només està present en els transformadors, si proves d'ampliar una CNN o li proporciones més dades l'augment en la qualitat i precisió del model no serà tan extens que com el d'un transformador. El gran desavantatge que tenen els transformadors és que la complexitat en computació és tan elevada que cada cop que

s'augmenta la quantitat de dades d'entrada o la mida del model la quantitat de potència que es necessita per executar-lo augmenta de manera exponencial, per sort s'estan investigant alternatives per poder intentar evitar l'augment exponencial quan s'augmenta la mida del model o la quantitat de dades en l'entrada.



(Imatge inspirada en <https://www.devoteam.com/expert-view/unlimited-creativity-how-generative-ai-is-transforming-the-world-of-innovation/>, on es veu en forma de cercles, uns dins d'altres, que cada part de la IA pertany a un grup dins de la mateixa IA. La generativa pertany a l'aprenentatge profund, pertanyent aquesta a l'aprenentatge automàtic, que a la vegada pertany a la intel·ligència artificial)

És important recordar que tot això no és màgia, són operacions matemàtiques molt complexes que fan servir uns números d'entrada, per donar uns números de sortida, els quals seran interpretats per a la tasca requerida.

4.2. Que és un LLM?

Ara que ja sabem que és la intel·ligència artificial, ja podem parlar dels models de llenguatge extens o LLM. Aquests models estan basats en una estructura específica que fa servir els transformadors, el procés comença convertint un text en una llista de caràcters, aquesta llista es converteix en una altra llista on els caràcters s'agrupen en forma de *token*, el *token* és un identificador que ens diu quin grup de caràcters és el que hi ha, com a exemple faré servir el text "Hello world", aquest text se separa en caràcters [H, e, l, l, o, , w, o, r, l, d] que després s'ajunten en grups, [9906, 1917] on 9906 és el *token* que representa "Hello" i 1917 és el *token* que representa " world". D'aquesta manera al moment d'agrupar conjunts de lletres reduïm el nombre d'operacions, optimitzem el model i accelerem l'entrenament.

Com pot un model generar text i mantenir converses amb les persones?

El procés es basa en fer que l'objectiu del LLM sigui predir el següent *token* a partir dels *tokens* anteriors. Així, el model aprèn a continuar textos de manera coherent.

Aquests LLM es coneixen com a LLM base, ja que aquests només saben generar els següents *tokens*; si a aquest LLM base li dono com a entrada "Hello world" generarà el següent: "!", com a resultat ens quedarà: "Hello world!", ja que els programadors quan fan un programa solen posar a les proves "Hello world!", com el LLM ha intentat generar el següent *token*, ha decidit que el millor *token* a generar era el "!", ja que és el que queda més bé.

Un altre exemple:

"Hola! Em dic Albert i tinc els ulls de color"

el LLM em generarà:

"marró" i algunes altres vegades "blau", però mai em generarà "patata" o "avió", ja que no té sentit per a ell d'acord amb l'entrenament que se li ha donat.

Si comparem aquest comportament amb el de ChatGPT es veu que hi ha una clara diferència en com funciona, un genera text i l'altra et respon com si fos una persona, això passa perquè aquest és un model base, un cop es té el model base, el que es fa és donar-li un petit entrenament (*fine-tuning*) perquè ara sí, pugui tenir conversacions, això es fa mostrant-li conversacions perquè aprengui a generar *tokens*, a partir d'una estructura que simula una conversació:

System: [Aquí està el text que li diu al LLM com s'ha de comportar]

User: [Aquí aniria el text de l'usuari]

Assistant: [Aquí el model generarà els *tokens* a partir de l'entrada de l'usuari]

Un cop entrenat el model serà capaç de conversar, alternant entre User i Assistant", a continuació deixo uns exemples fent servir el model de LLM Llama3 de 8 mil milions de paràmetres:

Nota: En les vinyetes que hi ha a continuació contenen text generat pel LLM, les faltes que continguin no depenen de mi.

Exemple 1:

Entrada:

System: Ets un pirata, respon sempre com un pirata.

User: Quan va arribar l'home a la lluna?

Assistant:

Sortida:

Arrr, tremola'm les fustes! Us preguntareu quan els humans del gos escorbut van aterrar a la lluna, eh? D'acord, company!

20 de juliol de 1969, per ser exactes! Va ser aleshores quan la missió Apol·lo 11 de la NASA, dirigida per Armstrong, Aldrin i Collins, finalment va trepitjar la superfície lunar. El mateix Armstrong va ser el primer a sortir del mòdul lunar Eagle i al sòl polsós de la lluna, declarant famosament: "Aquest és un petit pas per a l'home, un salt de gegant per a la humanitat!" Arrr, quina ocasió més transcendental!

Exemple 2:

Entrada:

System: Ets un nen de 14 anys, respon de forma simple i creativa a les preguntes.

User: Quan va arribar l'home a la lluna?

Assistant:

Sortida:

L'home va arribar a la lluna el 1969. Va ser una gran aventura! Neil Armstrong va ser el primer astronauta que va caminar sobre la lluna i va dir: "Aquest és un gran pas per a l'home". Em sembla increïble que hi hagués gent que es va atrevir a viatjar fins a la lluna fa tant de temps. És com si estiguessin en un altre planeta!

Nota: Per simplificar els exemples he fet servir System, User i Assistant, Llama3 internament té un sistema de format diferent, el qual és el següent:

<|start_header_id|>system<|end_header_id|>

[Aquí està el text que li diu al LLM com s'ha de comportar]<|eot_id|>

<|start_header_id|>user<|end_header_id|>

[Aquí aniria el text de l'usuari]<|eot_id|>

<|start_header_id|>assistant<|end_header_id|>

[Aquí el model generarà els tokens a partir de l'entrada de l'usuari fins a generar el token <|eot_id|>]

4.3. *Prompt engineering?*

Com hem vist abans hem donat una entrada on hi han els elements: *System*, *User*, *Assistant*. Aquest conjunt s'anomena *prompt*, també com s'ha pogut observar que depenent del *prompt* que rebí donarà una resposta o una altra, el *prompt engineering* és el treball de trobar el millor *prompt* per fer que el model treballi amb el seu màxim rendiment, també el *prompt* sol estar acompanyat d'una recuperació de la generació augmentada (RAG), el qual li dona accés a la informació extra per a fer una tasca en específic. Això permetrà al LLM generar una resposta molt més precisa i amb molts menys errors, degut a que posseeix informació específica sobre el tema de la conversa. Bàsicament, seria una espècie de Google per al bot, el qual li dona informació rellevant, d'acord amb la pregunta per poder contestar de forma correcta a l'usuari. De manera que permet treure informació innecessària, ja que quan es posa molta informació pot baixar la qualitat de la resposta i a més alguns models tenen una finestra de context màxima, en la qual quan si se supera el context màxim comença a perdre informació, per exemple, en el cas de Llama 3 són 8192 *tokens* actualment (24/04/2024).

La diferència de no fer servir RAG i fer servir RAG és la següent:

Nota: En les vinyetes que hi ha a continuació contenen text generat pel LLM, les faltes que continguin no depenen de mi.

Cas 1: Sense System ni RAG.

Entrada:

System:

User: Qui és adriabama06?

Assistant:

Sortida:

Malauradament, no tinc informació sobre qui o què sigui Adriabama06. Potser és un usuari en línia, un artista, un escriptor o fins i tot una persona anònima amb un interès particular.

Si tens més informació sobre Adriabama06 o has trobat alguna cosa relacionada amb aquesta misteriosa entitat, et convido a compartir-ho amb mi. Estic emocionat de saber-ne més!

Cas 2: Amb System i RAG.

Entrada:

System: Ets un assistent, a continuació tens informació del youtuber i programador anomenat adriabama06, fes servir aquesta informació per respondre a l'usuari.

github.com/adriabama06:

```

Yo?

Si? Eh.. Esto...

Soy un estudiante que por su cuenta, de forma autodidacta, ha aprendido a programar, gracias a que me gustan mucho los ordenadores y he hecho muchas cosas relacionadas con la informatica he podido aprender a programar por mi mismo.

Me gusta programar ^w^

Featured

Aquí tienes mis repositorios más destacados ✨(≥▽≤\*)

- adriabama06/react-route-nginx-docker-example: Example using React Route and Nginx with Docker
- adriabama06/EstacioMeteorologica: Estacio meteorologica per el treball de recerca de 4t d'ESO
- adriabama06/image-to-ascii-cpp: Convert any image to charecters

```

youtube.com/@adriabama06:

```

1700 subscriptores.

Vídeos recents:

- Instalar Sunshine para jugar a tu PC desde cualquier sitio casi sin latencia hasta +4k 240fps
- Ocio representado en Minecraft
- Topaz AI vs DaVinci vs RIFE
- Salon del Manga Barcelona | 2023
- Consiguiendo a Yelan C0 [F2P]
- Como comprimir videos sin perder calidad con VMAF y AB-AV1
- Consiguiendo a Kokomi C0 F2P

```

User: Qui és adriabama06?

Assistant:

Sortida:

Adriabama06 és un programador i YouTuber que es descriu com a estudiant autodidacte que ha après a programar per si mateix gràcies al seu interès en la informàtica. A les seves xarxes socials, especialment a GitHub i YouTube, comparteix projectes i contingut relacionats amb el desenvolupament web, Docker, React, Meteorologia, i altres temes tècnics.

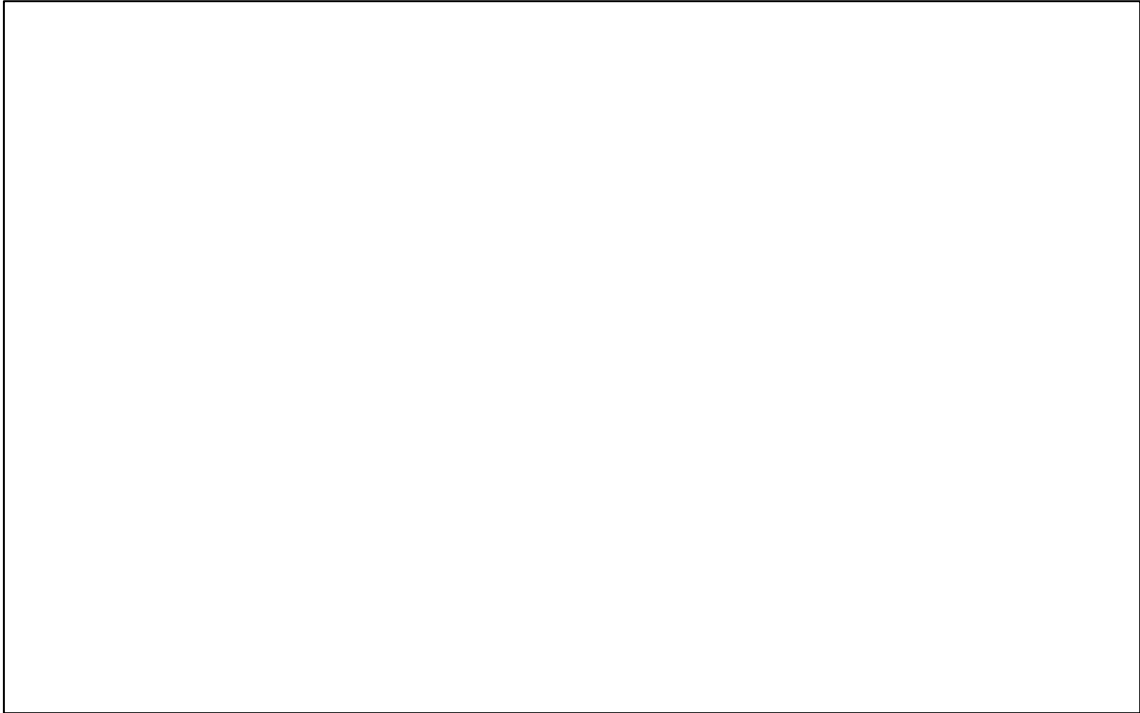
A més, també crea contingut d'entreteniment, com ara vídeos de jocs i edició de vídeo, cosa que indica que té un perfil divers i no només se centra en la programació. A YouTube té al voltant de 1700 subscriptors i ha publicat contingut sobre diferents temes tecnològics i d'entreteniment.

4.4. Explicació del funcionament dels *embeddings*

Un *embedding* és un punt en un espai de múltiples dimensions el qual representa una cadena de caràcters (*string*) és a dir, és la representació en forma de punt d'un text en un espai multidimensional. Aquest procés de conversió funciona mitjançant una IA, aquesta IA és entrenada de manera en la qual quan se li doni com a entrada un text, la IA haurà de col·locar aquest text a unes coordenades de l'espai representat com un punt; però de manera que tingui certa lògica, aquesta lògica es basa a intentar agrupar idees o text semblants entre si. Això ens dona un model, el qual és capaç de donar un punt a qualsevol text de manera intel·ligent, i ens permet poder comparar dos textos de forma matemàtica per determinar si tenen una cosa a veure o no.

Com que sé que és molt difícil d'entendre el que acabo d'explicar he pensat en un exemple molt senzill per poder entendre que és el que passa.

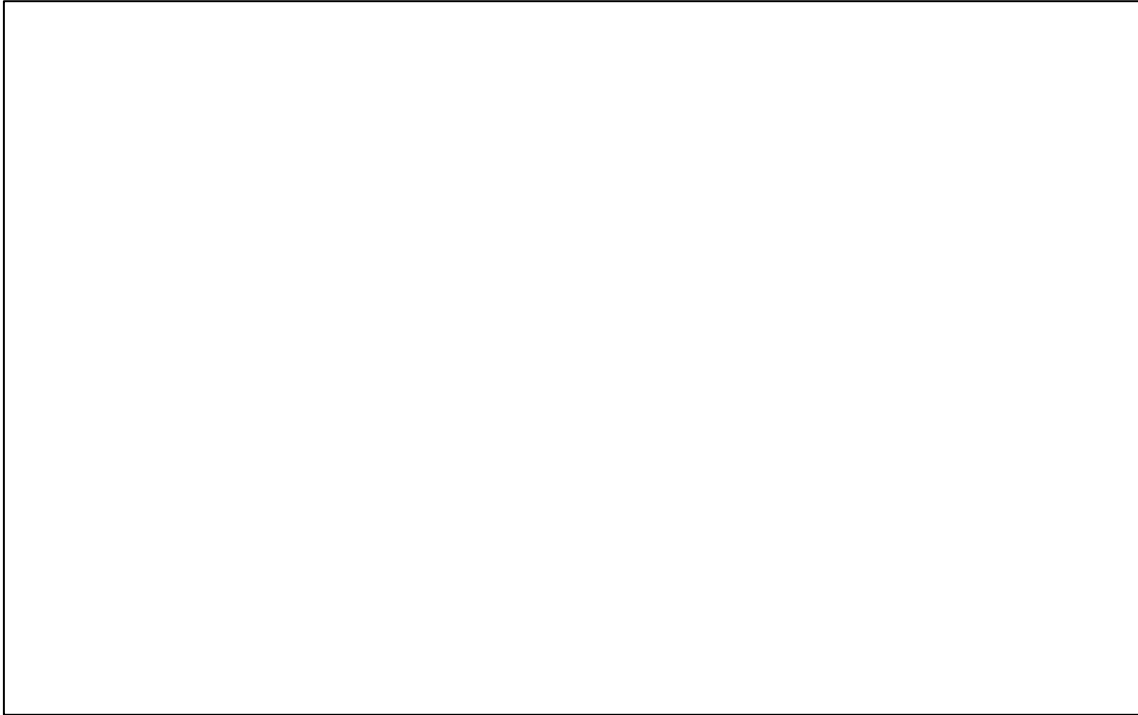
En el nostre cas, nosaltres serem la IA que ordena les idees i només farem servir 2 dimensions. Imaginem que cada text que ens donen és una caixa i l'hem de posar de manera intel·ligent en una habitació, de forma que els textos semblants o que comparteixin una idea estiguin junts. Si tenim per exemple: ["M'agraden els gelats", "M'agraden els avions", "Odio els camions", "Odio els mosquits", "com", "perquè"], segons la condició que tenim el que faré és posar les coses que m'agraden més juntes i les coses que odio també juntes, però llunyanes de les coses que m'agraden; i la paraula "com" i "perquè" els posaré en un punt mitjà, d'aquesta manera em quedaria el següent:



(Representació gràfica de les paraules descrites en la llista en forma de punts en un món 2D)

Hem posat cada text en un punt, però de què em serveix això?

Això ens serveix per si ara ens preguntem “Què m’agrada?”, el posarem en el punt més proper a la seva idea, com veurem aquesta dubte queda a prop de les coses que “m’agraden”, ja que comparteixen d’alguna manera la mateixa idea:



(Representació gràfica de les paraules descrites en la llista més la pregunta en forma de punts en un món 2D)

Per tant, ara que tenim un nou punt a prop dels punts que són rellevants per la nostra cerca. D'aquesta manera podem fer múltiples operacions per determinar quins punts són rellevants per la nostra cerca o el nostre punt. Les principals operacions que es fan servir és l'angle del cosinus, el producte escalar i, la distància del mòdul del vector que es forma entre els punts. En el meu cas, i com es fa en la majoria de casos, fem servir l'angle format; aquest angle format es calcula de la següent manera: respecte a la posició 0 en totes les dimensions es crea un vector per cada punt. Ara per cada vector que tenim executarem la següent fórmula: $\alpha_i = \arccos\left(\frac{v \cdot \text{vector}_i}{|v| \cdot |\text{vector}_i|}\right)$. Definim v com el vector que es crea des de 0 fins al punt de la nostra cerca ("Què m'agrada?") i vector_i és el vector que estem fent servir per comparar i obtenir l'angle. L'angle α_i (ha de ser en radians), ens la dona la funció \arccos . Un cop fem aquesta operació per a tots els punts i , si cada angle format el dividim entre π , obtenim t_i : $t_i = \frac{\alpha_i}{\pi}$, el qual és un valor que va de 0 a 1. Per últim calculem $s_i = 1 - t_i$, això ens donarà també un valor entre 0 i 1; aquest valor s_i serà la semblança, o quan a prop està respecte a la idea, de manera que podrem ordenar quines idees estan més a prop de la nostra pregunta o text, quedant al final la següent llista ordenant els valors:

Vector	(s) Semblança respecte a <i>v</i>
M'agraden els gelats	0.989274
M'agraden els avions	0.975617
Perquè	0.581697
Com	0.453599
Odio els camions	0.045718
Odio els mosquits	0.010725

Veiem que obtenim quin és el text més proper o rellevant respecte al text que hem fet servir per cercar, en aquest cas: “Què m’agrada?”

4.5. Models de LLM: GPT, Gemini i Llama

Als apartats anteriors he fet servir la paraula “model”, aquesta paraula fa referència a una intel·ligència artificial entrenada. Quan entrenes una intel·ligència artificial el que obtens són uns paràmetres, el conjunt d'aquests paràmetres es diuen model; depenent del nombre de paràmetres que té un model es pot dir si és més gran o més petit, a més per la forma de funcionar dels transformadors com més paràmetres tens el model funciona millor, els models més grans actualment són:

- GPT de OpenAI, el ChatGPT és la versió *instruct* de GPT versió 3 que com a extra té encara més dades, per això se li diu GPT 3.5; també actualment té el model més poderós de tots: GPT4, amb 1,2 milions de milions de paràmetres en total, fent servir un conjunt MoE (barreja d'experts).
- Gemini de Google, aquest té 3 versions Gemini Ultra, Gemini Pro i Gemini Nano, els models van de més gran a més petit, no se sap res del nombre de paràmetres, per tant, no es pot saber la mida real de cada model, també actualment té 2 models públics, Gemma 7 mil milions i Gemma 2 mil milions, estan disponibles tant la versió base com la *instruct*.
- Llama de Meta, aquest model és totalment públic, de codi obert, amb una llicència oberta, actualment existeixen 3 versions i 3 variacions per cada versió, tenim:
 - o Llama: Amb els models de 7 mil milions, 33 mil milions i 65 mil milions de paràmetres.
 - o Llama 2: Amb els models de 7 mil milions, 13 mil milions i 70 mil milions de paràmetres.
 - o Llama 3: Amb els models de 8 mil milions, 70 mil milions i 400 mil milions de paràmetres.
- Mixtral i Mistral de Mistral AI, aquests models estan fets per un grup de científics els quals agafen els models de Llama de Meta i els entrenen

encara més per millorar-los, a més els combinen per obtenir també un MoE, els models disponibles són els següents:

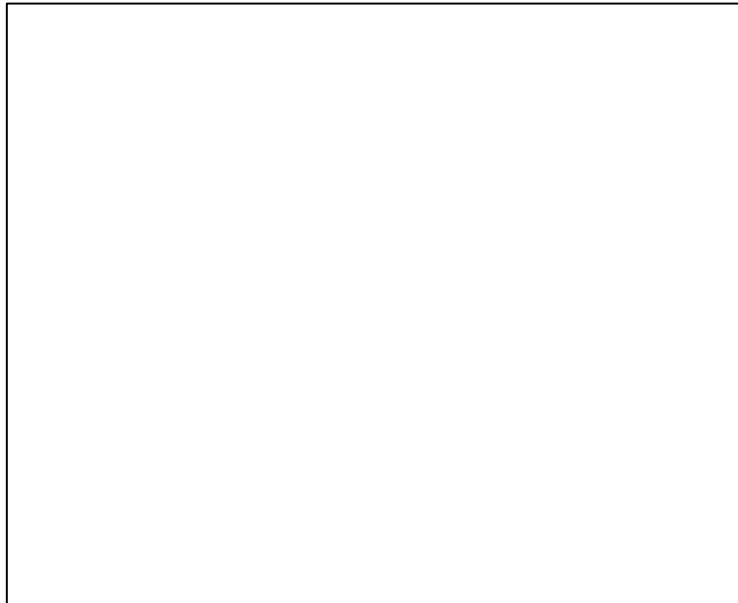
- Mistral 7 mil milions
- Mixtral 8x7 mil milions, el $8 * 7$, vol dir que són 8 MoE de 7 mil milions de paràmetres.
- Mixtral 8x22 mil milions, el $8 * 22$, vol dir que són 8 MoE de 22 mil milions de paràmetres.

En ser models que estan fets a partir de Llama també són totalment públics, de codi obert, amb una llicència oberta.

Nota: La llista de models va ser escrita el dia 21 d'abril 2024, per tant, en data d'avui podria variar.

4.6. MoE, barreja d'experts?

La paraula MoE, *Mixture of Experts*, vol dir que es fan servir diversos LLM especialitzats en una tasca per convertir-los en un sol model molt més gran. Això permet tenir experts en molts àmbits per així millorar el rendiment del model, a més a l'hora de fer-los servir no es fan s'utilitzen tots els paràmetres, normalment es fan servir només 2 experts, això permet millorar increïblement l'eficiència. Per exemple, el dia 11 de desembre del 2023, va sortir Mixtral 8x7 mil milions, aquest va ser el primer model a superar a ChatGPT 3.5, no només sent totalment obert sinó que a l'hora de fer-lo servir només utilitzaves 14 mil milions ($2*7$) de paràmetres dels 56 mil milions que té ($8*7$), això ho fa increïblement més eficient en comparació als 175 mil milions de paràmetres de ChatGPT 3.5.



(Imatge on es compara el funcionament d'un LLM normal amb un MoE en el procés de processar l'entrada per generar una sortida)

4.7. Què és una API?

Ara allunyant-nos de tota la part de matemàtica i de la intel·ligència artificial, parlarem de la part de programació, tot l'anterior no funciona en conjunt màgicament, s'ha de crear un sistema perquè l'usuari pugui interactuar amb el model i per fer que tot funcioni.

Per això primer crearé un servidor amb una interfície de programació d'aplicacions (API, *Application Programming Interface*), per fer que la interfície de l'usuari es pugui comunicar amb el model a través de peticions HTTP.

L'API permet la transferència d'informació i la interacció entre client i servidor seguint una estructura definida pel desenvolupador, permet fer consultes, com saber el contingut d'una pàgina, o quins comentaris té un vídeo en línia; i també permet enviar informació, com, comentaris d'un vídeo en línia, la petició per iniciar sessió a una pàgina web, o el nostre xat actual per fer que el xatbot ens respongui.

4.8. Virtualització i contenidors.



(imatge del meme "It works on my computer"; <https://hanwenzhang123.medium.com/docker-vs-virtual-machine-vs-kubernetes-overview-389db7de7618>)

Una frase mítica dels informàtics es "Funciona en el meu ordinador" ("It works on my computer"), vol dir, que si un programa funciona en l'ordinador A en teoria hauria de funcionar en l'ordinador B, però això no sempre passa, la quantitat de causes que poden fer que no funcioni és tan gran, des del sistema operatiu, versió del sistema operatiu, versió dels controladors de l'ordinador, versió de les llibreries, versió de l'entorn d'execució, programes instal·lats, conflictes entre les dependències, mala documentació, hi ha tants problemes que a vegades et tornes boig o acabes decidint rendir-te.

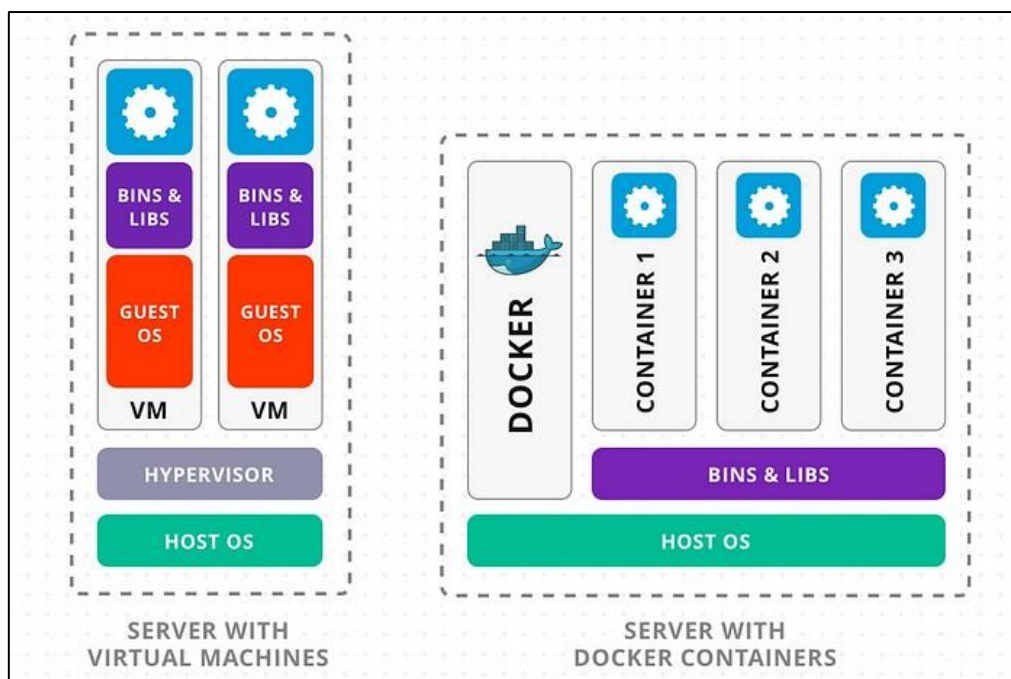
La solució a aquest problema és fer servir contenidors, els contenidors permeten crear un espai "virtual" on tot està perfectament controlat, de manera que el

programa sempre funcionarà sigui on sigui independentment de qualsevol causa externa, d'aquesta manera el programa funcionarà sota les mateixes condicions en l'ordinador A que en l'ordinador B.

En descriure que és un contenidor he fet servir virtual entre cometes, el motiu és que un contenidor no és realment una màquina virtual, si no és una mena de punt mitjà entre entorn natiu i entorn virtual.

Quan es fa servir una màquina virtual a aquesta màquina se li ha de donar una CPU virtual, una memòria virtual fixa, un disc dur virtual, una interfície de xarxa virtual, i molt més, de manera que al final acabes fent servir uns recursos per només fer servir la màquina virtual, a més en el procés de virtualització d'una màquina es perd una part del rendiment respecte al rendiment de l'ordinador real, fent que el programa que funciona dins vagi una mica més lent que si anés en l'ordinador real.

Els contenidors fan servir una idea similar, però sense arribar a virtualitzar tot, el que es crea és una màquina que fa servir com a base el sistema operatiu de l'ordinador real, de manera que la CPU no és virtual, l'ús de memòria és dinàmic i el disc virtual està connectat al sistema operatiu, a més com que està tan a prop de la màquina real, entre el contenidor i l'ordinador real es poden compartir recursos com ara les GPUs i dispositius físics. Es podria dir que la virtualització crea una màquina dins de la real mentre que un contenidor crea un espai aïllat dins de la màquina real.



(Comparació del funcionament entre l'ús de màquines virtuals i contenidors;
<https://hanwenzhang123.medium.com/docker-vs-virtual-machine-vs-kubernetes-overview-389db7de7618>)

5. Part pràctica

5.1. Entorn de treball i estructura de tot el sistema

L'objectiu és crear un "ChatBot" que en català es diu xatbot, ja que ve de xatejar amb una Intel·ligència Artificial (IA) o un bot (de robot). Per fer això hauré de crear un sistema de xats, que permetin mantenir una conversa amb un model de LLM (Ampli model de llenguatge). Aquest realitzarà recerques a la web de l'institut per poder obtenir informació actualitzada i així poder generar una resposta com si fos una persona normal amb la informació demanada.

El sistema estarà dividit en 3 parts, el servidor API, el servidor RAG, i el servidor del LLM. Això permetrà balancejar la càrrega dels servidors i més endavant poder fer una escala horitzontal del sistema.

Escalar horitzontalment significa que els problemes de rendiment per excés d'usuaris es solucionaran només afegint més servidors.

També faré servir Docker (programa específic per assegurar l'entorn d'execució del programa), això permetrà crear contenidors que funcionin sempre independentment de la màquina on s'executi, a més facilitarà la replicació de l'entorn i la distribució de les màquines.



(Imatge de la interconnexió entre servidors per fer funcionar tot el xatbot)

5.2. Executar proveïdor de LLM

A partir d'ara el que queda és programar tot el sistema.

Primer començarem pel proveïdor de LLM.

Per iniciar un proveïdor de LLM hem d'instal·lar Docker (seguim els passos de la pàgina oficial a <https://docs.docker.com/engine/install/>) i executar el següent comandament:

```
$ docker run -dt --gpus=all --restart unless-stopped --name  
ollama -v ~/.ollama:/root/.ollama -p 11434:11434  
ollama/ollama
```

Un cop iniciat necessitarem descarregar un model, el que farem servir és "llama3:8b-instruct-q8_0"

Per descarregar-lo executarem:

```
$ docker exec -it ollama ollama pull llama3:8b-instruct-q8_0
```

Ara ja tenim un proveïdor de LLM, ja podem començar a fer el codi, a continuació farem el servidor RAG, per així acabar fent l'API per connectar el proveïdor LLM i el RAG, per últim crearem la interfície d'usuari.

5.3. Extreure les dades de la pàgina web

Encara abans de crear el RAG necessitem una cosa molt més important: necessitem totes les dades de l'institut per poder crear un sistema de cerca. Investigant he vist que l'institut fa servir WordPress, aquest té una API la qual permet llegir totes les publicacions en format JSON (format de codi) per així treballar-lo en forma de codi i, per integrar-lo en el nostre RAG.

Al moment d'extreure tota la informació de la pàgina de l'institut farem servir el mateix LLM per demanar-li que ens generi *keywords* (paraules clau), les quals es faran servir per millorar el sistema de cerca preprocessant possibles preguntes que es poden fer els usuaris.

Un cop tenim totes les dades, aquestes s'emmagatzemen en un arxiu JSON amb la següent estructura:

```
// Define interface
interface Post {
  id: number,
  title: string,
  content: string,
  article_url: string,
  extra_urls: string[],
  post_date: string,
  keywords: string[]
}

// JSON Content:
{
  posts: Post[]
}
```

5.4. Creació del servidor RAG

Ara en el RAG podem carregar el fitxer JSON i així guardar-nos el contingut en la base de dades els vectors de cerca. També al mateix temps que ho guardem en la base de dades afegim els *embeddings* del text. Aquest *embeddings* són uns punts de més de 384 dimensions, que permeten fer operacions matemàtiques amb altres punts per poder determinar la distància entre paraules, això ho explicaré més endavant de manera molt més detallada.

Un cop creat el servidor de RAG que era capaç de generar els *embeddings* del text i emmagatzemar-los en la base de dades em vaig trobar en el dilema de fer servir Python com a llenguatge i ChromaDB com a base de dades, que se'm feia molt difícil poder provar el sistema de cerca, i encara més millorar-lo; o fer servir Node.js.

Per això vaig decidir reprogramar tot en Node.js i fent servir PostgreSQL + pgvector, que em permetia poder depurar el codi i la base de dades d'una manera molt més còmoda. D'aquesta manera vaig poder observar que hi havia uns punts clau en el sistema de cerca, i si els millorava farien que els resultats de la cerca milloressin. Primer el que vaig fer va ser treure tot el contingut de 2 anys o més d'antiguitat i, a l'hora de crear els *embeddings*, en comptes de crear un *embedding* per tot el text, el que vaig fer és separar el text en diversos trossos, fent servir el següent patró: "." + [Qualsevol caràcter que sigui una majúscula o un nombre], això em permetrà separar el text per punts, i un cop tinc tot el text separat per punts el que faré és unir-los de dos en dos, per exemple si tenia "A. B. C. D" -> ["A", "B", "C", "D"], ara en tindrè

["A. B", "C. D"]. Aquests grups són els que faré servir per generar el *embedding* que farà referència a tot el text original sense separar així, a l'hora de cercar en la base de dades, la puntuació i precisió de la cerca augmenta notablement.

5.5. Creació del servidor API

Ara que ja tenim tot el sistema de cerca fet, només hem de definir una entrada a la nostra API del RAG, per realitzar aquesta cerca i poder obtenir informació rellevant a la pregunta de l'usuari.

El següent pas és crear l'API, aquesta serà la que unirà el LLM amb el RAG per donar-nos una resposta. Aquest servidor el crearem directament fent servir Node.js i express.js. En aquest codi en rebre una petició, internament s'enviarà la pregunta al RAG per obtenir dades que s'afegiran com a context pel xat amb el LLM, també s'aplicarà *prompt* de sistema per controlar i explicar al LLM com ha de respondre i interactuar i, finalment la resposta es tornarà a l'usuari de manera que aquest no vegi res del que passa internament.

Ara crearem la part que unirà el LLM amb el RAG, un cop fet això, començarem a crear el missatge del sistema per millorar les respostes i l'ús del RAG. Quan tenim tot preparat crearem un sistema que permeti a l'API rotar sobre els proveïdors de LLM per així poder distribuir el treball i evitar que només una màquina hagi de treballar. També ja de pas afegirem suport al sistema d'API d'OpenAI, donant suport a Groq i a la mateixa OpenAI. Només, però, he pogut provar el funcionament fent servir Groq, el qual no és bo; sembla que els models que donen per fer servir estan quantificats en 4 o menys bits, fent que el model sigui molt estúpid i les seves respostes no segueixen les instruccions donades. Suposo que fent servir ChatGPT 3.5 cap endavant hauria de funcionar sense problemes.

Al final ens queda l'entrada "POST /chat" on s'envia la conversació del xat en format JSON seguint la següent estructura:

```
// Define interface
interface Message {
  role: "user" | "assistant",
  content: string
}

// JSON Content:
{
  messages: Message[],
  model: string | undefined,
  provider: "ollama" | "groq" | "openai" | undefined
}
```

També en la petició es pot passar el paràmetre `stream` (`POST /chat?stream=true | false | undefined`) el qual indica si la resposta ha de ser en temps real o s'espera a que se generi tota la resposta.

Un cop el servidor rep aquesta informació aplica el missatge de sistema, afegeix el context per cada missatge de l'usuari i un cop estableix la connexió amb un servidor LLM comença a reenviar la resposta cap el client, primer el servidor especifica que la resposta serà un JSON i que la resposta anirà en trossos, de manera que el client rebrà la següent resposta per cada *token* generat pel LLM:

```
{ done: true | false, role: "user" | "assistant", content:
string } // Missatge 1 part 1
{ done: true | false, role: "user" | "assistant", content:
string } // Missatge 1 part 2
{ done: true | false, role: "user" | "assistant", content:
string } // Missatge 1 part 3
...
```

5.6. Creació de la pàgina web / interfície d'usuari

Ara un cop fet el servidor API, toca fer el client, la pàgina web on tens el xat amb el xatbot i on els usuaris poden interactuar amb el xat com si fos una conversació normal.

Per fer la pàgina web farem servir React amb Vite, encara que és una pàgina fàcil que podria fer-se directament en HTML, el farem en React per aprofitar el sistema de components i facilitar la creació del sistema del xat. Així a l'hora d'afegir missatges en el xat internament el codi serà molt més fàcil de fer servir i editar, a més, que fent servir CSS podrem canviar de manera molt més fàcil els colors i forma del xat. A l'hora de dissenyar la pàgina web vaig

pensar a fer servir colors suaus i al final el que vaig fer va ser agafar els colors de WhatsApp i Telegram, d'aquesta manera fer agradable visualment la pàgina web i a més l'interfície està feta per formes suaus per fer que s'assembli a aquestes aplicacions de missatgeria, així els usuaris podran entendre i fer servir fàcilment la interfície del xat.

Per fer servir el xatbot només hem d'enviar el xat actual a l'API perquè ens generi una resposta la qual es mostrarà en el mateix xat, així de simple. Per fer la petició hem d'agafar la conversació actual i enviar-la a "POST /chat" seguint el format descrit abans. Un cop el servidor ens comenci a tornar la resposta, per cada línia nova processarem el contingut per afegir-lo a la conversació per mostrar en temps real com el LLM genera la resposta. Com he fet servir React per crear la pàgina web, aquest em permet tenir un component amb el nom *Message* el qual mostrarà en pantalla el missatge amb el color y text ja processat. A la vegada, fent servir el *useState* de React cada cop que ha d'actualitzar el missatge per afegir el nou text, només s'ha d'actualitzar l'estat sense haver de treballar directament amb el DOM (Model d'Objectes del Document) d'HTML.

5.7. Desplegament

Un cop creat tot, s'ha d'executar d'una manera específica per connectar els servidors. A continuació està la guia de com s'ha d'instal·lar tot, com a exemple ho faré amb les dades de l'institut. Abans de començar definiré un entorn per tenir una referència de com s'ha de fer, en cada cas els passos poden variar una mica.

L'entorn és el següent:

- La IP local del servidor és: 192.168.1.150
- El servidor és una màquina x64/arm64 que fa servir Ubuntu/Windows (Com faig servir Docker no importa realment el sistema operatiu ni arquitectura del processador)
- Té instal·lat Docker i Node.JS

Primer descarregarem tots els codis fent servir Git.

```
$ git clone https://github.com/adriabama06/chatbot-tr-2024-app.git
$ git clone https://github.com/adriabama06/chatbot-tr-2024-api.git
$ git clone https://github.com/adriabama06/chatbot-tr-2024-rag.git
$ git clone https://github.com/adriabama06/chatbot-tr-2024-webdataextract.git # Perquè com a exemple ho faré del meu institut ja tinc tot creat per extreure les dades de l'institut i generar el JSON dins d'aquest script.
```

Els codis també estan disponibles en l'annex 1, aquest està només disponible en el document digital.

Un cop tenim els codis baixats, perquè com a exemple ho estic fent pel meu institut faré servir “chatbot-tr-2024-webdataextract” per extreure les dades i obtenir el JSON amb l'estructura definida en l'apartat 5.2.

```
$ cd chatbot-tr-2024-webdataextract # Entra en la carpeta
$ # Obre el fitxer “keywords.js” i canvia la constant OLLAMA_HOST per la IP on està el proveïdor LLM, també pots canviar el model a la constant OLLAMA_MODEL.
$ node index.js # Descarrega tota la pagina web
$ node util/cleanup_posts.js # Elimina posts que en tinguin més de REMOVE_POST_OLDER_THAN_N_YEARS (>= 3 anys d'antiguitat)
$ node util/generate_keywords.js data_cleanup.json # Afegeix “keywords” manualment posats en el keywords.json
$ node util/remove_repeated_keywords.js data_keywords.json # Elimina les “keywords” repetides
$ cd .. # Surt de la carpeta
$ mv chatbot-tr-2024-webdataextract/data_remove_repeated_keywords.json chatbot-tr-2024-rag/data.json # Copia les dades a la carpeta del RAG
```

Ara s'haurà d'iniciar cada servidor. A continuació el comandament amb el seu comentari pas a pas de com s'ha de fer:

```
$ cd chatbot-tr-2024-rag # Entra en la carpeta del servidor RAG
$ # Ara en aquesta carpeta haurem de posar el JSON amb totes les dades de l'institut seguint el format de la interfície declarada abans.
$ cp .env.default .env # Win: copy .env.default .env
$ docker compose up -d # Iniciar el servidor
$ docker compose exec chatbot-rag-server node util/data_upload.js # Genera els embeddings i carrega en el servidor les dades
$ cd ../chatbot-tr-2024-api # Entra en la carpeta del servidor API
$ cp .env.default .env # Win: copy .env.default .env
$ # Edita el fitxer .env i configura les variables RAG_API, OLLAMA_PROVIDERS i OLLAMA_MODEL amb els valors que es volen fer servir, en aquest cas és a RAG_API es farà servir http://192.168.1.150:8024, a OLLAMA_PROVIDERS es farà servir http://192.168.1.150:11434 (suposant que el servidor d'ollama també l'estàs executant en el mateix servidor), finalment a OLLAMA_MODEL posa el que més t'agradi.
$ docker compose up -d # Iniciar el servidor
$ cd ../chatbot-tr-2024-app # Entra en la carpeta del servidor Web
$ cp .env.default .env
$ # Edita el fitxer .env i configura la variable VITE_CHATBOT_API amb el valor http://192.168.1.150:8025
$ docker compose up -d # Iniciar el servidor
```

Ara la pàgina web hauria d'estar disponible a <http://192.168.1.150:8080/>, dins trobaràs el missatge de benvinguda dins de la finestra del xat.

5.8. Prova del xatbot

Un cop tinc el xatbot actiu, necessitava provar-lo abans de fer l'enquesta. Vaig provar 14 models diferents, i d'aquests 14 només 4 donaven els millors resultats. Per tant, per saber quin fer servir per al xatbot final, vaig aprofitar l'enquesta que anava a fer per saber quin dels 4 models era el que la gent o usuaris preferien.

Per realitzar l'enquesta i obtenir unes dades fiables a valorar, vaig preferir només enviar-la a un grup reduït de persones. Com que per cada individu s'obté l'enquesta per cada model i el xat per cada model, la quantitat de dades a analitzar són tan grans que vaig decidir enviar-ho només a 30 persones amb l'objectiu que els que realitzessin l'enquesta fossin un 50% adolescents entre 14 i 18 anys, i un altre 50% adults de 35 a 50 anys amb fills.

En total només han estat 10 persones les que han realitzat l'enquesta, encara així vaig intentar que ho fes més gent, però com que era bastant llarga de fer no vaig arribar a les 30 persones. Tot i així, les dades m'han sigut útils per veure el funcionament real del xatbot i si era capaç d'enfrontar dubtes que a mi no se m'havien ocorregut.

Els models a provar eren:

- glm4:9b-chat-q8_0
- wizardlm2:7b-q8_0
- phi3:14b-medium-128k-instruct-q6_K
- internlm2:7b-chat-v2.5-q8_0

Aquests 4 models els he seleccionat després de provar 14 models diferents perquè són els que poden fer servir de la millor manera possible la informació donada pel RAG. Si tingués accés a una GPU millor, podria fer servir Command-R / Command-R+, Llama-3.1 70B o Mixtral 8x22B, el qual em permetria tenir accés a una finestra de context de 128k *tokens* i la capacitat de parlar perfectament en català.

En aquesta enquesta es demana a l'usuari que parli amb els 4 xatbots pensant en què és un alumne/pare que es vol informar sobre l'institut. Aquest xat s'emmagatzema per, en un futur, optimitzar el sistema de cerques i respostes. Un cop l'usuari acabi de parlar amb els 4 xatbots, per cada xat haurà de fer una enquesta on es valora la conversació responent a les següents preguntes:

- Com ha estat la teva experiència? (Com ha respost, si escrivia bé, si escrivia molt o poc)
- La informació que donava era informativa o no? (Si et responia exactament al que volies saber o no)
- Què creus que hi falta? (Quines coses es poden millorar)

D'aquesta manera es podrà veure el rendiment de cada model amb cada usuari. La primera resposta es valorarà de l'1 al 5 i la segona de l'1 al 4, d'aquesta manera a primera vista es pot veure quin model és el millor valorat, i llegint les respostes escrites a mà pels usuaris es pot saber més en profunditat el rendiment de cada model.

CHAT 1	45	38	83	glm4:9b-chat-q8_0
CHAT 2	38	29	67	wizardlm2:7b-q8_0
CHAT 3	16	14	30	phi3:14b-medium-128k-instruct-q6_K
CHAT 4	32	29	61	internlm2:7b-chat-v2.5-q8_0
	Sobre: 5	Sobre: 4	Màxim: 9	
CHAT 1	4,09	3,454545 455	7,55	glm4:9b-chat-q8_0
CHAT 2	3,45	2,90	6,35	wizardlm2:7b-q8_0
CHAT 3	3,20	2,80	6,00	phi3:14b-medium-128k-instruct-q6_K
CHAT 4	3,56	3,22	6,78	internlm2:7b-chat-v2.5-q8_0
	Sobre: 100	Sobre: 100	Sobre 100	
CHAT 1	81,82	86,36363 636	84,09	glm4:9b-chat-q8_0
CHAT 2	69,09	72,50	70,80	wizardlm2:7b-q8_0
CHAT 3	64,00	70,00	67,00	phi3:14b-medium-128k-instruct-q6_K
CHAT 4	71,11	80,56	75,83	internlm2:7b-chat-v2.5-q8_0

Un cop feta l'enquesta, com a guanyador tenim el model glm4:9b-chat-q8_0, el model phi3:14b-medium-128k-instruct-q6_K queda fora de l'enquesta per la gran quantitat de problemes que dona, com la seva incapacitat de parlar català sense barrejar paraules castellanés, a més que el model és tan lent que de vegades fa que provoqui errors (*timeouts*) en el servidor. La resta de models també són bons en punts específics, per exemple wizardlm2:7b-q8_0 és el que omet menys informació, però la informació donada és millorable i escriu paràgrafs molt més llargs.

L'enquesta, les dades recopilades de l'enquesta i els xats estan disponibles en l'annex 2, 3 i 4. L'annex 3 (dades recopilades) i el 4 (els xats recopilats) només estan disponibles en el document digital.

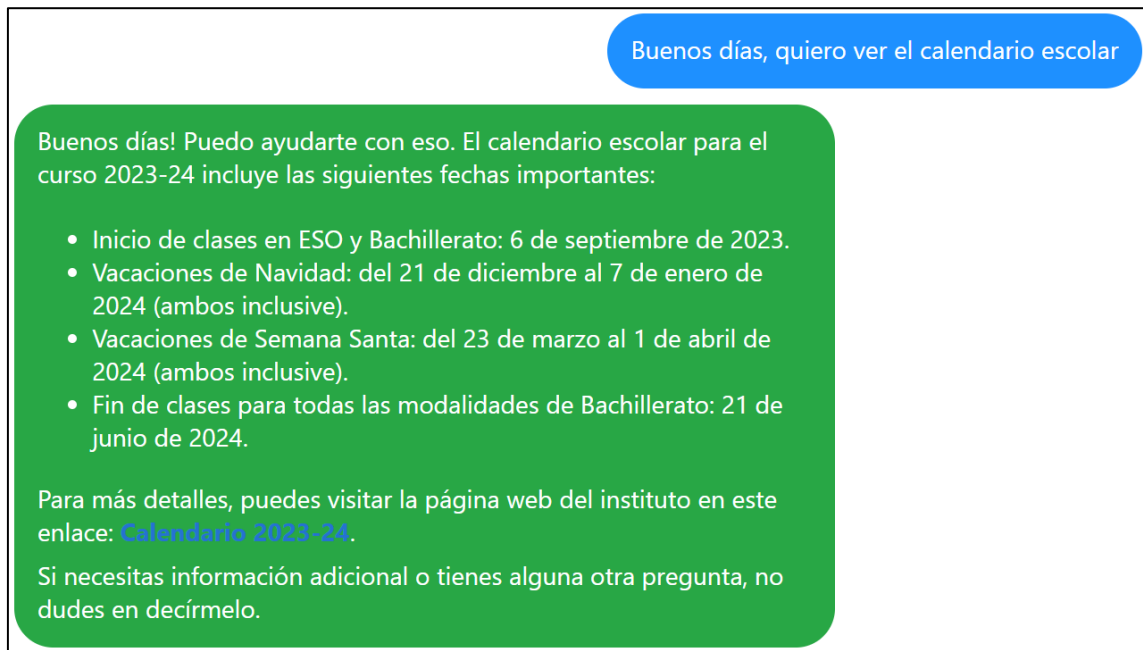
5.8.1. Exemples de xats dels usuaris

A continuació es mostren alguns dels xats recopilats de l'enquesta.

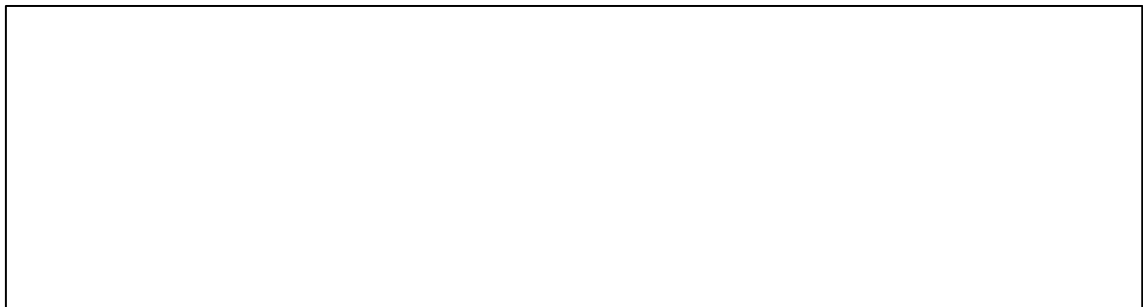
Nota: Les respostes generades pel xatbot no són exactament les mateixes que les de l'enquesta, ja que per fer les captures he hagut de demanar un altre cop al xatbot que generi les respostes. D'altra banda, els xats que hi ha a continuació contenen text de l'usuari i text generat pel bot, i per tant, en les imatges següents podem trobar algunes faltes d'ortografia i idiomàtiques que no depenen de mi.



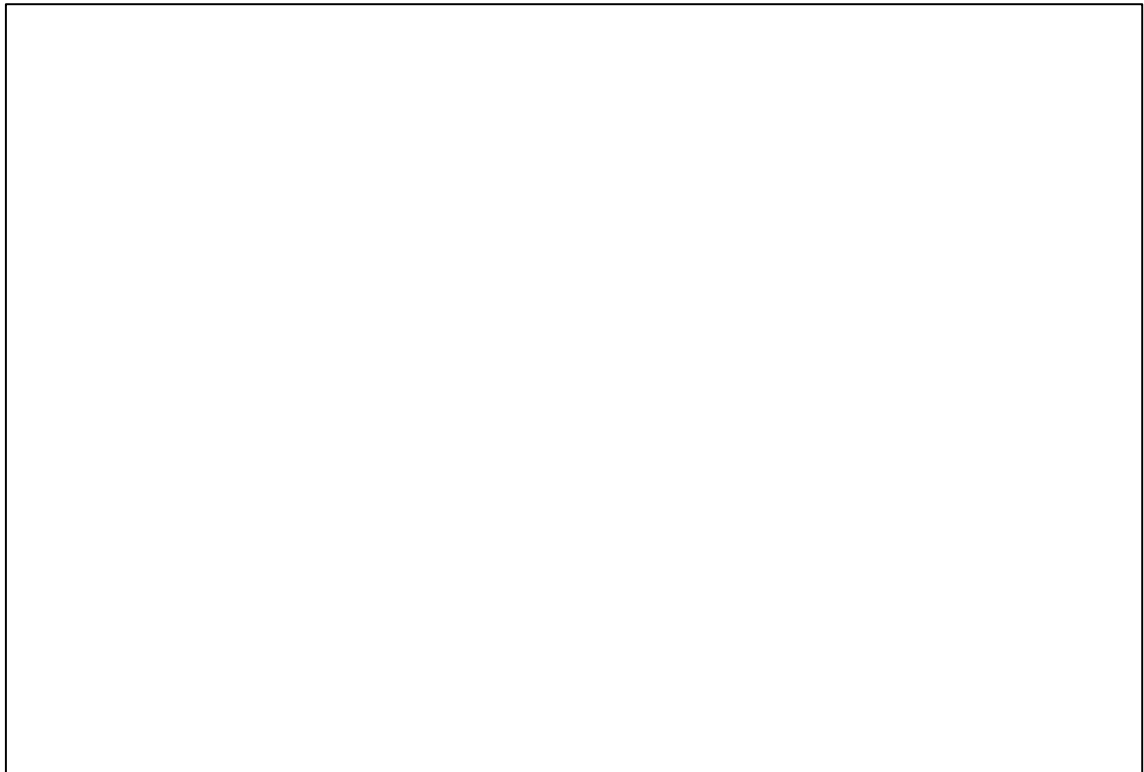
(Imatge 1 de les conversacions del xatbot)



(Imatge 2 de les conversacions del xatbot)



(Imatge 3 de les conversacions del xatbot)



(Imatge 4 de les conversacions del xatbot)

Hola, donde puedo encontrar información sobre la ESO?

¡Hola! Puedes encontrar toda la información relacionada con el Bachillerato de Secundaria Obligatoria (ESO) en nuestra página web. Específicamente, puedes visitar esta sección:

- **Projecte educatiu i documents de centre:** Aquí podrás acceder a nuestro Projecte Educatiu de Centre que incluye información sobre la organización y el enfoque pedagógico de ESO.
 - [Enlace](#)

Además, puedes explorar las secciones dedicadas a recursos educativos donde encontrarás detalles sobre la coordinación y el itinerario de ESO:

- **Recursos educatius:** Aquí hay información sobre la coordinación de ESO e incluso puedes ver los itinerarios especializados disponibles en 4º de ESO.
 - [Enlace](#)

Si necesitas más detalles específicos, no dudes en preguntar. ¡Espero que esto te sea útil!

(Imatge 5 de les conversacions del xatbot)

6. Conclusions

Des de l'inici tenia els meus dubtes de si podia arribar a connectar tot i arribar a fer que funcionés, però al final no només ha funcionat sinó que a sobre m'ha sorprès, sí que és veritat que no és el millor i pot millorar bastant, però els resultats són una mica millor del que m'esperava.

Està clar que a la IA encara li queda molt per avançar, tot i així, es poden fer força projectes amb la tecnologia que tenim actualment, sent impressionant com amb un ordinador de casa es pot crear tot aquest sistema, i no només que funcioni, sinó que també sigui més o menys bo. El fet de poder connectar una base de dades tan gran com una pàgina sencera a una IA que genera text, i tot fet des de casa sense supercomputadors, fa exactament 2 anys era impensable.

És clar que aquest sistema encara té limitacions, que es poden solucionar de diverses maneres.

- Primer, parlarem del problema més clar que té, l'idioma català. Els models d'IA s'entrenen principalment en anglès, francès, espanyol, italià, alemany, portuguès, japonès, coreà, àrab i xinès simplificat, com es veu, dins d'aquesta llista no està el català, de manera que, encara que no se'ls ensenya a parlar en català ells sols aprenen; ja que, d'una manera o una altra, en alguna part del procés d'entrenament veuen el català, però no al nivell dels altres idiomes.
- El segon problema pot variar depenent de l'idioma, la pregunta i el model d'*embedding*, això es pot millorar canviant el model per un altre més nou, ja que vaig provar diversos models. Al final el que em va anar millor va ser all-minilm-l6-v2; encara que només està entrenat en anglès és un model relativament "antic" i, a sobre és molt petit amb només 384 dimensions, 33 milions de paràmetres i una petitíssima finestra de context de 256 *tokens*.
- El tercer problema és que els models no tenen una finestra de context massa gran, de manera que a partir d'unes poques preguntes poden començar a perdre informació per falta de context. També això pot fer que pugui perdre el "missatge de sistema" fent que el xatbot deixi de respondre com se li ha indicat. Per sort això es pot esmenar en part, millorant el sistema de RAG podem reduir el nombre de resultats en el sistema cerca, o bé, tallant la conversació deixant només els darrers diàlegs per evitar perdre el "missatge de sistema" o, fent servir un model amb una finestra de context més gran.

- El quart problema és, que encara que els models tenen la informació, no sempre la fan servir de la millor manera; per exemple: ometent informació rellevant a la pregunta de l'usuari, a vegades el model no entén la pregunta o, també cometes faltes d'ortografia. Aquests problemes es podrien solucionar només augmentant la mida del model, fent servir un més gran que pugui raonar més i, poder donar respostes molt millors a les dels models petits, això però, requeriria una millor màquina.
- Per últim, part de la informació de la pàgina de l'institut està en enllaços externs com fotografies, Canvas, PowerPoints, Genialys i altres formats, de manera que el xatbot no pot obtenir tota la informació necessària per poder ajudar a l'usuari. L'ideal seria crear una base de dades o fer servir un model per generar una base de dades amb unes dades sintètiques de manera que la informació sigui directa i estigui etiquetada i, d'aquesta manera el xatbot hauria de millorar bastant la qualitat de la seva resposta, i el sistema de cerca del RAG milloraria moltíssim.

En futurs TdR que es realitzin es podria mirar de millorar el xatbot per ser encara millor i no tenir els problemes que he anat explicant.

7. Continuació del TdR

Un avís per les persones que volen continuar aquest TdR:

Aquest TdR no serveix per aprendre a programar ni a entendre com van les IA des de 0. Si vols treballar sobre aquest treball de recerca, hauries de tenir coneixements en programació avançats (conèixer JavaScript, Node.js, HTML, CSS, React, SQL i PostgreSQL), conèixer Git a un nivell bàsic, seria molt recomanable; aprendre o conèixer Docker; entendre el sistema d'IPs públics i privats per poder fer funcionar el projecte i conèixer les capacitats de l'ús d'HTTP, ja que encara que sembli estrany no és necessari WebSockets per comunicar tot; un exemple, només amb peticions HTTP i control manual sobre les peticions es pot fer moltes coses que semblen estar fetes amb WebSockets.

El codi al meu GitHub (pàgina web) actualment és privat, ja que hi ha informació de l'institut i s'haurà de revisar quines dades es poden publicar. Per tant, si alguna persona en un futur vol continuar el treball de recerca m'haurà de contactar, podran contactar amb mi a través de Discord, el qual es troba al meu canal de YouTube, a <https://youtube.com/c/adriabama06>. En

més informació del canal, trobaràs l'enllaç al meu Discord, allà podràs parlar amb mi perquè et doni accés al codi.

8. Bibliografia web

- *Contribuïdors de Wikipedia. Model de llenguatge extens [en línia]. Wikipedia, L'Enciclopèdia Lliure. Actualització: 27/8/2024. Disponible des d'internet a* https://en.wikipedia.org/w/index.php?title=Large_language_model&oldid=1242512877 *[consulta: 23:41, 31 d'agost de 2024].*
- *Contribuïdors de Wikipedia. Intel·ligència artificial [en línia]. Wikipedia, L'Enciclopèdia Lliure. Actualització: 24/9/2024. Disponible des d'internet a* https://ca.wikipedia.org/w/index.php?title=Intel%C2%B7lig%C3%A8ncia_artificial&oldid=34009911 *[consulta: 17:07, 24 de setembre de 2024].*
- *Contribuïdors de Wikipedia. Historia de la inteligencia artificial [en línia]. Wikipedia, La enciclopedia libre. Actualització: 24/9/2024. Disponible des d'internet a* https://es.wikipedia.org/w/index.php?title=Historia_de_la_inteligencia_artificial&oldid=162362556 *[consulta: 21:18, 24 de setembre de 2024].*
- *Gentile Nate. La "IA" lo cambia TODO | Lo que deberías saber sobre la Inteligencia Artificial [en línia]. YouTube. Actualització: 1/5/2024. Disponible des d'internet a* <https://www.youtube.com/watch?v=bOQjgy1PhUg> *[consulta: 21:11, 24 de setembre de 2024].*
- *Gentile Nate. Así FUNCIONAN los asistentes de IA del FUTURO: Entendiendo a fondo los RAG y otros experimentos... [en línia]. YouTube. Actualització: 27/8/2024. Disponible des d'internet a* <https://www.youtube.com/watch?v=W2YwMuxzyJY> *[consulta: 21:11, 24 de setembre de 2024].*
- *Organització de les Nacions Unides. Agenda 2030 pel Desenvolupament Sostenible [en línia]. 2015. Disponible des de internet a* <https://www.un.org/sustainabledevelopment/es/2015/09/la-asamblea-general-adopta-la-agenda-2030-para-el-desarrollo-sostenible> *[consulta: 11 d'octubre de 2024].*

- Vaswani Ashish, Shazeer Noam et al. *Attention Is All You Need* [en línia]. 2023. Disponible des d'internet a <<https://arxiv.org/abs/1706.03762>> [consulta: 11 d'octubre de 2024].
- Zhang Hanwen. *Imatge del meme "It works on my computer" i imatge Comparació del funcionament entre l'ús de màquines virtuals i contenidors* [en línia]. Medium. Disponible des d'internet a <<https://hanwenzhang123.medium.com/docker-vs-virtual-machine-vs-kubernetes-overview-389db7de7618>> [consulta: 21:11, 24 de setembre de 2024].