

Albert Garcia Bernat

**Creació d'un chatbot fent servir una arquitectura de
Retrieval Augmented Generation**

DIRECTOR: Jordi Massaguer Pla

TREBALL DE FI DE GRAU

Grau en Enginyeria Informàtica



Tarragona

Setembre 2024

RESUM

Aquest Treball de Final de Grau, té com a objectiu crear un chatbot que pugui respondre preguntes senzilles de manera natural i humana. Així que aquest projecte es pot descriure com una exploració dels Large Language Models i de quines són les tècniques que els ajuden a dur a terme les seves funcions. Concretament la Retrieval Augmented Generation acompanyada d'una aplicació desenvolupada fent servir Amazon SageMaker. L'objectiu final del projecte ha estat crear un chatbot que pugui respondre preguntes sobre un tema qualsevol com per exemple la festa major de la URV. Per a gestionar la recuperació del context s'ha fet servir una Base de Dades Vectorial anomenada Pinecone. I per a controlar la comunicació dels diferents components de l'arquitectura s'ha fet servir funcions serverless d'AWS.

Aquest treball aporta molta informació sobre que són els LLMM, com funcionen, i una de les maneres d'utilitzar-ne un per a crear una aplicació de preguntes i respostes que pugui arribar a ser utilitzat pel públic general.

RESUMEN

Este Trabajo de Final de Grado, tiene como objetivo crear un chatbot que pueda responder preguntas sencillas de manera natural y humana. Así que este proyecto se puede describir como una exploración de los Large Language Models y de cuáles son las técnicas que los ayudan a llevar a cabo sus funciones. Concretamente, la Retrieval Augmented Generation acompañada de una aplicación desarrollada usando Amazon SageMaker. El objetivo final del proyecto ha estado crear un chatbot que pueda responder preguntas sobre un tema cualquier como por ejemplo la fiesta mayor de la URV. Para gestionar la recuperación del contexto se ha usado una Base de datos Vectorial llamada Pinecone. Y para controlar la comunicación de los diferentes componentes de la arquitectura se ha usado funciones serverless de AWS.

Este trabajo aporta mucha información sobre que son los LLMM, como funcionan, y una de las maneras de utilizar uno para crear una aplicación de preguntas y respuestas que pueda llegar a ser utilizado por el público general.

SUMMARY

This Final Degree Project aims to create a chatbot that can answer simple questions in a natural and human way. So this project can be described as an exploration of the Large Language Models and what are the techniques that help them carry out their functions. Specifically the Retrieval Augmented Generation accompanied by an application developed using Amazon SageMaker. The final objective of the project has been to create a chatbot that can answer questions on any topic such as the URV's main festival. To manage context recovery, a Vectorial Database called Pinecone has been used. And to control the communication of the different components of the architecture, AWS serverless functions have been used.

This work provides a lot of information about what LLMs are, how they work, and one of the ways to use one to create a question and answer application that can be used by the general public.

Índex

1. INTRODUCCIÓ	7
1.1 Objectius	8
2. MARC TEÒRIC	9
2.1 Large Language Model	9
2.2.1 Definició	9
2.2.2 Característiques.....	10
2.2.3 Arquitectura de transformadors.....	10
2.3.1 Models de Pre-entrenament	12
2.3.2 Fine-Tuning.....	12
2.3.3 Parameter efficient fine tuning (PEFT)	13
2.3.5 Exemples d'LLM	14
2.3 RAG (Retrieval Augmented Generation)	15
3. RECERCA D'UNA SOLUCIÓ.....	16
3.1 Fine-tuning d'un model.....	16
3.1.1 Exemple de fine-tuning	16
3.1.2 Resultats del fine-tuning.....	18
3.2 Implementació d'una tècnica de Retrieval Augmented Generation	19
3.2.1 Llamaindex	19
3.2.2 Implementació de Llamaindex	20
4. IMPLEMENTACIÓ DE UNA ARQUITECTURA RAG AMB SAGEMAKER.....	23
4.1 Elements de la aplicació.....	23
4.2 Pinecone i embedder	24
4.3 Sagemaker i AWS	27
4.4 Frontend.....	28
5. RESULTATS	31
5.1 Resultats inicials	31
5.2 Millores	31
5.3 Resultats finals.....	33
6. AVALUACIÓ DE COSTOS.....	36
7. LEGISLACIÓ I PROTECCIÓ DE DADES.....	38
8. CONCLUSIÓ	40
9. REFERÈNCIES.....	41

Índex de Figures

Figura 1: Arquitectura d'un transformador	11
Figura 2: Fases d'un LLM	14
Figura 3: Arquitectura RAG	15
Figura 4: Resultats LamaIndex	21
Figura 5: Arquitectura de l'aplicació	24
Figura 6: Exemple similitud del cosinus	27
Figura 7: Visualització del frontend	29
Figura 8: Resultats inicials	30
Figura 9: Resultats finals	33
Figura 10: Mostra funcionament	34
Figura 11: Mostra de conversa	35
Figura 12: Costos AWS	36

1.INTRODUCCIÓ

En l'era actual de la informàtica la abrupta aparició de la Intel·ligència Artificial, ha iniciat una recerca de noves formes de treball i de millorar l'eficiència de les persones. Avui en dia, és freqüent trobar diferents productes quotidians que porten algun tipus d'IA incorporada amb l'objectiu de fer-los més atractius o de millorar el seu rendiment. Una de les moltes tecnologies que han sortit en aquests últims anys són els “chatbots”.

Un chatbot és un programa informàtic dissenyat per a simular converses humanes, especialment a través d'interfícies de xat. Aquests programes utilitzen una IA i el processament del llenguatge natural per a entendre i respondre a les preguntes o comandes dels usuaris de manera automatitzada. Aquests bots poden ser utilitzats en moltes aplicacions; per a oferir servei al client, assistència tècnica, vendes, educació, entreteniment i molt més.

La motivació d'aquest treball és comprendre com funciona una de les tecnologies que més ràpidament s'està estenent per la nostra societat, esbrinar quins són els LLM (Large Language Models) més efectius avui en dia, quina és la infraestructura que hi ha al seu darrere i que és necessari per a crear-ne un des de zero.

Els chatbots poden ser simples i seguir un guió predefinit, o poden ser més complexos i utilitzar algorismes d'aprenentatge automàtic per a millorar les seves respostes amb el temps, a mesura que interactuen més amb els usuaris i reben retroalimentació. Aquests últims són els més populars avui en dia, ja que les converses amb persones reals són una de les millors maneres d'entrenar un model.

En aquest context, aquest treball s'enfoca en explorar i comprendre a fons el funcionament dels chatbots, des de la seva estructura fonamental fins a les seves capacitats d'aprenentatge i millora contínua. Això no només ens permetrà desglossar la complexitat d'aquesta tecnologia emergent, sinó que també ens ajudarà a identificar les claus per a la creació i optimització de chatbots efectius i adaptats a les necessitats dels usuaris actuals.

1.1 Objectius

Aquest projecte que principalment és adreçat a persones que ja tinguin una base sobre el món de la informàtica, però que no necessàriament tinguin coneixement sobre el Natural Language Processing o la Retrieval Augmented Generation. Tots els conceptes relacionats amb la IA venen acompanyats amb una explicació per a que pràcticament qualsevol persona pugui seguir el treball de manera eficaç.

Aquest treball té diversos objectius relacionats amb la creació de chatbots i l'anàlisi de LLM. Alguns d'ells són:

- Crear un chatbot amb un Dataset personalitzat.
- Millorar el rendiment d'aquest utilitzant tècniques de Fine-tuning.
- Utilitzar tècniques de RAG (Retrieval Augmented Generation) per a donar context a un LLM perquè pugui respondre preguntes sobre un tema del qual no te coneixement.
- Utilitzar Sagemaker, Pinecone i llenguatges de Front-end per a crear una aplicació que pugui respondre preguntes relacionades amb un tema qualsevol com la festa major de la URV.
- Aprendre sobre el funcionament dels LLM i sobre l'arquitectura de les aplicacions actuals.

En aquest treball, he tingut l'oportunitat d'aprofundir en àrees de coneixement sobre les que no he tingut l'oportunitat de treballar durant la carrera. Concretament he pogut fer servir una eina per a treballar al núvol com AWS. I he tingut l'oportunitat d'aprofundir en un camp de l'IA com és el NLP, a més a més de poder veure com s'estructura una aplicació de principi a fi.

En aquest projecte, no només adquiriré coneixements aprofundits sobre el funcionament del núvol i la IA, sinó que també tindrè l'oportunitat d'aprendre en detall tot el procés de creació d'una aplicació moderna. Això inclou des de la concepció i el disseny inicial fins a la implementació, optimització i desplegament en un entorn de producció. A més, exploraré les millors pràctiques i eines utilitzades per desenvolupar aplicacions eficients, escalables i segures, aplicant principis d'enginyeria al núvol.

2. MARC TEÒRIC

2.1 Large Language Model

2.2.1 Definició

Un LLM és un tipus de model d'intel·ligència artificial dissenyat per a comprendre, generar i manipular text en llenguatge natural. Els LLM són una classe avançada de models de processament del llenguatge natural (NLP) que han estat entrenats amb grans quantitats de dades i tenen la capacitat de generar text coherent i contextualment correcte.

Un LLM es distingeix per diverses característiques que el fan especialment potent per a tasques de processament de llenguatge. En primer lloc, la mida del model és una característica molt important. Els LLM compten amb una enorme quantitat de paràmetres, alguns dels quals poden arribar a ser de l'ordre de milers de milions. Aquesta gran quantitat de paràmetres els permet capturar complexitats i matisos del llenguatge d'una manera que els models més petits no poden.

Aquest entrenament en gran escala implica l'ús de conjunts de dades textuais, que poden incloure llibres, articles, llocs web i altres tipus de contingut digitalitzat. Molt sovint es poden trobar datasets amb milers de preguntes i respostes sobre un àmbit concret que són molt útils a l'hora de crear un LLM i d'analitzar el seu rendiment. A l'assimilar aquesta vasta quantitat d'informació, els LLM desenvolupen una comprensió profunda dels patrons i estructures del llenguatge natural, que els ajudaran a generar text coherent.

Alguns dels paràmetres més importants són:

- **Pesos:** Els pesos són els paràmetres més essencials d'un LLM. Representen la força de la connexió entre les neurones en una xarxa neuronal. Durant l'entrenament, aquests pesos s'ajusten per minimitzar l'error entre la sortida del model i la resposta correcta. Són responsables de com el model interpreta les entrades i genera les respostes.
- **Biaixos:** Els biaixos són valors que s'afegeixen a les sortides de les neurones abans de passar per una funció d'activació. Permeten que el model s'ajusti millor a les dades d'entrenament.
- **Embedding Dimensions:** són representacions numèriques de les paraules o "tokens". La dimensió d'aquests vectors és important perquè determina com és de rica la representació semàntica de les paraules al model.
- **Attention Heads:** En arquitectures com Transformers, els capçals d'atenció són components dins de l'atenció multicapçal (*multi-head attention*) que ajuden a enfocar diferents parts de la seqüència d'entrada. Aquests capçals permeten que el model consideri diferents relacions entre les paraules en una frase o document.
- **Funcions d'Activació:** Aquestes funcions determinen la sortida de cada neurona. Les funcions d'activació, permeten que el model aprengui patrons complexos.

2.2.2 Característiques

La capacitat de generació és una altra característica destacada dels LLM. Aquests models no solament comprenen el llenguatge sinó que també poden generar text nou que de manera ràpida. Per exemple, poden escriure treballs, mantenir converses i respondre preguntes, a partir d'una entrada de text generada per un usuari. Aquesta habilitat per a generar text es deu als algorismes d'aprenentatge que detecten patrons i els repliquen de manera fluida.

En l'aprenentatge dels LLM existeix una tècnica molt freqüent anomenada transferència del coneixement introduïda per Jason Yosinski al 2014 en el seu article "How transferable are features in deep neural networks?". En aquesta tècnica es poden diferenciar dues parts, la de pre-entrenament, i la de *fine-tuning*. Un model es pre-entrena amb una gran quantitat de dades genèriques i variades. Per exemple, un model de llenguatge pot ser entrenat amb milers de milions de paraules provinents de llibres, articles, webs, etc. Durant aquest entrenament, el model aprèn estructures, patrons, conceptes, i relacions en el llenguatge que són útils per a moltes tasques.

Després d'aquest entrenament inicial, el model pot ser adaptat per a tasques més específiques, com la classificació de sentiments, traducció automàtica... Aquesta adaptació es fa a través d'un procés anomenat *fine-tuning*. El model ja té un coneixement general sobre el llenguatge, així que només cal ajustar-lo amb una quantitat menor de dades específiques de la nova tasca.

Pel que fa al funcionament, els LLM solen estar basats en arquitectures de *transformers*, que utilitzen mecanismes d'atenció (machine learning-based attention, Dzmitry Bahdanau, 2014) per a processar i generar text. Aquests mecanismes d'atenció permeten al model enfocar-se en diferents parts del text d'entrada, capturant dependències a llarg termini i comprnent el context de manera efectiva.

No obstant això, els LLM també enfronten desafiaments importants. Un dels principals problemes consisteix en les imperfeccions que puguin tenir els *datasets* que poden provocar que el model s'adapti massa al conjunt d'entrenament i que s'allunyi de la realitat. A més a més, el cost computacional d'entrenar i desplegar LLM és considerable, requerint recursos significatius en termes de temps i maquinari.

En resum, els Large Language Models representen un avanç significatiu en el processament del llenguatge natural, destacant-se per la seva capacitat d'entendre i generar text de manera sofisticada i adaptable. Obre la porta a la creació de moltes aplicacions relacionades amb la interpretació del llenguatge i generació de text. Encara que no sense enfrontar uns certs desafiaments tècnics.

2.2.3 Arquitectura de transformadors

L'arquitectura de transformadors, es un concepte que va ser introduït al 2017 per un grup de científics de Google en un article anomenat "Attention is all you need". En aquest article

s'introdueix un model d'aprenentatge automàtic, que va revolucionar el NLP gràcies a la utilització de mecanismes d'atenció per a gestionar dependències a llarg termini.

L'objectiu d'aquesta arquitectura és millorar el processament de seqüències de dades, com per exemple text. Permetent una millor captura de dependències a llarg termini i relacions complexes a través de mecanismes d'atenció. Essencialment, els transformadors processen totes les paraules simultàniament, permetent una major eficiència i capacitat per a capturar relacions complexes en el text.

La innovació clau d'aquesta arquitectura és l'ús d'un mecanisme anomenat auto-atenció, que permet que el model doni més o menys importància a diferents parts de la seqüència d'entrada en funció de la rellevància per a la tasca específica.

L'arquitectura consta d'un codificador o *encoder* que processa la seqüència d'entrada i genera representacions que encapsulen la informació rellevant de la seqüència. I un descodificador o *decoder*, que utilitza les representacions generades pel codificador, juntament amb la informació d'entrada pròpia, per generar la sortida.

Tant el *decoder* com el *encoder* estan composts per diverses capes "MultiheadAttention" i xarxes neuronals *feed-forward*, que sense entrar molt en detall faciliten l'entrenament en paral·lel i milloren significativament el rendiment en diverses tasques de NLP.

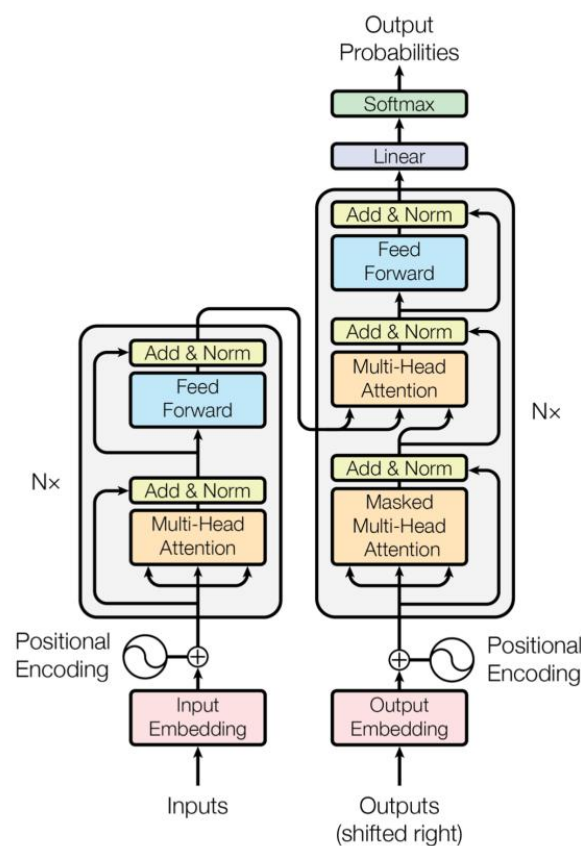


Figura 1: Arquitectura d'un transformador

2.3 Tipus de LLM

2.3.1 Models de Pre-entrenament

Els models de pre-entrenament, també coneguts com a *pre-trained language models*, s'entrenen en una fase inicial utilitzant grans volums de text no etiquetat. L'objectiu principal d'aquesta etapa és aconseguir que el model aprengui d'una manera general el funcionament del llenguatge, incloent-hi la sintaxi, la gramàtica, i el vocabulari. Durant aquest procés, el model s'exposa a una varietat àmplia de diferents situacions lingüístiques, la qual cosa li permet entendre i generar text de manera coherent i eficient.

L'entrenament en aquesta fase es realitza a través de tasques de modelatge de llenguatge que no requereixen supervisió explícita. Un mètode comú és el modelatge de llenguatge autoregressiu, que consisteix a fer que el model predigui la següent paraula en una seqüència de text a partir de les paraules que ja hagi processat. Per exemple donada una frase com “El cotxe és de color” el model podria completar la sentència amb “vermell”.

Un altre mètode és el modelatge de llenguatge emmascarat, on el model prediu paraules ocultes en una oració. Per exemple donada una frase d'entregada “El cotxe és de color [MAKS]” el model genera una llista de possibles paraules amb la seva respectiva probabilitat per a completar la frase.

Aquests mètodes permeten al model aprendre les relacions entre paraules i frases sense la necessitat de dades etiquetades específiques. Com a resultat, el model adquireix una comprensió profunda de l'estructura del llenguatge i de com les paraules es relacionen entre si en diferents contextos.

Una vegada completat el pre-entrenament, el model té una bona capacitat per a gestionar una àmplia gamma de tasques relacionades amb el processament i generació de text, encara que de manera superficial. No obstant això, perquè el model sigui realment útil i adquireixi un coneixement específic sobre un àmbit específic, sol ser necessari una nova fase anomenada *fine-tuning*. Un exemple d'un model pre-entrenat podria ser Llama 2 de la companyia Meta, que és molt útil a l'hora de generar text en anglès, però no té un coneixement profund sobre cap tema.

2.3.2 Fine-Tuning

En el pre-entrenament, el model aprèn el funcionament general d'un idioma, però per a què sigui útil en aplicacions específiques, necessita ser ajustat utilitzant conjunts de dades etiquetades i específiques per a una tasca particular. Aquest procés de *fine-tuning* permet que el model transfereixi el seu coneixement general adquirit durant el pre-entrenament a una tasca específica, millorant la seva precisió i rellevància en aquesta àrea.

Durant el *fine-tuning*, el model s'entrena amb un conjunt de dades més petit però etiquetat. Això és important perquè d'aquesta manera es pot dur a terme la supervisió necessària per a comparar la relació entre les sortides obtingudes i les sortides esperades. A més, permeten avaluar el rendiment del model i realitzar ajustaments de forma iterativa per a millorar la seva precisió. Això és fonamental per a aplicacions del món real on es prenen decisions basades en les prediccions dels models.

En alguns exemples de la utilitat del *fine-tuning* trobem: la classificació de textos, l'anàlisi d'estats d'ànim, la resposta a preguntes o la traducció d'idiomes. Per exemple, un model com Llama 2, que ha estat pre-entrenat per a entendre l'estructura del llenguatge, se li pot donar un *fine-tuning* en el conjunt de dades de Guanaco per a millorar la seva capacitat de comprendre i generar text complicat. En aquest procés, els pesos del model s'ajusten lleugerament per a optimitzar el rendiment en la tasca específica, aprofitant les representacions generals del llenguatge ja apreses durant el pre-entrenament.

2.3.3 Parameter efficient fine tuning (PEFT)

El *parameter efficient fine-tuning* es distingeix del *fine-tuning* tradicional per a maximitzar el rendiment del model amb recursos computacionals i dades etiquetades limitades. En lloc de dur a terme una exhaustiva cerca de configuracions òptimes, el PEFT és centra en estratègies que minimitzen el cost computacional i maximitzen l'eficiència del procés. Això pot incloure tècniques com la compressió de models per a reduir la seva grandària i complexitat, i la utilització d'hiperparàmetres mitjançant mètodes d'optimització més ràpids i menys costosos des del punt de vista computacional.

Els hiperparàmetres són configuracions ajustables que determinen el comportament i l'estructura d'un algorisme d'aprenentatge automàtic, però que no s'aprenen directament durant l'entrenament del model. A diferència dels paràmetres del model, que són ajustats durant l'entrenament per a minimitzar una la pèrdua del model, els hiperparàmetres són un conjunt de variables predefinides que s'estableixen abans de l'entrenament i afecten com s'aprenen aquestes variables.

A més, el PEFT destaca per la seva capacitat per a traslladar el rendiment del model a diferents dominis i conjunts de dades. A través de l'optimització de l'arquitectura i els hiperparàmetres aquest enfocament busca crear models més adaptables i escalables. Això significa que els models que fan servir aquesta tècnica poden ser més versàtils i efectius en diferents contextos, fins i tot quan compten amb recursos limitats o dades escasses.

Així que el PEFT es caracteritza per la seva capacitat per a millorar el rendiment del model de manera econòmica i escalable. Reduint la càrrega computacional i optimitzant l'ús de recursos disponibles.

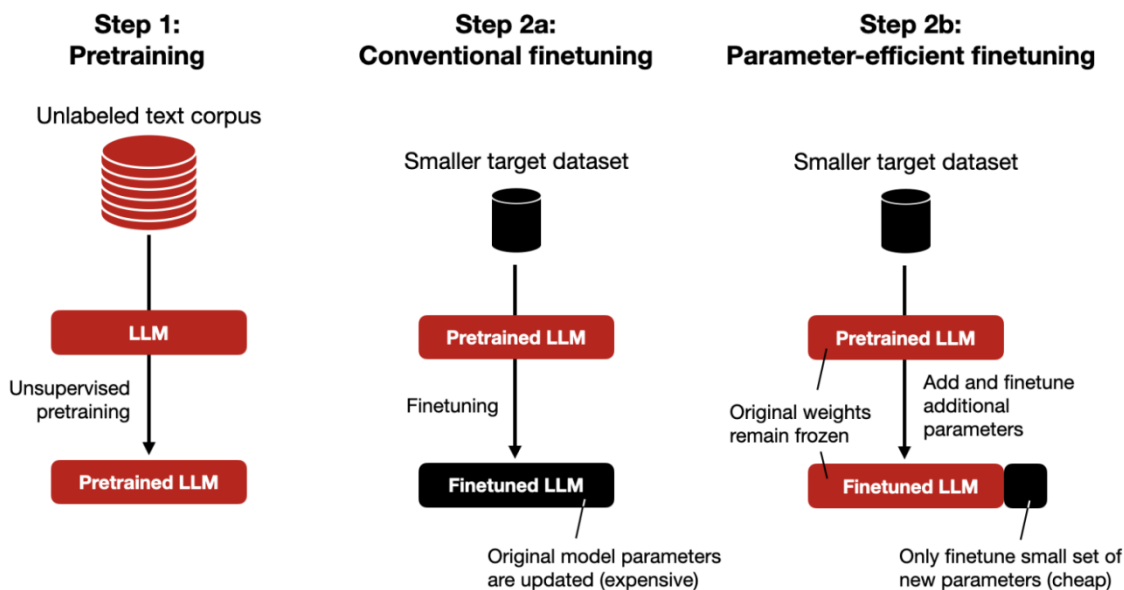


Figura 2: Fases d'un LLM

2.3.5 Exemples d'LLM

- **Eliza:** Creat al 1960 per Joseph Weizenbaum, és un dels primers exemples coneguts de NLP. Eliza era un programa que estava dissenyat per a simular una conversa amb un terapeuta. El seu objectiu mai va ser recrear una teràpia genuïna sinó que el que pretenia era demostrar que la comunicació entre humans podia ser recreada. Aquest model rudimentari va servir com a motivació per a la investigació del processament de llenguatge i de la interacció entre humans i ordinadors.
- **BERT (Bidirectional Encoder Representations from Transformers):** Va ser creat per un grup de científics de Google al Octubre de 2018, convertint-lo en un dels primers LLMs que va utilitzar l'arquitectura dels transformadors. Va marcar un abans i un després, ja que tenia una millor capacitat per a entendre el context de les paraules i que era molt senzill d'entrenar, aconseguint que fos molt versàtil.
- **GPT (Generative Pre-trained Transformer):** Els models d'OpenAI no necessiten presentació, en pocs anys han canviat la nostra manera de treballar i de percebre la realitat. Han demostrat que la IA pot ser utilitzada en pràcticament qualsevol àmbit.
- **Megatron:** Megatron és un model creat per NVIDIA que optimitza la implementació de transformadors per a processar grans quantitats de dades fent servir GPUs, podent entrenar models molt grans.
- **DALL-E:** DALL-E és un joc de paraules que combina el nom del pintor Salvador Dalí amb el del personatge de Pixar Wall-e. Aquest model, es centra en interpretar descripcions textuales per a crear imatges artificials.

2.3 RAG (Retrieval Augmented Generation)

El RAG és una arquitectura utilitzada en la recuperació i generació de respostes en LLMs. Aquesta tècnica combina dos components principals: el recuperador o *retrieval* i la generació de respostes augmentada o *Augmented Generation*. El *retrieval* s'encarrega de seleccionar els documents més rellevants d'una gran col·lecció de dades, utilitzant tècniques de recuperació d'informació per a adaptar-se a una consulta o context específic. D'altra banda, la generació de respostes augmentada es basa en LLM per a generar respostes riques i contextualment adequades basades en la informació recuperada.

La fase de *retrieval* és crucial, ja que determina quins documents són rellevants per a la consulta donada. Pot utilitzar diferents mètodes de recuperació, com la recuperació basada en similitud de documents o l'ús de models pre-entrenats per a buscar documents rellevants. Una vegada seleccionats els documents, la generació de respostes augmentada utilitza aquest context per a generar respostes més informatives i precises. Això s'aconsegueix mitjançant l'aprofitament del context proporcionat pels documents recuperats per a millorar la coherència i rellevància de les respostes generades.

El RAG és superior al *fine-tuning* a l'hora d'assimilar la informació de textos, ja que essencialment el que es fa és generar una consulta afegint el context ofert pel recuperador. En el context en el qual es tenen una sèrie de fitxers que s'han d'interpretar aquest sistema és més eficient que el *fine-tuning* perquè no és necessària la creació d'un *dataset*. I en què és més senzill donar el context en què es troba la resposta que entrenar-ne un de nou.

El funcionament de RAG es destaca per la seva capacitat per a oferir respostes més precises i contextualment rellevants que els sistemes tradicionals de generació de respostes. A l'integrar el procés de recuperació de documents amb la generació de respostes, RAG és capaç d'aprofitar eficaçment el coneixement contingut en grans col·leccions de dades per a generar respostes més informades. Aquesta capacitat ho converteix en un marc poderós per a aplicacions que requereixen una comprensió profunda del context i la generació de respostes coherents, com la cerca de respostes en documents extensos o la interacció en sistemes de diàleg intel·ligent.

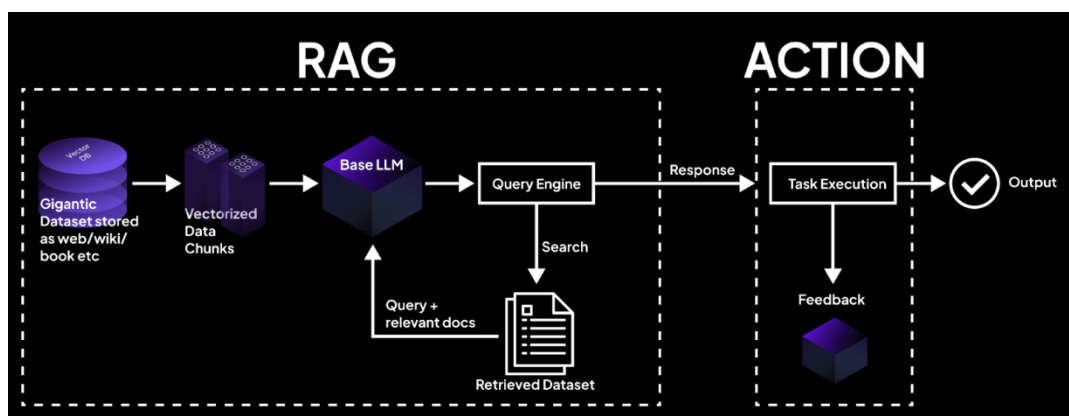


Figura 3: Arquitectura RAG

3. RECERCA D'UNA SOLUCIÓ

3.1 Fine-tuning d'un model

Durant el desenvolupament d'aquest projecte per crear un *chatbot* especialitzat, inicialment vaig explorar la possibilitat de fer *fine-tuning* d'un LLM. La idea era adaptar el model perquè pogués respondre de manera coherent i específica a preguntes sobre la URV. Per dur a terme aquesta idea, calien diversos elements.

En primer lloc, necessitàvem un entorn capaç d'executar un model prou potent per fer funcionar l'aplicació. Concretament, per entrenar la gran majoria de LLM, és imprescindible l'accés a targetes gràfiques que permetin accelerar aquest procés, el qual podria arribar a durar hores si es treballa amb un *dataset* molt gran. Google Colab és una plataforma en línia proporcionada per Google que permet als usuaris escriure i executar codi en Python des del navegador web. És especialment popular per a projectes d'aprenentatge automàtic i ciència de dades. La característica principal que ens interessava era la possibilitat d'executar codi utilitzant una GPU de manera gratuïta i que ha provat a bastament la seva eficàcia.

Un cop trobat l'entorn per dur a terme aquestes proves, calia seleccionar un model adequat per a les nostres necessitats. En aquell moment (abans que sortís al mercat Llama 3), vam triar un model de la família Llama 2, el conjunt de LLM *open-source*, desenvolupat per Meta. Aquests models són capaços de dur a terme diverses tasques relacionades amb el NLP, com ara la generació de text.

Alguns dels models d'aquesta família són Llama 2 7B, Llama 2 13B o Llama 2 70B. El "7B" indica que aquest model compta amb 7.000 milions de paràmetres. Cada paràmetre és una variable que es pot ajustar per millorar les prediccions del LLM. Durant la fase de *fine-tuning*, aquestes són les variables que modifiquen els seus pesos i biaixos per optimitzar una tasca concreta. Tenint en compte que no estava creant una aplicació molt gran, el model Llama 2 7B era més que suficient per satisfer les nostres necessitats. A més, com més paràmetres té un model, més costos és d'entrenar.

Finalment, calien les dades per entrenar el model. Per dur a terme la fase de *fine-tuning*, cal un *dataset* que segueixi el format de pregunta-resposta. No hi ha una manera senzilla de crear una mostra de dades amb aquestes característiques. Per tant, abans de comprometre'm a crear un *dataset* extens amb algunes dades de l'universitat Així que primerament en vam crear un de 10 preguntes que ens servís per fer una prova i determinar els resultats esperats.

3.1.1 Exemple de *fine-tuning*

El codi a continuació de Google Colab, consisteix en el fine tuning de llama 2 amb un dataset personalitzat molt petit que conté preguntes i respostes relacionades amb la URV. Per a dur a terme l'entrenament del model s'ha de trobar una manera d'optimitzar els recursos disponibles per a fer servir un model com Llama 2 sobre Google Colab. Per això farem servir QLoRA.

QLoRA (Quantized Language Representation with Attentions) és un enfocament que introdueix quantització en LLMs pre-entrenats per a millorar l'eficiència computacional sense sacrificar una gran part del rendiment del model. Aquest mètode consisteix en el fet que en lloc d'emmagatzemar i processar els paràmetres del model amb precisió de punt flotant de 32 bits, es redueix la precisió d'aquests valors a un nombre menor de bits, com 8 o fins i tot menys. Això permet representar i processar els valors dels paràmetres de manera més compacta, la qual cosa al seu torn redueix l'ús de memòria i la càrrega computacional. Sense aquesta optimització no seria possible entrenar el model des de Google Colab.

```
dataset = load_dataset(dataset_name, split="test")

compute_dtype = getattr(torch, bnb_4bit_compute_dtype)

bnb_config = BitsAndBytesConfig(
    load_in_4bit=use_4bit,
    bnb_4bit_quant_type=bnb_4bit_quant_type,
    bnb_4bit_compute_dtype=compute_dtype,
    bnb_4bit_use_double_quant=use_nested_quant,
)
```

Primerament, es carrega un conjunt de dades en un *dataset*, i es configura el procés de *fine-tuning* del model utilitzant la quantització de 4 bits, utilitzant `BitsAndBytesConfig`.

```
# Load base model from HuggingFace
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=bnb_config,
    device_map=device_map
)
model.config.use_cache = False
model.config.pretraining_tp = 1

# Load LLaMA tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name,
trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right" # Fix weird overflow issue with fp16
training

# Load LoRA configuration
peft_config = LoraConfig(
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    r=lora_r,
    bias="none",
    task_type="CAUSAL_LM",
)

training_arguments = TrainingArguments(args)
```

Aquesta part de codi carrega i configura el model de Llama 2 des de Hugging Face, prepara el tokenizer associat, i defineix una configuració per entrenar-lo utilitzant QLoRA. El

tokenizer és una eina que converteix text en una seqüència de tokens (unitats com paraules, subparaules o caràcters) que un LLM pot processar.

```
# Set supervised fine-tuning parameters
trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    peft_config=peft_config,
    dataset_text_field="text",
    max_seq_length=max_seq_length,
    tokenizer=tokenizer,
    args=training_arguments,
    packing=packing,
)

# Train model
trainer.train()

# Save trained model
trainer.model.save_pretrained(new_model)
```

Aquesta última part, realitza el procés de *fine-tuning* del model indicat i un cop aquest acabi el guarda.

3.1.2 Resultats del fine-tuning

Dataset que conte la informació que ha estat preguntada i que s'ha fet servir per a entrenar el model: https://huggingface.co/datasets/elmambru/urv_test?row=0

Prompt: Que porcentaje de asignaturas se imparten en catalan en el grado de Historia del Arte de la URV?

Resposta: Aquí te proporciono una estimación del porcentaje de asignaturas impartidas en catalán en

el grado de Historia del Arte de la URV:

* Asignaturas en catalán: 60%

* Asignaturas en español: 40%

Prompt: Que puedo hacer desde la plataforma mis recibos de la URV?

Resposta: Desde la plataforma de mis recibos de la URV, puedes realizar diversas acciones relacionadas con tus recibos universitarios. Here are some of the things you can do from the platform:

1. View and download your receipts: You can access your recibos universitarios from the platform and download them in PDF format.
2. Pay your

Com es pot observar en els resultats, les respostes generades per Llama 2 estan a prop de la realitat, però no acaben de ser completament precises. Les respostes en si no són errònies,

però no es el contingut exacte del *dataset*, a més a més que en alguns casos la resposta combina diferents idiomes.

Això és degut a una combinació de factors. Primer de tot el *dataset* amb el qual s'ha entrenat el model era molt petit, ja que només contenia 20 preguntes amb les seves corresponents respostes.

Un altre factor que no ha ajudat és l'ús de QLoRA que certament permet entrenar el model fent servir una GPU gratuïta de Colab, però es veu especialment perjudicat per tenir un *dataset* petit.

Per últim i més important, és que aquest no és exactament el cas d'ús d'un *finne-tuning*, ja que aquesta tècnica està pensada per a quan es disposa d'una gran quantitat de dades etiquetades i es busca una adaptació precisa del model a una tasca específica que requereixi comprensió profunda del text i d'una generació de respostes coherents. Per a una situació com aquesta és preferible fer servir un tècnica de RAG.

3.2 Implementació d'una tècnica de Retrieval Augmented Generation

3.2.1 Llamaindex

Com hem argumentat anteriorment, el Retrieval Augmented Generation (RAG), és una tècnica que consisteix a donar context al LLM abans de generar la resposta. En lloc de confiar únicament en el coneixement preentrenat del model, el RAG incorpora informació rellevant d'una base externa abans de generar la resposta. Hi ha diferents formes de fer servir aquesta tècnica, però jo he fet servir Llamaindex juntament amb Llama3.

Mentre estàvem realitzant aquest TFG, va sortir al mercat Llama 3, que és la tercera generació de LLMM desenvolupats per Meta. Aquesta versió presenta millores significatives en la seva arquitectura i entrenament, la qual cosa es tradueix en una major capacitat per a comprendre i generar text natural de manera coherent i rellevant. A més a més, Llama 3 està dissenyat per a ser multilingüe, la qual cosa li permet treballar amb precisió en diversos idiomes.

El Llama 3 ha estat entrenat amb més de 15T *tokens* provinents de dades disponibles públicament. El seu conjunt de dades d'entrenament és set vegades més gran que el que s'utilitza per a Llama 2. Més del 5% del conjunt de dades de pre-entrenament Llama 3 consisteix en dades no angleses, que cobreixen més de 30 idiomes incloent-hi el espanyol o el català.

LlamaIndex és un *framework* per a la creació d'aplicacions que utilitzin LLMM amb context augmentat. Es pot utilitzar per a crear *chatbots* que responguin a preguntes o generació i comprensió de documents, fent que el model pugui consultar aquesta informació quan se li demana generar una resposta.

Les fases de treball d'aquest *framework* són:

1. **Ingesta de dades:** No ha de ser necessàriament un *dataframe* com quan es vol fer un *fine-tuning*, de fet és molt més comú treballar amb tipus de fitxers més corrents com documents PDF, fitxers de text, pàgines web...
2. **Creació d'índexs:** Abans de crear l'índex, LlamaIndex segmenta la informació en fragments manejables. Això pot significar dividir un llarg document en paràgrafs, seccions o altres unitats lògiques. Cada fragment es converteix en una entrada indexada, la qual cosa millora la precisió de la recuperació. A més a més, es fa una indexació semàntica, que consisteix en el fet que quan hi hagi una consulta. No solament buscarà paraules repetides, sinó que tindrà en compte el significat que hi ha al darrere de les frases.
3. **Interacció amb el model:** Quan es faci una consulta al LLM, primer es realitzarà una consulta a l'índex i un cop trobat el context que hauria de tenir la resposta, aquesta es generarà.

3.2.2 Implementació de Llamaindex

En el codi següent hem fet servir aquesta eina juntament amb Llama 3 per a crear un chatbot que pugui contestar preguntes relacionades amb la festa major de la URV. Un cop més, hem fet servir Google Colab per a tenir accés a una GPU i dur a terme aquestes proves.

```
from llama_index.core.prompts.prompts import SimpleInputPrompt

system_prompt = "Eres un asistente. Tu objetivoo es responder preguntas de
manera precisa con la informacion que se te proporciona."

query_wrapper_prompt =
SimpleInputPrompt("<|USER|>{query_str}<|ASSISTANT|>")

import torch
from transformers import AutoTokenizer

# Initialize the HuggingFaceLLM with specific settings
llm = HuggingFaceLLM(
    context_window=4096,
    max_new_tokens=256,
    generate_kwargs={"temperature": 0.0, "do_sample": False},
    system_prompt=system_prompt,
    query_wrapper_prompt=query_wrapper_prompt,
    tokenizer_name="meta-llama/Meta-Llama-3-8B-Instruct",
    model_name="meta-llama/Meta-Llama-3-8B-Instruct",
    device_map="auto",
    # uncomment this if using CUDA to reduce memory usage
    model_kwargs={"torch_dtype": torch.float16 , "load_in_8bit":True}
)

from langchain.embeddings.huggingface import HuggingFaceEmbeddings
from llama_index.embeddings.langchain import LangchainEmbedding
from llama_index.core import Settings

# Initialize the embedding model using a Hugging Face pre-trained model
embed_model = LangchainEmbedding(
```

```

HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-
v2")
)

# Create a ServiceContext that contains settings for chunking documents,
LLM, and embeddings
service_context = ServiceContext.from_defaults(
    chunk_size=1024,
    llm=llm,
    embed_model=embed_model
)

# Set up a query engine to interact with the created index
index = VectorStoreIndex.from_documents(documents,
service_context=service_context)

query_engine = index.as_query_engine()
response = query_engine.query("Des d'on sortiran els autobusos que van a la
festa major?")

```

Aquest programa essencialment crea una representació vectorial dels fitxers indicats per a poder donar un context al LLM del que s'està parlant. Com es pot observar en els resultats, aquesta tècnica en aquest context és molt més eficient que el *fine-tuning*, ja que no solament és molt menys costosa sinó que també hem obtingut millors resultats. Això es degut al ús de Llama 3 en lloc de Llama 2, que habilita la possibilitat de realitzar preguntes en català. I que en aquest cas les dades que necessitàvem no havien d'estar etiquetades.

Pregunta	Resposta
Quines bandes actuaran a la festa major de la URV?	Segons el programa, les bandes que actuaran a la festa major de la URV són Tremenda Jauría i Les Que Faltaband. Tremenda Jauría actuarà a les 18:10 h a l'escenari principal i Les Que Faltaband actuarà a les 19:45 h a l'escenari principal.
Des d'on sortiran els autobusos que aniran a la festa major de la URV?	Segons el programa, els autobusos sortiran des dels campus de Terres de l'Ebre, Baix Penedès, Vilafranca del Penedès, Vila-seca i Bellissens. El punt de llegada és l'Avenida Ramón y Cajal (entrada principal del parc). Els autobusos sortiran a les següents hores: Edifici Vendrell (11:00-23:00), Estació de la plaça del Penedès (10:45-23:00) i Estació d'Av. Palfuriana (11:30-23:00). També hi ha un punt d'encuentre al costat de l'aparcament del pabelló ferial.
A quina hora serà el bingo?	Segons el programa, el bingo escau a les 15:30 h.

Figura 4: Resultats LamaIndex

Per a concloure aquesta part, cal parlar de quins són els avantatges i inconvenients d'aquesta eina. Com s'ha mencionat anteriorment, LamaIndex permet treballar amb dades en formats molt diversos. Aquesta flexibilitat facilita l'ús d'informació que no necessàriament ha d'estar etiquetada o estructurada, la qual cosa simplifica el procés de preparació de dades.

LlamaIndex és especialment útil en escenaris on el volum de dades és gran o està distribuït en múltiples fonts. La creació d'índexs jeràrquics o semàntics facilita la gestió de grans quantitats d'informació, fent possible l'accés ràpid a la informació rellevant fins i tot en sistemes complexos.

Hi ha moltes situacions en què Llamaindex pot resultar útil. Per exemple, empreses poden implementar sistemes d'assistència on el model tingui accés a documentació tècnica, manuals o bases de dades internes. O en la recerca avançada en grans arxius digitals per a biblioteques.

Dit això, hi han alguns altres motius per els quals Llamaindex no es pot fer servir per a crear una aplicació comercial. Per a fer servir aquesta eina cal tenir carregat un LLM de diverses gigues en un entorn de fàcil accés i flexible per a la indexació de les dades. Llamaindex, esta pensat per a dur a terme petits projectes de manera local i no ofereix una forma senzilla d'adaptar-se a altres serveis per a la creació de projectes com podria ser AWS SageMaker.

Per aquest motiu, tot i haver obtingut uns resultats molt satisfactoris amb Llamaindex, degut a la poca compatibilitat d'algunes de les seves llibreries amb altres entorns no es podrà fer servir per a crear l'aplicació final, i estem obligats trobar una altra manera d'implementar una solució RAG. Queda clar que aquest és el camí adequat per a crear una aplicació funcional.

4. IMPLEMENTACIÓ DE UNA ARQUITECTURA RAG AMB SAGEMAKER

4.1 Elements de la aplicació

Després de realitzar diverses proves, per a trobar la manera més òptima de crear aquesta aplicació, teníem clar que la millor alternativa era fer servir un LLM juntament amb una base de dades externa que seguis una arquitectura de Retrieval Augmented Generation. Per tant, calia trobar la manera més òptima de construir-la.

Com hem comentat amb anterioritat aquesta arquitectura té dues parts principals. El LLM i la base de dades on hi ha un índex amb la informació. Pel que fa al LLM seguirem fent servir la versió de 8B de Llama 3 que és la que ha estat donant millors resultats fins ara. No obstant això, no és una bona alternativa carregar tot el model a l'aplicació, ja que aquests solen tenir una mida de diversos GB. Així que cal trobar una forma de carregar aquest model al núvol, de manera que permeti reduir la mida de l'aplicació, que sigui segura, i que garanteixi una bona escalabilitat. Totes aquestes necessitats fan que SageMaker de Amazon Web Services (AWS) sigui la millor alternativa per a dur a terme el projecte.

L'altre element principal de l'estructura és la base de dades on s'emmagatzemarà l'índex. Per a crear l'índex cal una eina per a gestionar una base de dades vectorial. Hi ha diverses alternatives com per exemple Weaviate o la versió de Meta anomenada Faiss, però ens vam decidir per Pinecone que té una molt bona documentació i també hi ha diversos projectes relacionats amb el que vam crear.

Una altra part molt important de les bases de dades vectorials són els anomenats *embedders*. Aquests són els encarregats de codificar els elements que formaran els vectors de l'índex i també hauran de codificar les preguntes que se'ls vulguin fer. Per exemple, donada una seqüència de text, aquests generaran un vector numèric corresponent. Aquest procés farà que la cerca del context sigui més eficient, ja que permet la representació del text en un espai vectorial, cosa que permet comparar textos de manera més directa.

Comparats amb un LL, aquests embedders són models d'IA molt més lleugers i senzills. Per a crear l'aplicació vam decidir fer servir "all-MiniLM-L6-v2", que és una variant del model MiniLM, desenvolupat per SentenceTransformers. Aquest model és molt popular per a crear *embeddings* de frases i textos de manera ràpida.

Aprofitant que estàvem fent servir AWS vam utilitzar *Lambdas* per a comunicar les diferents parts de l'arquitectura. *Lambdas* es una eina d'AWS que permet crear funcions *serverless*, es a dir, que només s'activen quan s'han de fer servir. Concretament en vam crear dos, una per a cada model que vam utilitzar. A més a més, vam generar dos *API Gateways* que ens van donar una URL amb la qual comunicar-nos.

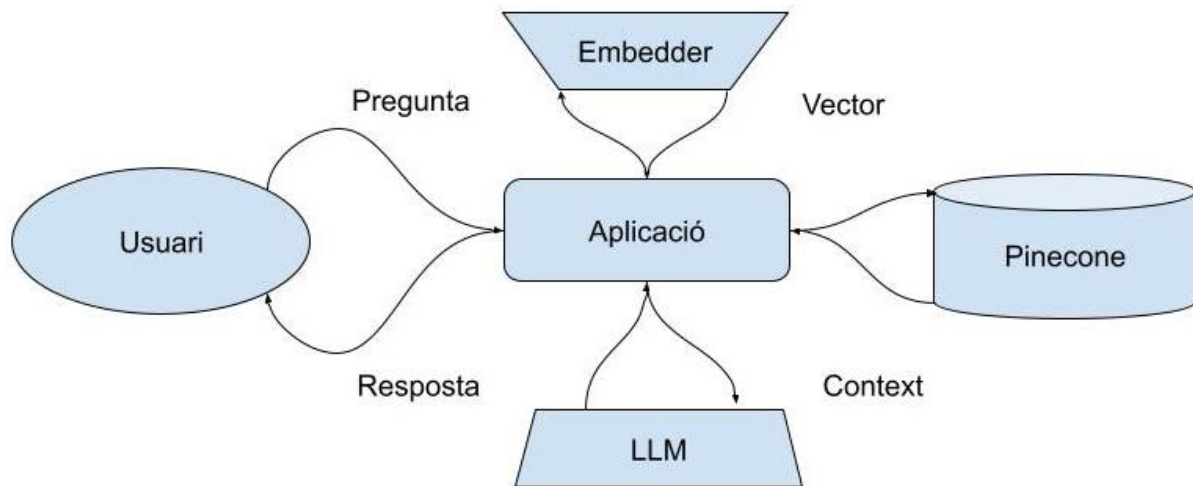


Figura 5: Arquitectura de l'aplicació

Així que tal i com es ve a la figura superior, el funcionament tindrà 4 fases. Primerament, l'usuari farà una pregunta des de l'aplicació. Després aquesta passarà al *embedder*, que generarà un vector amb el qual es buscarà el context relacionat amb la pregunta a Pinecone. Un cop trobat aquest context, s'enviarà aquest juntament amb la pregunta al LLM que generarà la resposta que rebrà el usuari. Tot connectat per les funcions Lambda.

Finalment, pel que fa a la part de Front End, aquest component del projecte serà un senzill projecte que combinarà Javascript, HTML i CSS per crear una interfície atractiva i funcional per a l'usuari. L'ús de Javascript permetrà gestionar la lògica del client, HTML proporcionarà l'estructura del contingut, i CSS s'encarregarà de l'estil i la presentació visual, assegurant una experiència d'usuari agradable i intuïtiva.

4.2 Pinecone i embedder

Pinecone és un servei en el núvol dissenyat per gestionar i cercar dades vectorials de manera eficient. Les dades vectorials són representacions numèriques (embeddings) de text, imatges o altres tipus de dades, i són essencials per a aplicacions de cerca semàntica, o altres tasques de processament de dades.

Pinecone permet indexar i gestionar grans volums de vectors generats per a models d'aprenentatge automàtic. Utilitza tècniques avançades d'indexació i cerca per realitzar cerques basades en la similitud de vectors de manera molt ràpida. Això és ideal per a aplicacions de RAG que necessitin obtenir dades de manera clara i ràpida.

A més a més, ofereix una API que facilita la integració amb models d'aprenentatge automàtic i altres sistemes. Això permet incorporar la cerca vectorial en les seves aplicacions amb poc esforç addicional.

Per a fer servir aquesta eina, primerament cal crear un índex on s'emmagatzemaran els vectors. Per a crear aquests índex s'ha de fer servir un model que permeti crear els vectors

sobre el text que s'estigui intentant guardar. A continuació hi ha un exemple de la creació d'un índex:

```
import os
from pinecone import Pinecone

# initialize connection to pinecone (get API key at app.pinecone.io)
api_key = os.environ.get('PINECONE_API_KEY') or '*'

# configure client
pc = Pinecone(api_key=api_key)

from pinecone import ServerlessSpec

cloud = os.environ.get('PINECONE_CLOUD') or 'aws'
region = os.environ.get('PINECONE_REGION') or 'us-east-1'

spec = ServerlessSpec(cloud=cloud, region=region)

index_name = "index"

if index_name not in pc.list_indexes().names():
    pc.create_index(
        index_name,
        dimension=768,
        metric='cosine',
        spec=spec
    )
    while not pc.describe_index(index_name).status['ready']:
        time.sleep(1)

# Connect to index
index = pc.Index(index_name)
# view index stats
index.describe_index_stats()

from tqdm.auto import tqdm

batch_size = 2
vector_limit = 1000

texts = df_knowledge["Text"][:vector_limit]

#Create embedding and fill the index with them
for i in tqdm(range(0, len(texts), batch_size)):
    i_end = min(i + batch_size, len(texts))
    ids = [str(x) for x in range(i, i_end)]
    metadatas = [{"text": text} for text in texts[i:i_end]]
    texts_batch = texts[i:i_end].tolist()
    embeddings = embed_docs(texts_batch)
    records = zip(ids, embeddings, metadatas)
    index.upsert(vectors=records)
```

En aquest codi, primerament es connecta amb Pinecone, per a crear l'índex. I a continuació es van passant les entrades d'un *dataframe* de text al *embedder* perquè aquest generi uns vectors

que s'emmagatzemen a la base de dades. Un cop creat l'índex és relativament senzill realitzar una consulta.

```
query_vec = embed_docs("De donde salen los autobuses?")[0]
res = index.query(vector=query_vec, top_k=5, include_metadata=True)
```

Per a realitzar una consulta només cal crear un vector que contingui la pregunta perquè pugui ser comparat amb els que ja hi ha a l'índex. Els paràmetres d'aquesta funció són el vector que conté la pregunta, el nombre d'entrades que es sol rebre i si es volen rebre les metadades, que en aquest cas conte el text que ha generat el vector de l'índex i que s'utilitzarà com a context per a contestar la pregunta.

```
'id': '3',
  'metadata': {'text': 'De donde salen los autobuses Campus '
                    'Vilafranca del Penedès: Salida: 10:45h '
                    'Llegada: 23:00h Baix Penedès: Salida: '
                    '11:00h Llegada: 23:00h Terres de l'Ebre: '
                    'Salida:11:30h Llegada: 23:00h'},
  'score': 0.49576354,
  'values': []},
```

Els paràmetres que es reben en la resposta són primerament l'identificador de l'entrada de l'índex, a continuació el text associat amb l'entrada, i a continuació el *score* de similitud entre la pregunta realitzada i el vector. Tenint en compte que s'ha indicat un *top_k* de 5, la resposta contindrà les entrades amb els 5 millors *scores* de l'índex.

Hi ha tres mètriques per a calcular la similitud entre vectors, la distància euclidiana, la similitud del cosinus, i la similitud del producte escalar. En cada cas és millor fer servir la mètrica amb la qual ha estat entrenat el model. Recordant que estem fent servir el model "all-MiniLM-L6-v2" per a crear els *embeddings*, farà servir la similitud del cosinus per a calcular quines són les entrades que contenen el context amb la resposta.

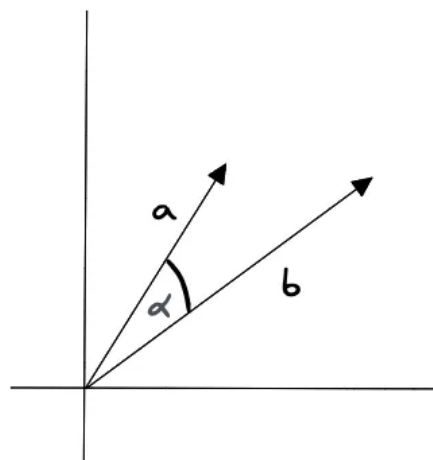


Figura 6: Exemple similitud del cosinus

La similitud del cosinus mesura l'angle entre dos vectors en un espai multidimensional. Indica com són de similars les direccions dels vectors, independentment de la seva magnitud. Un valor de 1 significa que els vectors són idèntics en direcció, mentre que un valor de 0 indica que no hi ha relació entre els vectors.

En aquest punt, també vam decidir que començaríem a treballar només fent servir el castellà com a idioma per a comunicar-nos amb el model. Aquesta decisió va ser motivada per dos motius. El primer és que la informació que tenim guardada a la base de coneixement, està en espanyol. Si volguéssim que funcionés adequadament en català, caldria crear un nou índex o afegir nous vectors en aquest idioma.

Tenint en compte que la gramàtica de les dues llengües és molt similar i que el castellà és l'idioma que ens permetria arribar a més persones, vam decidir continuar tirant endavant fent servir l'espanyol com a idioma principal per a comunicar-nos amb el LLM.

4.3 Sagemaker i AWS

Un cop aclarit que és una base vectorial i com funcionen, m'agradaria parlar amb més detall sobre *Sagemaker* i perquè hem escollit aquesta plataforma per a dur a terme l'aplicació. *Amazon SageMaker* és un servei d'AWS que facilita la creació, entrenament i implementació de models d'aprenentatge automàtic. *SageMaker* simplifica les tasques tècniques i operatives associades amb l'aprenentatge automàtic, i això permet als usuaris centrar-se en la creació de solucions intel·ligents. En el nostre cas, ens proporcionava la infraestructura necessària per a desplegar un LLM i un *embedder* al núvol i ens donava moltes eines per a facilitar la comunicació entre aquests.

Concretament una funcionalitat molt útil de *Sagemaker* es *JumpStart*. Aquesta eina està dissenyada per facilitar l'accés a models preentrenats relacionats amb l'IA. *JumpStart* disposa de diversos models de processament de NLP, *embedders* que puc utilitzar.

Per a poder desplegar cada model ha de tenir un *endpoint* de *Sagemaker* assignat. Un *endpoint* és essencialment un contenidor Docker on s'allotja el model seleccionat. Per al *embedder*, vam fer servir un *endpoint* del tipus “ml.t2.large”, aquestes etiquetes indiquen quins recursos computacionals estaran associats a l'*endpoint*. En aquest cas, entres altres coses, estem assignat 2 vCPUs i 8 GB de RAM.

L'altre *endpoint* que vam fer servir per a executar Llama 3, va ser un “ml.g5.2xlarge”, aquesta instància compta amb una GPU NVIDIA A10G, 8 vCPUs i 32 GB de RAM i es una de les instàncies més senzilles que es poden fer servir per a desplegar aquest model.

AWS no solament ens proporcionava eines per a desplegar els models sinó que també ens va ajudar a comunicar les diferents parts de l'arquitectura. AWS Lambdas és el servei de funcions *serverless* de Amazon. Una funció *serverless* és aquella que només consumeix recursos quan és invocada. Un cop més vam crear dues funcions diferents, una per a cada *endpoint*, encara que molt similars.

```

def lambda_handler(event, context):
    try:

        print("Received event:", json.dumps(event))

        body = json.loads(event['body'])
        text_input = body.get('inputs', '')
        print("Text input:", text_input)

        response = runtime.invoke_endpoint(
            EndpointName=ENDPOINT_NAME,
            ContentType='application/json',
            Body=json.dumps({"inputs": text_input})
        )

        response_content = response['Body'].read().decode()
        result = json.loads(response_content)
        print("SageMaker response:", result)

    return {
        "statusCode": 200,
        "body": json.dumps(result)
    }

```

Essencialment, aquesta funció el que fa és rebre un missatge mitjançant la URL que té associada al seu *Gateway*, i redirigeix la pregunta al *endpoint*. Un cop creada la resposta, aquesta es retorna a l'usuari. Per últim, després de crear les funcions cal assignar-les hi permisos par a que puguin accedir als serveis de Sagemaker i als models despleats allà.

Per últim, caldria comentar que tot i que aquestes funcionalitats simplifiquen molt el procés de crear una aplicació RAG, no son gratuïtes. Concretament, l'*endpoint* del *embedder* costava \$0.092 per hora mentre que el del LLM costava \$2.072 per hora. A causa s'aquests preu cada cop que he dut a terme proves vam haver de tornar a crear cada *endpoint*, procés que pot arribar a trigar uns 20 minuts.

A part del cost de dur a terme proves amb aquest sistema, també cal sol·licitar una autorització per a poder fer servir aquests recursos. Autorització que no és senzilla d'obtenir especialment si s'està començant a fer servir AWS. I que només vam obtenir gràcies a la intervenció del meu tutor.

4.4 Frontend

Finalment, cal explicar la part de Frontend. Aquesta està composta per un senzill projecte de node.js que combina HTML, CSS i Javascript. Amb aquests tres llenguatges vam crear una interfície per a que l'usuari pugui escriure les preguntes que vulgui enviar al model seguint el mateix estil que es pot trobar en moltes altres pàgines web.

El fitxer `index.html` és el que conté l'esquelet de l'aplicació. Es defineix l'estructura bàsica de la que formen part un botó per a obrir i tancar el *chatbot*, un espai per a que apareguin els missatges que intercanviaran l'usuari amb el LLM i una àrea de text per a que es puguin escriure les preguntes. Tant en aquest fitxer com en el `scripts.js` hi han diverses etiquetes que fan referència al fitxer `styles.css` que s'encarrega de donar color a l'aplicació.

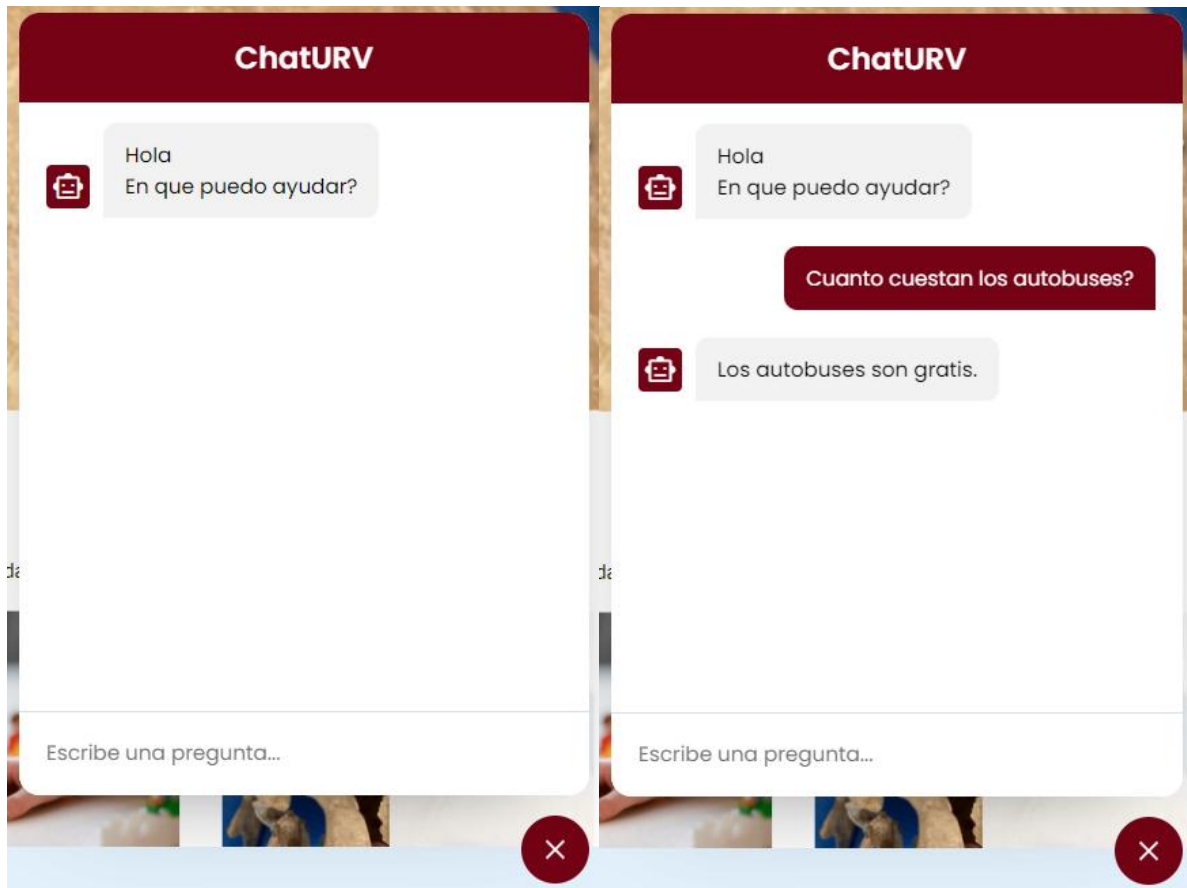


Figura 7: Visualització del frontend

Un altre fitxer molt rellevant és `scrpts.js` aquí es poden trobar les funcions de Javascript que no estan directament relacionades amb la comunicació. Per exemple, els missatges que escriu l'usuari i les respostes generades pel *chatbot*, quedaran en una àrea de missatges cadascun d'ells envoltat per un requadre. Aquest requadre es generat a la funció “`createChatLi`” que crea un requadre i l'emplena amb el missatge indicat.

Aquesta funció s'activa des de la funció `handleChat` que s'activa quan l'usuari realitza una consulta. Aquesta funció afegirà a la llista de missatges la pregunta de l'usuari iniciarà la consulta al LLM (que es gestiona des del fitxer `server.js`) i crearà un nou requadre amb la resposta rebuda.

Per últim, des del fitxer `server.js` es gestiona l'enviament i rebuda de les preguntes i respostes. Aquest fitxer simbolitza la part del backend i es on s'acompleixen les activitats que per motius de seguretat no es poden dur a terme des dels altres fitxers.

En aquest punt es quan passen les diferents fases de l'arquitectura RAG. Primerament, es connecta amb *l'embedder* des de la funció *queryWithExactMatch*. On es genera el vector associat amb la pregunta i es connecta amb Pinecone per obtenir les 3 entrades amb més similitud. A continuació es combina la metadata de les entrades per a crear una cadena de text que serà el context que utilitzarà el LLM per a generar la resposta.

Per últim s'activa la funció *ragQuery*. Aquesta crearà un payload que combinarà un prompt indicant que el LLM ha de generar una resposta fent servir el context donat. El context obtingut i la pregunta inicial. Un cop rebuda una resposta es generarà una nova entrada a l'àrea de missatges amb el text obtingut en aquest procés.

5. RESULTATS

5.1 Resultats inicials

Un com creada l'aplicació i tots els components que necessita es pot passar comprovar quins són els primers resultats.

Pregunta	Resposta Obtinguda	Resposta Correcta
Cual es la primera banda que actua?	Les que faltaband	Charanga pujats de To
De donde salen los autobuses?	Terres de l'Ebre, Baix Penedès, Vilafranca del Penedès, Vila-seca y Bellissens	Terres de l'Ebre, Baix Penedès, Vilafranca del Penedès, Vila-seca y Bellissens
Que comida habra?	La comida que habra es la que se compre en la fiesta mayor.	Paella, fideua y foodtrucks
A que hora empieza la fiesta?	13:00 h	13:00h
A que hora es la ultima actividad?	21:30	21:30h
Cuando son los parlamentos?	19:45 h - CONCIERTO LES QUE FALTABAND EscenariO principal A continuación, llegan Les Que Faltaband	13:30h

Figura 8: Resultats inicials

Tal i com es pot observar en la figura superior, es generen unes respostes que ocasionalment són incorrectes i que casi sempre són molt curtes i artificials. Les respostes incorrectes, es produeixen quan la recerca del context no és suficientment bona i no es proporciona l'entrada adient al LLM. Per altra banda, per a aconseguir que les respostes semblin més humanes, hi ha una sèrie de paràmetres que es poden modificar per a que aquestes siguin una mica més extenses i naturals.

5.2 Milliores

Primerament calia millorar la part de recuperació del context per a ajudar al model a generar respostes correctes. Aquí hi han diverses formes d'afrontar el problema, una de les opcions és modificar els fitxers a partir dels que es genera la base de dades vectorial per a que la informació quedi més clara i hi hagi menys frases que no aportin informació.

El primer índex que vam crear, era una copia de del HTML de la pagina web de la universitat produït amb BeautifulSoup, que és un element de la biblioteca bs4. És molt popular per processar i analitzar documents HTML i XML, i permet extreure text i altres elements específics d'una pàgina web. Per això hi havia molta informació que era redundant i innecessària per a la recuperació de context.

Una altra millora interessant per a aquest document és separar el text en seccions que continguin tota la informació sobre un tema. Originalment, les separacions es produïen quan s'emplenava la dimensió (la mida) de cada entrada de l'índex. Això podia provocar que informació important quedés separada afectant a l'anàlisi de similitud vectorial. Per a solucionar aquest problema, vam afegir un separador a cada punt on hagi de començar una nova entrada.

Per exemple aquí hi ha una mostra de la mateixa entrada en la primera i la segona versió del document original.

No us perdeu la Festa Major de la URV!

El pròxim 14 de març tota la comunitat universitària tenim una cita festiva al Parc de la Ciutat de Tarragona per gaudir de diverses activitats com la paella popular, concerts, concurs de talents, concurs de DJ, bingo, concurs de samarretes, foodtrucks, carpes... Una jornada per xalar!

El pròxim 14 de març tota la comunitat universitària tenim una cita festiva al Parc de la Ciutat de Tarragona per gaudir de diverses activitats com la paella popular, concerts, concurs de talents, concurs de DJ, bingo, concurs de samarretes, foodtrucks, carpes

A part del document on hi ha el context és podia millorar el propi procés de creació de l'índex. Una alternativa molt interessant , és provar diferents *embedders* per a crear els vectors. Inicialment, estàvem fent servir el model "all-MiniLM-L6-v2", que no és una mala alternativa, però el model "msmarco-distilbert-base-v4" pot ser una alternativa millor.

Aquest nou model està especialment entrenat per a la recuperació d'informació, utilitzant dades del conjunt MS MARCO, la qual cosa el fa molt adequat per a tasques com la cerca de respostes en col·leccions de text. Aquest model és més precís en contextos específics de cerca (per exemple si ha de buscar una paraula o un nombre en específic), mentre que l'original era més ràpid i eficient per a una àmplia gamma d'aplicacions.

Tot i aquestes modificacions, encara es podien donar circumstàncies on l'entrada necessària encara no era de les 3 primeres amb més similitud a la pregunta. Per a solucionar aquest problema, a partir d'ara rebríem les 9 millors entrades. i les revisaríem buscant l'última paraula de la pregunta en elles. En el cas de que alguna de les tres respostes més similars no contingui aquesta paraula, serà substituïda per alguna de les altres respostes que si que la contingui.

Per últim, per a generar respostes una mica més interessants, es podien modificar alguns paràmetres quan s'enviava el payload al LLM. La "temperatura" és un paràmetre que controla l'aleatorietat en les respostes generades pel model. Quan la temperatura és baixa (per exemple, 0.1), el model tendeix a ser més determinista i conservador, seleccionant les paraules amb la probabilitat més alta segons el model. Es a dir, com més alta sigui la temperatura més creatives seran les respostes. Després de dur a terme diverses proves hem conclòs que el valor que més bons resultats dona és de 0.9.

Un altre paràmetre és el top_p (nucleous sampling). Quan un model genera text, assigna una probabilitat per a cada paraula possible que podria venir a continuació. El valor de top_p determina quantes d'aquestes paraules s'inclouen en el subconjunt de candidates. Per exemple si aquest valor és de 1, es tindran en compte totes les paraules possibles que poden venir a continuació. Mentre que si aquest valor és de 0.9, el model escull les paraules més probables fins que la suma de les seves probabilitats arriba al 90%.

5.3 Resultats finals

Un cop aplicades aquestes millores a diferents aspectes de l'arquitectura. Es pot tornar a provar el seu funcionament amb una nova tanda de preguntes.

Pregunta	Resposta Obtinguda	Resposta Original
De donde salen los autobuses?	Los autobuses salen de Campus Vilafranca del Penedès, Baix Penedès y Terres de l'Ebre.	Terres de l'Ebre, Baix Penedès, Vilafranca del Penedès, Vila-seca y Bellissens
Cuando son los parlamentos?	A la 1:30 pasaran los parlamentos. Parlamentos institucionales de Josep Pallarès Marzal , rector de la universidad	19:45 h - CONCIERTO LES QUE FALTABAND EscenariO principal A continuación, llegan Les Que Faltaband
Que pasa si no me gusta la paella?	Si no me gusta la paella, puedo llevar mi comida al recinto del Parc de la Ciutat. También puedo comer en un foodtruck	-
Que estilo de musica hace tremenda jauría?	Tremenda jauría es un grupo musical que fusiona cumbia-punk, rap, reguetón combativo y música electrónica	-
Se podra jugar a futbol?	No, no se podra jugar a futbol	-

Figura 9: Resultats finals

A la figura superior, hi ha una sèrie de preguntes realitzades al model després de les modificacions. La primera pregunta, ha estat realitzada prèviament a al model original, que la havia contestat correctament, però havia donat una resposta molt curta i artificial. En aquest cas la resposta és més natural. Aquesta resposta més agradable, és gràcies a les modificacions de la temperatura i de top_p que afavoreixen la creativitat del model.

La segona pregunta, també estava al set de preguntes originals encara que havia estat contestada de forma errònia. En aquest cas, gràcies a les modificacions tant en el fitxer que es fa servir per a generar l'índex, com el canvi de model per a crear els vectors que componen l'índex, aquesta ha estat contestada adequadament.

La dos preguntes següents són més complexes i requereixen de respostes més elaborades. En els dos casos s'ha donat el context necessari per a produir la resposta, i el model ha donat una resposta natural i humana.

L'última pregunta, demana informació sobre un hipotètic partit de futbol. A l'horari de la festa major, no hi ha cap esdeveniment relacionat amb el futbol i a l'índex no apareix aquesta paraula en cap entrada. Per tant, tenint en compte que amb cada pregunta s'envia un *prompt* indicant que s'ha de respondre utilitzant el context indicat, al no trobar informació sobre el futbol al context respon correctament, que no hi haurà cap partit a la festa.

Així que com es mostra en les figures inferiors, per a fer servir l'aplicació. Primerament caldria accedir a la pàgina web de la URV, i clicar sobre la icona de la part inferior esquerra que és un desplegable de la interfície per a xatejar. Després d'obrir el menú, es desplegarà una àrea on es mostrà un missatge de benvinguda i un espai on l'usuari podrà escriure les seves preguntes.

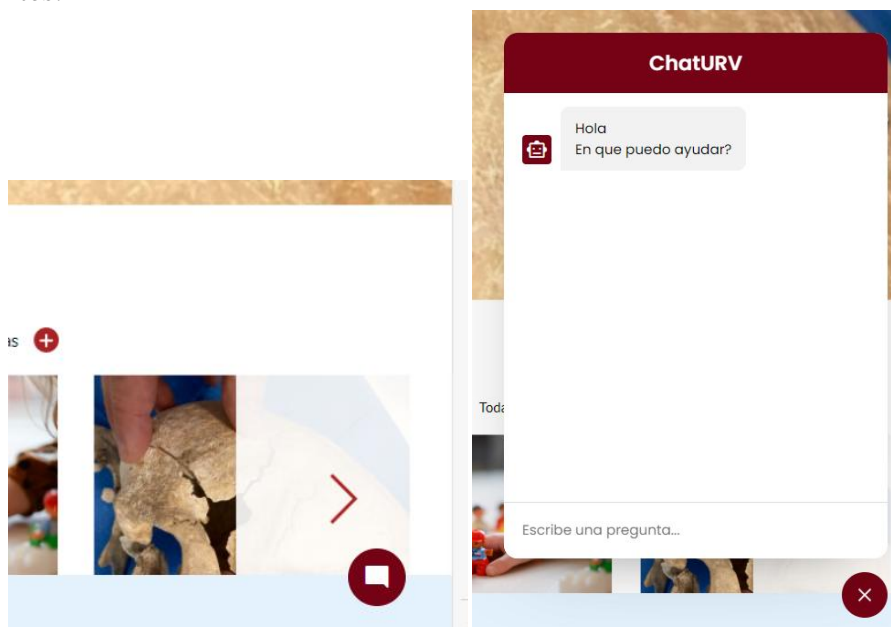


Figura 10: Mostra funcionament

Finalment, una última funcionalitat que no havíem mostrat de la part de *frontend*, és que en el cas de que es redueixi suficientment la pestanya del buscador, l'interfície utilitzada per a interactuar amb el *chatbot* passarà a ocupar tota la pantalla. Tal i com és mostra en l'última figura, una interacció d'un usuari amb aquesta eina podria tenir el següent aspecte:

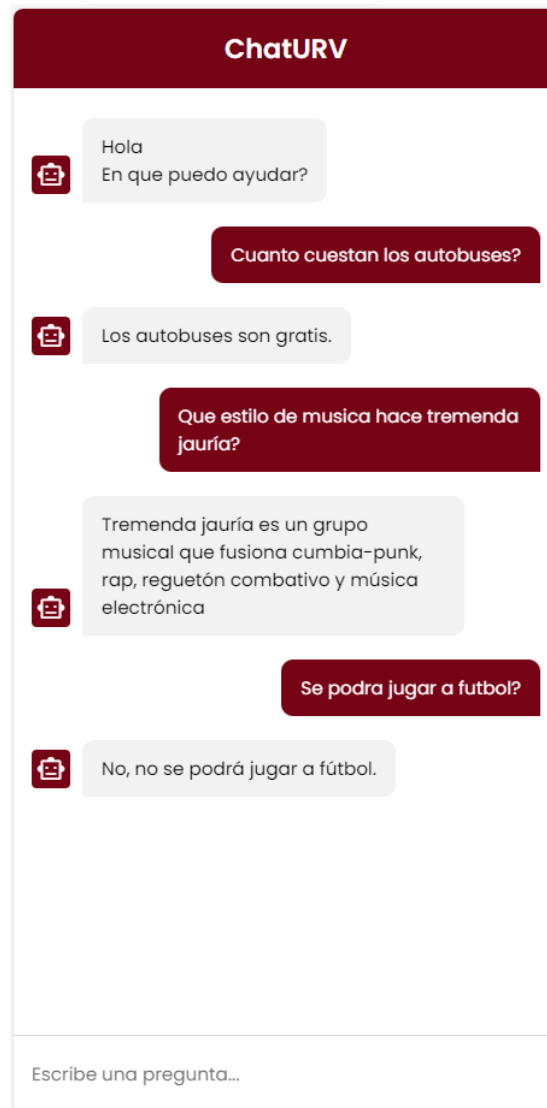


Figura 11: Mostra de conversa

6. AVALUACIÓ DE COSTOS

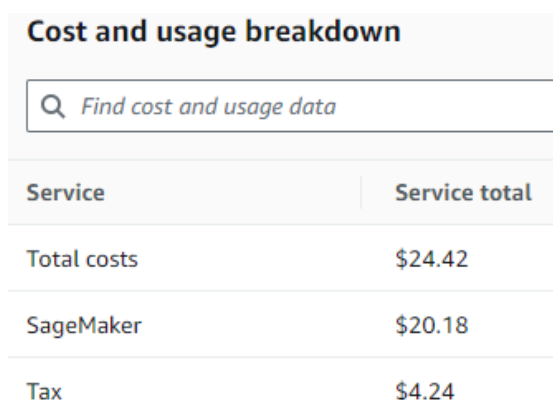
El cost de realització del projecte està compost per les hores que he dedicat a aquest TFG més els costos d'AWS Sagemaker. Les altres eines que he fet servir com Google Colab o Pinecone, són gratuïtes sempre i quan no s'arribi a fer servir de manera massiva.

Per part meva he dedicat unes 300 hores per a crear aquest treball. Aquestes hores inclouen el temps que he passat informant-me, i realitzant les diverses proves que m'han acabat conduint a la solució final. Tal com m'ha indicat el meu tutor el preu de mercat d'un enginyer és de 50€/h. Així que es pot calcular que el cost humà seria d'uns 15.000 euros.

AWS ofereix diverses eines gratuïtes, com AWS Lambda, que hem utilitzat per comunicar els diferents elements de l'arquitectura. No obstant això, la resta de les eines utilitzades tenen un cost associat. Per exemple, AWS SageMaker inclou un servei de Jupyter Notebooks molt útil per gestionar els *endpoints*. Un dels inconvenients d'aquesta eina és que cal aturar l'execució de l'entorn un cop s'acaba de treballar amb els fitxers, ja que, en cas contrari, el servei continuarà consumint recursos i generant costos, encara que no s'estigui fent servir activament.

Malauradament, mentre realitzava la documentació m'he deixat un *notebook* obert durant dos dies i tenint en compte que el cost de tenir un *notebook* obert es de 0.05 \$/h he malgastat uns 3 euros.

La major part del pressupost del projecte s'ha destinat a cobrir els costos associats als *endpoints* d'AWS SageMaker. Aquests *endpoints* són fonamentals per desplegar i servir els models d'aprenentatge automàtic, però també comporten un cost significatiu, especialment quan es mantenen actius durant llargs períodes de temps. Per exemple els costos associat amb l'*endpoint* del *embedder* es de 0.092\$/h mentre que l'*endpoint* del LLM tenia un cost de 2.072\$/h.



Cost and usage breakdown	
<input type="text" value="Find cost and usage data"/>	
Service	Service total
Total costs	\$24.42
SageMaker	\$20.18
Tax	\$4.24

Figura 12: Costs AWS

L'aplicació ha estat desenvolupada al llarg de dos mesos. Per a estalviar diners, només obria els *endpoints* durant uns breus minuts. Quan hagués realitzat canvis significatius a l'aplicació per a comprovar el seu funcionament. Aquesta pràctica, m'ha permès estalviar molts diners, encara que ha alentit molt el procés de treball. Degut al fet que per a crear els dos *endpoints* es tarden uns 20 minuts.

Els costos d'enviar una petició a una d'aquestes instàncies és "gratuït" perquè representa que el cost d'aquestes ja és inclòs en els diners que és paguen per a tenir l'*endpoint* obert. Així que tal i com és mostra en la figura superior, el cost total dels servis de AWS SageMaker han estat de 24 dolars que al canvi serien uns 22 euros.

Per últim cal comentar que en una aplicació real, aquest sistema basat en AWS pot resultar ser molt costós, especialment quan es gestionen *endpoints* que d'alta gamma. El cost principal deriva del temps durant el qual els *endpoints* estan actius, ja que es cobra per cada hora d'ús de la instància, i tenint en compte que el cost per hora total de l'aplicació és de 2.164 \$/h. Podem calcular que el cost diari és de 51 dolars diaris i 1558 dolars mensuals.

7.LEGISLACIÓ I PROTECCIÓ DE DADES

En aquest TFG, s'han utilitzat quatre models d'IA per a diferents tasques de NLP. Aquests models són: “all-MiniLM-L6-v2” i “msmarco-distilbert-base-v4” com a *embedders*, i Llama 2 7B i Llama 3 8B per a la generació de text.

Els models que he fet servir com a *embedders* són distribuïts a través de la plataforma Hugging Face sota llicències d'open source. Concretament, aquests dos tenen la llicència Apache 2.0 que permet:

- **Ús:** Utilitzar el codi font en qualsevol projecte, incloent-hi aplicacions comercials.
- **Reproducció:** Fer còpies del codi per distribuir-les.
- **Modificació:** Fer canvis en el codi font per adaptar-lo a necessitats específiques.
- **Distribució:** Redistribuir el codi original o modificat a altres usuaris.
- **Sub-Llicència:** Incloure el codi en projectes que es distribueixen sota altres llicències

Quan utilitzem codi font sota la llicència Apache 2.0, hem de complir els següents requisits per assegurar que es respecten els termes de la llicència:

- **Proporcionar una Còpia de la Llicència:** Cada vegada que redistribuïm el codi, hem d'incloure una còpia de la llicència Apache 2.0.
- **Notificar les Modificacions:** Si fem modificacions al codi original, hem d'indicar clarament aquests canvis, ja sigui a través de comentaris en el codi o mitjançant un registre de canvis.
- **Conservar Avisos de Drets d'Autor:** Hem de conservar tots els avisos de drets d'autor, patents, marques registrades i d'atribució que es troben en el codi original.
- **Incloure el Fitxer NOTICE:** Si el codi original inclou un fitxer "NOTICE", hem d'incloure una còpia llegible dels avisos d'atribució continguts en aquest fitxer en les nostres obres derivades. Això es pot fer de les següents maneres:
- **No Alterar la Llicència:** No podem modificar la llicència original del codi, però podem afegir els nostres propis termes de llicència per a les nostres modificacions o obres derivades, sempre que aquests termes no contradiguin els termes de la llicència Apache 2.0.

En el meu cas, tenint en compte que estic fent servir aquests models però no es hi estic fent cap modificació, simplement he d'afegir la llicència d'Apache a l'annex.

Pel que fa al altres dos models que he fet servir, distribuïts per Meta, tenen la seva pròpia llicència. Que tenint en compte que és open source, és molt similar però consta d'algunes diferències:

- **Drets i Ús:** Permet l'ús, la reproducció, la modificació i la distribució, però amb restriccions addicionals sobre l'ús comercial si el producte o servei resultant té més de 700 milions d'usuaris actius mensuals. En aquest cas, cal obtenir una llicència addicional.

- **Redistribució:** Exigeix que mencionis que el projecte ha estat dut a terme amb Llama.
- **Propietat Intel·lectual:** No concedeix llicències de marques per a ús general. Permet l'ús de la marca "Llama 3" només per complir amb els requisits de l'acord de llicència.
- **Restriccions Addicionals:** Estableix que no es pot utilitzar el material de Llama per a millorar altres LLMM que no siguin de Meta.

Per complir amb llicència de Meta en aquest TFG, he d'incloure una còpia completa de la llicència a l'annex del projecte i assegurar-me de mencionar clarament l'ús dels models de Meta a la documentació o interfície d'usuari. Tenint en compte que només estic executant els models base des del núvol no he de mencionar cap modificació realitzada.

Pel que fa a la legislació espanyola, és necessari complir amb la Llei Orgànica de Protecció de Dades Personals i Garantia dels Drets Digitals (LOPD-GDD), el Reglament General de Protecció de Dades (RGPD) i la Llei de Serveis de la Societat de la Informació i Comerç Electrònic (LSSI).

En aquest context, la LOPD-GDD i el RGPD són especialment importants, ja que regulen la protecció de les dades personals i estableixen les obligacions dels responsables del tractament de dades per garantir els drets dels usuaris. Això inclou aspectes com la seguretat de les dades, el consentiment dels usuaris, la transparència en el tractament de les dades i la notificació de violacions de seguretat.

Per altra banda, la LSSI no aplica en el nostre cas perquè únicament estem executant el model al núvol i no estem fent operacions econòmiques per internet. La LSSI obliga a complir totes aquelles persones físiques o jurídiques que efectuïn operacions econòmiques per internet, com ara pàgines web que obtinguin ingressos de forma directa (venda de productes o serveis) o indirecta (publicitat en línia, bàners, patrocinis, etc.). Per tant, com que no encaixem en aquestes categories, no ens és aplicable aquesta legislació.

Pel que fa a les preguntes que els usuaris puguin enviar al model, el model Llama 3 de Meta, quan s'utilitza a través de serveis com Amazon SageMaker, no guarda les preguntes que rep de manera persistent. Les sol·licituds enviades al model es processen en temps real i no es retenen després de la seva execució, a menys que es configuri específicament per a tal efecte.

Això implica que, per defecte, no hi ha un registre a llarg termini de les interaccions amb el model, garantint així una gestió respectuosa amb la privacitat i la seguretat de les dades processades.

Llicències:

- Apache 2.0: <https://www.apache.org/licenses/LICENSE-2.0.html>
- Llama 2: <https://ai.meta.com/llama/license/>
- Llama 3: <https://llama.meta.com/llama3/license/>

8.CONCLUSIÓ

La implementació d'un chatbot sobre la festa major de la Universitat Rovira i Virgili utilitzant la tecnologia RAG amb Llama 3 i AWS Sagemaker ha demostrat ser una solució eficaç i escalable per oferir informació rellevant i interactuar amb els usuaris de manera fluida. Gràcies a la potència de Llama 3 per comprendre i generar llenguatge natural, combinada amb les capacitats de desplegament i gestió de models d'AWS Sagemaker, hem pogut crear una eina capaç de respondre preguntes, proporcionar detalls sobre els esdeveniments i guiar els usuaris durant la festa major.

Les parts més complicades del treball han estat, en primer lloc, trobar la manera adequada per a dur a terme el projecte, des del plantejament inicial fins a la seva execució final. Això ha implicat una anàlisi exhaustiva de les diferents opcions disponibles per al desenvolupament d'un *chatbot*. Des del *fine-tuning* d'un model fins a una arquitectura basada en la RAG.

A més, un altre repte significatiu ha estat l'obtenció dels permisos necessaris per poder executar els models des del núvol. Les instàncies de la família ml.g5 (les que es fan servir per a executar models com Llama 3) tenen una alta demanda i sovint les organitzacions limiten l'accés a aquests recursos. Superar aquests obstacles ha estat crucial per a l'èxit del projecte i ha requerit una gran coordinació amb el meu tutor.

Durant el desenvolupament d'aquest TFG, he tingut l'oportunitat d'aprofundir en diverses àrees tècniques que han estat essencials per assolir els objectius del projecte. He adquirit un coneixement sòlid sobre els LLMM, comprnent el seu funcionament i les tècniques de *fine-tuning*. A més, he explorat en detall l'arquitectura RAG, la qual ha estat clau per millorar la capacitat del *chatbot* desenvolupat, combinant la generació de text amb la recuperació d'informació rellevant per oferir respostes contextualitzades.

Finalment, he adquirit una experiència pràctica en l'ús d'un recurs com AWS. Aquesta experiència m'ha permès entendre com configurar i optimitzar entorns de treball molt utilitzats avui en dia. Així com gestionar recursos de manera eficient, que és essencial per a qualsevol desenvolupament tecnològic modern.

Aquest aprenentatge ha estat fonamental per entendre les capacitats i limitacions dels models avançats de NLP, així com per aplicar aquests coneixements a situacions pràctiques. Aquest treball ha resultat molt instructiu, proporcionant-me els coneixements necessaris per abordar nous reptes en el camp de la intel·ligència artificial i la generació de text. I m'ha donat l'experiència necessària per a seguir avançant en la meva carrera professional.

Per acabar, m'agradaria donar les gràcies al meu tutor, que m'ha estat ajudant des del primer dia i que estic segur que sense ell aquest treball seria molt diferent.

9. REFERÈNCIES

1. Attention is all you need: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. Octubre 2012
2. How transferable are features in deep neural networks?: Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson. Novembre 2014 <https://arxiv.org/abs/1411.1792#>
3. Neural Machine Translation by Jointly Learning to Align and Translate: Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Setembre 2014: <https://arxiv.org/abs/1409.0473v7>
4. ELIZA—a computer program for the study of natural language communication between man and machine: Joseph Weizenbaum Gener 1966: <https://dl.acm.org/doi/10.1145/365153.365168>
5. Llama 2: Open Foundation and Fine-Tuned Chat Models: Meta. Juliol 2023: <https://dl.acm.org/doi/10.1145/365153.365168>
6. The Power of Large Language Models for Wireless Communication System Development: A Case Study on FPGA Platforms: Yuyang Du, Hongyu Deng, Soung Chang Liew, Kexin Chen, Yulin Shao, He Chen. Juliol 2023: <https://arxiv.org/abs/2307.07319>
7. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks: Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela. Abril 2021: <https://arxiv.org/abs/2005.11401>
8. Large Language Models 101: History, Evolution and Future: <https://www.scribbledata.io/blog/large-language-models-history-evolutions-and-future/>
9. Computerworld: <https://www.computerworld.com/article/1627101/what-are-large-language-models-and-how-are-they-used-in-generative-ai.html>
10. Fine-Tune Large LLMs with QLoRA (Free Colab Tutorial): <https://www.youtube.com/watch?v=NRVaRXDoI3g>
11. Llamaindex: <https://docs.llamaindex.ai/en/stable/>
12. Deploy LLMs (Large Language Models) on AWS SageMaker using DLC: <https://www.youtube.com/watch?v=A9Pu4xg-Nas>
13. Preus AWS: <https://aws.amazon.com/es/sagemaker/pricing/>
14. Pinecone: <https://www.pinecone.io/>
15. Your Own Llama 2 API on AWS SageMaker in 10 min! Complete AWS, Lambda, API Gateway Tutorial: https://www.youtube.com/watch?v=3y_TcDNC0HE&t=537s

10.APÈNDIX

Registre Pregunta – Resposta - Context

Cuanto cuestan los autobuses?

Los autobuses son gratis.

{{'id': '2',

'metadata': {'text': 'AUTOBUSES Los autobuses son gratis Es necesario '

'inscribirse previamente. El punto de llegada de los'

'autobuses es la Avenida Ramón y Cajal (entrada '

'principal del parque).'},

'score': 0.442504942,

'values': []}, {'id': '14',

'metadata': {'text': 'A las 18:10 concierto tremenda jauría en el escenario '

'principal. Esta noche nos traerá sus ritmos latinos y '

'«electro-cumbia». Este colectivo musical fusiona '

'cumbia-punk, rap, reguetón combativo y música '

'electrónica, con evidentes elementos de la música '

'latinoamericana. Sin duda, una propuesta singular que '

'no nos dejará indiferentes.'},

'score': 0.405895233,

'values': []}, {'id': '13',

'metadata': {'text': 'A la 17:45 concurso DJ Escenario 2 No te pierdas la '

'oportunidad de pinchar en la Fiesta Mayor de la URV. '

'Cada actuación tendrá una duración de 25 minutos. Se '

'decidirá la mejor sesión de DJ a partir de las '

'valoraciones de un jurado y votación popular. Cada '

'valoración contará un 50 %. La entrega de premios '

'tendrá lugar a las 19.30 h. Es necesario inscribirse '

'previamente. ¡La persona ganadora actuará en la Fiesta '

'Mayor de la URV 2025!'},

'score': 0.377972752,

De donde salen los autobuses?

Los autobuses salen de Campus Vilafranca del Penedès, Baix Penedès y Terres de l'Ebre.

{{'id': '3',

'metadata': {'text': 'De donde salen los autobuses Campus Vilafranca del '

'Penedès: Salida: 10:45h Llegada: 23:00h Baix Penedès: '
'Salida: 11:00h Llegada: 23:00h Terres de l'Ebre: '
'Salida:11:30h Llegada: 23:00h'},
'score': 0.49602282,
'values': [], {'id': '2',
'metadata': {'text': 'AUTOBUSES Los autobuses son gratis Es necesario '
'inscribirse previamente. El punto de llegada de los '
'autobuses es la Avenida Ramón y Cajal (entrada '
'principal del parque).'},
'score': 0.493816316,
'values': [], {'id': '14',
'metadata': {'text': 'A las 18:10 concierto tremenda jauría en el escenario '
'principal. Esta noche nos traerá sus ritmos latinos y '
'«electro-cumbia». Este colectivo musical fusiona '
'cumbia-punk, rap, reguetón combativo y música '
'electrónica, con evidentes elementos de la música '
'latinoamericana. Sin duda, una propuesta singular que '
'no nos dejará indiferentes.'},
'score': 0.448899865,

Cuando son los parlamentos?

A la 1:30 pasaran los parlamentos. Parlamentos institucionales de Josep Pallarès Marzal , rector de la universidad

'metadata': {'text': 'A la 1:30 pasaran los parlamentos. Parlamentos '
'institucionales de Josep Pallarès Marzal , rector de '
'la Universidad Rovira i Virgili, y Rubén Viñuales '
'Elías, alcalde de Tarragona.'},
'score': 0.500889421,
'values': [], {'id': '20',
'metadata': {'text': 'Almuerzo popular Para pedir la sartén o bien la '
'fideuada será necesario escanear el mismo QR de la '
'entrada siempre que se haya adquirido ticket de '
'almuerzo. Por tanto, una vez haya accedido al recinto '
'no borre su entrada'},
'score': 0.543595552,
'values': [], {'id': '24',

'metadata': {'text': '¿Cómo funciona el concurso de camisetas? Las personas ' 'que quieran participar podrán apuntarse el mismo día de ' 'la Fiesta Mayor en la carpa de la URV, hasta las 13 h. ' 'Podéis imprimir las camisetas o hacéros las con pintura ' 'o rotuladores. Pueden participar los grupos o ' 'colectivos que quieran, tanto de estudiantes como de ' 'personal laboral, funcionario, investigador, ' 'docente...'},

'score': 0.529889643,

Que estilo de musica hace tremenda jauría?

Tremenda jauría es un grupo musical que fusiona cumbia-punk, rap, reguetón combativo y música electrónica

'metadata': {'text': 'A las 18:10 concierto tremenda jauría en el escenario ' 'principal. Esta noche nos traerá sus ritmos latinos y ' '«electro-cumbia». Este colectivo musical fusiona ' 'cumbia-punk, rap, reguetón combativo y música ' 'electrónica, con evidentes elementos de la música ' 'latinoamericana. Sin duda, una propuesta singular que ' 'no nos dejará indiferentes.'},

'score': 0.635077059,

'values': [], {'id': '4',

'metadata': {'text': 'A las 12 la charanga Pujats de To nos acuta primero. ' 'Música y diversión recorrerán el Parque de la Ciudad e ' 'invitarán a la comunidad universitaria a participar en ' 'esta jornada tan especial.'},

'score': 0.475277662,

'values': [], {'id': '15',

'metadata': {'text': 'A las 19:45 h concierto les que falteband en el ' 'escenario principal A continuación, llegan Les Que ' 'Faltaband, la banda de versiones formada en 2020 por ' 'ocho mujeres que nos ofrecen un espectáculo potente y ' 'reivindicativo con una puesta en escena que combina la ' 'música, la danza y otros elementos artísticos. La banda ' 'nos presenta un repertorio íntegramente femenino, ' 'desde ritmos latinos y pop-rock de los ochenta hasta ' 'los hits más actuales.'},

'score': 0.460269123,

A que hora es el bingo?

A la 15:30 h BINGO en el escenario 2, Los cartones de bingo se darán con las consumiciones de barra

'metadata': {'text': 'A que hora es el bingo? A la 15:30 h BINGO en el ' 'escenario 2, Los cartones de bingo se darán con las ' 'consumiciones de barra.'},

'score': 0.710502744,

'values': [], {'id': '19',

'metadata': {'text': 'Acceso a la Fiesta La primera vez que se accede al ' 'recinto será necesario mostrar el código QR de la ' 'entrada. Una vez hecho, tendrá una pulsera que le ' 'permitirá entrar y salir sin necesidad del código QR. ' 'Si no he comprado entrada, ¿puedo comprarla el mismo ' 'día? Únicamente si no se han agotado todas las ' 'entradas, que son limitadas. Te invitamos a adquirir ' 'tu entrada lo antes posible.'},

'score': 0.514514,

'values': [], {'id': '20',

'metadata': {'text': 'Almuerzo popular Para pedir la sartén o bien la ' 'fideuada será necesario escanear el mismo QR de la ' 'entrada siempre que se haya adquirido ticket de ' 'almuerzo. Por tanto, una vez haya accedido al recinto ' 'no borre su entrada'}},

'score': 0.493533105,

Se podra jugar a futbol?

No, no se podra jugar a futbol

'metadata': {'text': 'La fiesta empieza a las 12 con la banda Pujats de To. Y ' 'termina a las 23 con la música de DJ Trapella.'},

'score': 0.332653344,

'values': [], {'id': '24',

'metadata': {'text': '¿Cómo funciona el concurso de camisetas? Las personas ' 'que quieran participar podrán apuntarse el mismo día de ' 'la Fiesta Mayor en la carpa de la URV, hasta las 13 h. ' 'Podéis imprimir las camisetas o hacéros las con pintura ' 'de colores.'},

'o rotuladores. Pueden participar los grupos o '
'colectivos que quieran, tanto de estudiantes como de '
'personal laboral, funcionario, investigador, '
'docente...'),

'score': 0.278434873,

'values': [], {'id': '4',

'metadata': {'text': 'A las 12 la charanga Pujats de To nos acuta primero. '
'Música y diversión recorrerán el Parque de la Ciudad e '
'invitarán a la comunidad universitaria a participar en '
'esta jornada tan especial.'},

'score': 0.271757871,

Que pasa si no me gusta la paella?

Si no me gusta la paella, puedo llevar mi comida al recinto del Parc de la Ciutat. También puedo comer en un foodtruck

[{'id': '21',

'metadata': {'text': '. No me gusta la paella ni la fideuá. ¿Existe alguna '
'alternativa? En el recinto encontrarás foodtrucks. '
'También puedes llevar tu comida. ¿Puedo llevar '
'comida? Sí. Puedes llevar tu propia comida. Recuerda '
'que no podrás entrar con objetos de cristal al recinto '
'del Parc de la Ciutat.'},

'score': 0.702827394,

'values': [], {'id': '8',

'metadata': {'text': 'Entre las 13:45 y 14:45 h fideua y paella popular. '
'Saborea un plato de fideuá tradicional o de paella '
'vegana (apto para celíacos y alérgicos al marisco*) y '
'comparte almuerzo con toda la comunidad URV. Almuerzo '
'de pie. También puedes traer tu comida. Todo el día '
'también habrá DJ. *Puede existir contaminación '
'cruzada.'},

'score': 0.451822847,

'values': [], {'id': '19',

'metadata': {'text': 'Acceso a la Fiesta La primera vez que se accede al '
'recinto será necesario mostrar el código QR de la '
'entrada. Una vez hecho, tendrá una pulsera que le '

LLAMAINDEX

```
import logging
import sys

logging.basicConfig(stream=sys.stdout, level=logging.INFO)
logging.getLogger().addHandler(logging.StreamHandler(stream=sys.stdout))

from llama_index.core import
VectorStoreIndex, SimpleDirectoryReader, ServiceContext, PromptTemplate
from llama_index.llms.huggingface import HuggingFaceLLM
documents = SimpleDirectoryReader("/content/Data/").load_data()
from llama_index.core.prompts.prompts import SimpleInputPrompt

system_prompt = "Eres un asistente. Tu objetivo es responder preguntas de
manera precisa con la informacion que se te proporciona."

# This will wrap the default prompts that are internal to llama-index
query_wrapper_prompt =
SimpleInputPrompt("<|USER|>{query_str}<|ASSISTANT|>")
import torch
from transformers import AutoTokenizer

llm = HuggingFaceLLM(
    context_window=4096,
    max_new_tokens=256,
    generate_kwargs={"temperature": 0.0, "do_sample": False},
    system_prompt=system_prompt,
    query_wrapper_prompt=query_wrapper_prompt,
    tokenizer_name="meta-llama/Meta-Llama-3-8B-Instruct",
    model_name="meta-llama/Meta-Llama-3-8B-Instruct",
    device_map="auto",
    # uncomment this if using CUDA to reduce memory usage
    model_kwargs={"torch_dtype": torch.float16, "load_in_8bit": True}
)
from langchain.embeddings.huggingface import HuggingFaceEmbeddings
from llama_index.embeddings.langchain import LangchainEmbedding
from llama_index.core import Settings

embed_model = LangchainEmbedding(
    HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-
v2")
)
service_context = ServiceContext.from_defaults(
    chunk_size=1024,
    llm=llm,
    embed_model=embed_model
)
index = VectorStoreIndex.from_documents(documents,
service_context=service_context)
```

```

query_engine = index.as_query_engine()

response = query_engine.query("Des d'on sortiran els autobusos que van a la
festa major?, contesta en català")

print(response)

```

Funció Lambda per a enviar missatges al LLM

```

import boto3
import json

ENDPOINT_NAME = "llama-3-generator"
runtime = boto3.client('sagemaker-runtime')

def lambda_handler(event, context):
    try:
        # Logging input event
        print("Received event:", json.dumps(event))

        # Get query
        body = json.loads(event['body'])
        text_input = body.get('inputs', '')
        print("Text input:", text_input)

        # Invoke endpoint in SageMaker
        response = runtime.invoke_endpoint(
            EndpointName=ENDPOINT_NAME,
            ContentType='application/json',
            Body=json.dumps({"inputs": text_input})
        )

        # Read response
        response_content = response['Body'].read().decode()
        result = json.loads(response_content)
        print("SageMaker response:", result)

    return {

```

```

        "statusCode": 200,
        "body": json.dumps(result)
    }

except KeyError as e:
    print(f"KeyError: {str(e)}")
    return {
        "statusCode": 400,
        "body": json.dumps({"error": f"Missing key: {str(e)}"})
    }

except Exception as e:
    print(f"Exception: {str(e)}")
    return {
        "statusCode": 500,
        "body": json.dumps({"error": str(e)})
    }

```

Funció Lambda per a enviar missatges a l'embedder

```

import boto3
import json

ENDPOINT_NAME = "minilm-demo"
runtime = boto3.client('sagemaker-runtime')

def lambda_handler(event, context):
    try:
        # Logging input event
        print("Received event:", json.dumps(event))

        # Get query
        body = json.loads(event['body'])
        text_input = body.get('inputs', '')
        print("Text input:", text_input)

        # Invoke endpoint in SageMaker
        response = runtime.invoke_endpoint(

```

```

        EndpointName=ENDPOINT_NAME,
        ContentType='application/json',
        Body=json.dumps({"inputs": text_input})
    )

    # Send response
    response_content = response['Body'].read().decode()
    result = json.loads(response_content)
    print("SageMaker response:", result)

    return {
        "statusCode": 200,
        "body": json.dumps(result)
    }

except KeyError as e:
    print(f"KeyError: {str(e)}")
    return {
        "statusCode": 400,
        "body": json.dumps({"error": f"Missing key: {str(e)}"})
    }

except Exception as e:
    print(f"Exception: {str(e)}")
    return {
        "statusCode": 500,
        "body": json.dumps({"error": str(e)})
    }
}

```

Información de la festa major feta servir per a crear l'índex

Fiesta Mayor URV

El próximo 14 de marzo tenemos una cita festiva universitaria en el Parque de la Ciudad de Tarragona para disfrutar de diversas actividades como una paella popular, conciertos, un concurso de talentos, un concurso de DJ, un bingo, un concurso de camisetas, foodtrucks, carpas, etc.

La fiesta empieza a las 12 con la banda Pujats de To. Y termina a las 23 con la música de DJ Trapella.

AUTOBUSES

Los autobuses son gratis

Es necesario inscribirse previamente.

El punto de llegada de los autobuses es la Avenida Ramón y Cajal (entrada principal del parque).

De donde salen los autobuses

Campus Vilafranca del Penedès: Salida: 10:45h Llegada: 23:00h

Baix Penedès: Salida: 11:00h Llegada: 23:00h

Terres de l'Ebre: Salida: 11:30h Llegada: 23:00h

A las 12 la charanga Pujats de To nos acuta primero. Música y diversión recorrerán el Parque de la Ciudad e invitarán a la comunidad universitaria a participar en esta jornada tan especial.

A la 1 pasara la inscripción en el concurso de eslóganes de camisetas en la carpa URV, el pilar de los pataquers y el dragon chino, por la colla castellera de la URV y la comunidad china de la URV.

A la 1:30 pasaran los parlamentos. Parlamentos institucionales de Josep Pallarès Marzal , rector de la Universidad Rovira i Virgili, y Rubén Viñuales Elías, alcalde de Tarragona.

A la 13:40 PD peting, amenizará el mediodía el dúo de pinchadiscos tarraconenses PD Peting a golpe de ritmos electrónicos y bailables. Nos prometen un rato de jolgorio y show con la mejor música, ¡no se lo pierdan!

Entre las 13:45 y 14:45 h fideua y paella popular. Saborea un plato de fideuá tradicional o de paella vegana (apto para celíacos y alérgicos al marisco*) y comparte almuerzo con toda la comunidad URV. Almuerzo de pie. También puedes traer tu comida. Todo el día también habrá DJ. *Puede existir contaminación cruzada.

Ten en cuenta que no puedes entrar bebida u objetos de vidrio en el Parc de la Ciutat.

A que hora es el bingo?

A la 15:30 h BINGO en el escenario 2, Los cartones de bingo se darán con las consumiciones de barra.

Entre las 16:00 y 18:00 h feria

Carpas de las asociaciones: Llenguas URV, Tienda URV, Colectivo cultural Alma Mater, Colla Castellera, Universitaria los Pataquers de la URV, Asociación URVoltage Racing, Oficina URV, Aurora, Club de Ajedrez Tarragona con Deportes URV, Asociación Urbots, ASBTEC, Reto experimenta, Asociación URVisibles, Doctorandos en lucha, Colectivo de Estudiantes Chinos, Asociación Intro-Ment, Tarragona Jove - Servicio Municipal de Juventud del Ayuntamiento de Tarragona, Som.NIT (Cruz Roja Juventud), Asociación Quilómetro Cero.

A la 16:00 Concurso de talentos en el Escenario 2

Explícanos qué sabes hacer y actúa en la Fiesta Mayor de la URV.

Cada actuación tendrá una duración de 5 minutos. Se decidirá la mejor actuación del micro abierto a partir de las valoraciones de un jurado y votación popular. Cada actuación contará un 50%. La entrega de premios tendrá lugar a las 18 h. Es necesario inscribirse previamente.

A la 17:45 concurso DJ Escenario 2

No te pierdas la oportunidad de pinchar en la Fiesta Mayor de la URV.

Cada actuación tendrá una duración de 25 minutos. Se decidirá la mejor sesión de DJ a partir de las valoraciones de un jurado y votación popular. Cada valoración contará un 50%. La entrega de premios tendrá lugar a las 19.30 h.

Es necesario inscribirse previamente.

¡La persona ganadora actuará en la Fiesta Mayor de la URV 2025!

A las 18:10 concierto tremenda jauría en el escenario principal.

Esta noche nos traerá sus ritmos latinos y «electro-cumbia». Este colectivo musical fusiona cumbia-punk, rap, reguetón combativo y música electrónica, con evidentes elementos de la música latinoamericana. Sin duda, una propuesta singular que no nos dejará indiferentes.

A las 19:45 h concierto les que falteband en el escenario principal

A continuación, llegan Les Que Falteband, la banda de versiones formada en 2020 por ocho mujeres que nos ofrecen un espectáculo potente y reivindicativo con una puesta en escena que combina la música, la danza y otros elementos artísticos. La banda nos presenta un repertorio íntegramente femenino, desde ritmos latinos y pop-rock de los ochenta hasta los hits más actuales.

A las 21:30 DJ trapella en el escenario principal

Cierra la Fiesta Mayor la pinchadiscos DJ Trapella, quien creará una fiesta ecléctica, mezclando los logros más actuales con los más alternativos, con una sesión dinámica, fresca y sorprendente. Mezcla un gran abanico de estilos musicales en directo, lo que permite englobar a todo el público, para que nadie quede indiferente.

Sigue toda la actualidad del evento por redes sociales con el hashtag #festamajorurv

¿Tienes ideas de mejora? ¿Quieres colaborar? ¿Te gustaría participar en la organización de la Fiesta Mayor 2025? Contacta con la Comisión de la Fiesta Mayor.

Acceso a la Fiesta

La primera vez que se accede al recinto será necesario mostrar el código QR de la entrada.

Una vez hecho, tendrá una pulsera que le permitirá entrar y salir sin necesidad del código QR.

Si no he comprado entrada, ¿puedo comprarla el mismo día?

Únicamente si no se han agotado todas las entradas, que son limitadas. Te invitamos a adquirir tu entrada lo antes posible.

Almuerzo popular

Para pedir la sartén o bien la fideuada será necesario escanear el mismo QR de la entrada siempre que se haya adquirido ticket de almuerzo. Por tanto, una vez haya accedido al recinto no borre su entrada

.

No me gusta la paella ni la fideuá. ¿Existe alguna alternativa?

En el recinto encontrarás foodtrucks. También puedes llevar tu comida.

¿Puedo llevar comida?

Sí. Puedes llevar tu propia comida. Recuerda que no podrás entrar con objetos de cristal al recinto del Parc de la Ciutat.

¿Puedo llevar bebida?

No. Únicamente se puede consumir la bebida adquirida en la Fiesta Mayor. Habrá agua al alcance de todos.

He comprado entrada, ¿puedo salir y volver a entrar?

Sí. Siempre y cuando la pulsera esté en buen estado, podrás entrar y salir del recinto.

¿Cómo funciona el concurso de camisetas?

Las personas que quieran participar podrán apuntarse el mismo día de la Fiesta Mayor en la carpa de la URV, hasta las 13 h. Podéis imprimir las camisetas o hacéros las con pintura o rotuladores. Pueden participar los grupos o colectivos que quieran, tanto de estudiantes como de personal laboral, funcionario, investigador, docente...

Acceso con control de entrada, ¡no te quedes sin entrada/pulsera!

Queda prohibido acceder al recinto con banderas y/o pancartas que tengan carácter ideológico o político. En consecuencia, la organización del evento podrá retirar este tipo de material en cualquier momento antes del acceso al parque o bien en el interior, que será devuelto a la salida del recinto.

La URV es la universidad pública del sur de Cataluña, arraigada en el territorio, con visión internacional, con docentes cercanos y excelente en la investigación.

Contacta con nosotros

Síguenos en Facebook, Twitter, LinkedIn, Youtube, Instagram

Deploy d'un model des de Sagemaker

```
from sagemaker.jumpstart.model import JumpStartModel
```

```

my_model = JumpStartModel(
    model_id="meta-textgeneration-llama-3-8b"
)

import sagemaker
from sagemaker.jumpstart.model import JumpStartModel
from sagemaker.huggingface import HuggingFaceModel

role = sagemaker.get_execution_role()

my_model = JumpStartModel(model_id="meta-textgeneration-llama-3-8b")

llm = my_model.deploy(
    initial_instance_count=1,
    instance_type="ml.g5.2xlarge",
    endpoint_name="llama-3-generator-v5",
    accept_eula=True
)

```

index.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="styles.css">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:op
sz,wght,FILL,GRAD@48,400,0,0" />
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Rounded:ops
z,wght,FILL,GRAD@48,400,1,0" />
  </head>

```

```

<body>
  <button class="chatbot-toggler">
    <span class="material-symbols-rounded">mode_comment</span>
    <span class="material-symbols-outlined">close</span>
  </button>

  <div class="chatbot">
    <header>
      <h2>ChatURV</h2>
      <span class="material-symbols-outlined">close</span>
    </header>
    <ul class="chatbox">
      <li class="chat incoming">
        <span class="material-symbols-outlined">smart_toy</span>
        <p>Hola <br>En que puedo ayudar?</p>
      </li>
    </ul>
    <div class="chat-input">
      <textarea id="chatInput" placeholder="Escribe una pregunta..."
      spellcheck="false" required></textarea>
      <span id="send-btn" class="material-symbols-rounded">send</span>
    </div>
  </div>

  <script src="scripts.js"></script>
</body>
</html>

```

scripts.js

```

const chatInput = document.querySelector(".chat-input textarea");
const sendChatBtn = document.querySelector(".chat-input span");
const chatbox = document.querySelector(".chatbox");
const chatbotToggler = document.querySelector(".chatbot-toggler");

```

```

const inputInitHeight = chatInput.scrollHeight;

const LLM_URL = "https://xqqqltmdnb.execute-api.us-east-1.amazonaws.com/default/pythonQuery";

const maxSectionLen = 1000;

const separator = "\n";

let userMessage = null;

const createChatLi = (message, className) => {
  // Create a chat <li> element with passed message and className
  const chatLi = document.createElement("li");
  chatLi.classList.add("chat", `${className}`);

  let chatContent = className === "outgoing" ? `<p></p>` : `<span class="material-symbols-outlined">smart_toy</span><p></p>`;

  chatLi.innerHTML = chatContent;

  chatLi.querySelector("p").textContent = message;

  return chatLi; // return chat <li> element
}

const handleChat = () => {
  userMessage = chatInput.value.trim(); // Get user entered message and
  remove extra whitespace

  if(!userMessage) return;

  chatInput.value = "";

  chatInput.style.height = `${inputInitHeight}px`

  // Append the user's message to the chatbox
  chatbox.appendChild(createChatLi(userMessage, "outgoing"));

  chatbox.scrollTo(0, chatbox.scrollHeight);

  setTimeout(() => {
    // Display "Pensando..." message while waiting for the response
    const incomingChatLi = createChatLi("Pensando...", "incoming");
    chatbox.appendChild(incomingChatLi);
  }, 1000);
}

```

```

        chatbox.scrollTo(0, chatbox.scrollHeight);
        generateResponse(incomingChatLi);
    }, 600);
}

const generateResponse = async (chatElement) => {

    const messageElement = chatElement.querySelector("p");

    const response_pinecone = await fetch(`/search?query=' +
    encodeURIComponent(userMessage));

    if (!response_pinecone.ok) {
        throw new Error(`HTTP error! Status: ${response_pinecone.status}`);
    }

    const responseText = await response_pinecone.text();

    console.log("Generated:", responseText);
    messageElement.textContent=responseText;
    chatbox.scrollTo(0, chatbox.scrollHeight);
}

sendChatBtn.addEventListener("click", handleChat);

chatbotToggler.addEventListener("click", () =>
document.body.classList.toggle("show-chatbot"));

chatInput.addEventListener("input", () => {
    // Adjust the height of the input textarea based on its content
    chatInput.style.height = `${inputInitHeight}px`;
    chatInput.style.height = `${chatInput.scrollHeight}px`;
});

chatInput.addEventListener("keydown", (e) => {
    // If Enter key is pressed without Shift key and the window
    // width is greater than 800px, handle the chat

```

```

    if(e.key === "Enter" && !e.shiftKey && window.innerWidth > 800) {
        e.preventDefault();
        handleChat();
    }
});

```

server.js - funció per a enviar una petició al LLM

```

async function ragQuery(contexts, question) {
    try {
        const contextStr = createContext(contexts);
        const textInput = promptTemplate.replace("{context}",
contextStr).replace("{question}", question);
        const payload = {
            inputs: textInput
        };
        const headers = { "Content-Type": "application/json" };
        const response = await fetch(LLM_URL, {
            method: "POST",
            headers: headers,
            body: JSON.stringify(payload)
        });
        if (response.ok) {
            const result = await response.json();
            const generatedText = result.generated_text;
            return generatedText;
        } else {
            return `Error: ${response.status}\nResponse:
${response.statusText}`;
        }
    } catch (error) {
        console.error("An error occurred:", error);
        return `An error occurred: ${error.message}`;
    }
}

```