UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Entrenament d'un model de IA per a la generació de prediccions meteorològiques en llenguatge natural.

Autor: Daniel Ramos Pérez

Director: Dr. Albert Clapés Sintes
Realitzat a: Departament de Matemàtiques i Informàtica.

Barcelona, 10 de juny de 2025

Resum

Aquest Treball de Fi de Grau presenta el disseny, la implementació i l'avaluació d'un sistema de generació de textos meteorològics automatitzat basat en intel·ligència artificial, concretament mitjançant el model de llenguatge Qwen2 afinat amb la tècnica LoRA. El procés comença amb la recopilació i neteja de dades històriques proporcionades per Troposferica -temperatura, humitat, precipitacions, vent, etc.-, a les quals s'aplica un exhaustiu preprocessat (normalització, eliminació de valors atípics i selecció de variables). A continuació, s'entrena Qwen2-LoRA per generar descripcions textuals de les condicions meteorològiques futures a partir d'aquestes dades, ajustant hiperparàmetres com la mida de lot, el nombre d'èpoques i la temperatura de mostreig.

La generació automàtica produeix butlletins meteorològics que resumeixen previsions de temperatura, probabilitat de pluja i vent, amb un estil coherent i comprensible gràcies a l'entrenament amb textos reals de l'empresa. Per validar la seva fiabilitat, es comparen quantitativament les prediccions amb els butlletins de referència (*groundtruth*) mitjançant mètriques com ROUGE i BLEU, i s'avalua la qualitat semàntica del text generat amb BERTScore i una anàlisi qualitativa realitzada per una meteoròloga experta. Els resultats mostren que el sistema assoleix puntuacions relativament altes.

Aquest treball demostra la viabilitat d'utilitzar models de llenguatge adaptats per a la predicció meteorològica, ressaltant avantatges com l'automatització de la redacció, la personalització de butlletins i l'estalvi de temps als meteoròlegs, al mateix temps que es manté un nivell de precisió adequat per a aplicacions pràctiques.

Abstract

This Bachelor's Thesis presents the design, implementation, and evaluation of an automated weather bulletin generation system powered by artificial intelligence, specifically using the Qwen2 language model fine-tuned with the LoRA technique. The process begins with the collection and cleaning of historical data provided by Troposferica—temperature, humidity, precipitation, wind, etc.—followed by thorough preprocessing (normalization, outlier removal, and feature selection). Next, Qwen2-LoRA is trained to generate textual descriptions of future weather conditions from these inputs, tuning hyperparameters such as batch size, number of epochs, and sampling temperature.

The automated pipeline produces weather bulletins summarizing temperature forecasts, precipitation probability, and wind conditions in a coherent and understandable style, thanks to training on company-specific texts. To validate reliability, predictions are quantitatively compared against ground truth bulletins using metrics such as ROUGE and BLEU, and semantic text quality is assessed with BERTScore as well as a qualitative review by an expert meteorologist. The results demonstrate that the system achieves relatively high scores across all evaluated metrics.

This work proves the feasibility of leveraging adapted language models for weather forecasting, highlighting benefits such as automated report writing, bulletin customization, and significant time savings for meteorologists, while maintaining the precision required for practical deployment.

Resumen

Este Trabajo de Fin de Grado presenta el diseño, la implementación y la evaluación de un sistema de generación automática de textos meteorológicos basado en inteligencia artificial, concretamente mediante el modelo de lenguaje Qwen2 afinado con la técnica LoRA. El proceso comienza con la recopilación y limpieza de datos históricos proporcionados por Troposferica —temperatura, humedad, precipitaciones, viento, etc.— a los que se aplica un exhaustivo preprocesamiento (normalización, eliminación de valores atípicos y selección de variables). A continuación, se entrena Qwen2-LoRA para generar descripciones textuales de las condiciones meteorológicas futuras a partir de estos datos, ajustando hiperparámetros como el tamaño de lote, el número de épocas y la temperatura de muestreo.

La generación automática produce boletines meteorológicos que resumen previsiones de temperatura, probabilidad de lluvia y viento, con un estilo coherente y comprensible gracias al entrenamiento con textos reales de la empresa. Para validar su fiabilidad, las predicciones se comparan cuantitativamente con los boletines de referencia (*ground truth*) mediante métricas como ROUGE y BLEU, y la calidad semántica del texto generado se evalúa con BERTScore y un análisis cualitativo realizado por una meteoróloga experta. Los resultados muestran que el sistema alcanza puntuaciones relativamente altas en todas las métricas consideradas.

Este trabajo demuestra la viabilidad de utilizar modelos de lenguaje adaptados para la predicción meteorológica, resaltando ventajas como la automatización de la redacción, la personalización de boletines y el ahorro de tiempo para los meteorólogos, al mismo tiempo que se mantiene un nivel de precisión adecuado para aplicaciones prácticas.

Agraïments

Vull agrair, primer de tot, a l'Albert Clapés, el meu tutor, qui ha estat constantment ajudant i recolzant aquest treball, i que, tot i tenir ambdós horaris complicats, s'ha esforçat a estar present en tot moment, sense ell aquest treball no hagués sigut possible.

Als meus pares, per haver-me transmès des de ben petit els valors de l'esforç i la perseverança, i per haver sacrificat hores i recursos en cada pas del meu camí, gràcies per construir-me un entorn carregat d'amor, seguretat i confiança. El vostre exemple m'ha inspirat a afrontar reptes amb fermesa i a valorar cada assoliment, i sense el vostre suport incondicional aquest projecte no hauria estat possible.

A la Glòria, la meva parella, pel seu amor i suport constants, gràcies per ser la meva companya de vida, per il·luminar amb la teva presència fins els dies més exigents i per intentar entendre'm quan fins i tot jo no ho feia. El teu amor m'ha donat força i confiança per fer realitat aquest projecte.

Als meus germans, que amb la seva rialla, innocència i entusiasme incondicional m'han recordat constantment la bellesa de les coses senzilles i m'han impulsat a continuar endavant quan més necessitava un motiu per somriure.

Als meus avis, plens de saviesa i tendresa, que m'han enfortit amb històries de vida, amb abraçades carregades de caliu i amb aquell exemple de constància que m'ha mostrat que, fins i tot en els dies més grisos, sempre hi ha una llum a la qual fer-se camí. Gràcies a tots ells per ser el seu suport incondicional i per fer possible que aquest projecte prengués forma.

Al grup d'amics de Sant Celoni i al grup d'amics de la universitat, gràcies per ser el meu motor en els moments difícils: per fer-me riure amb les vostres bromes, per compartir cafès (birres) i converses, i per oferir-me sempre un alè d'ànim. Sense tenir-vos al costat, aquest projecte hauria estat molt més solitari; gràcies per creure en mi.

També a l'empresa Troposferica, per cedir-me les dades que m'han permès entrenar el model i per permetre'm descobrir l'interessant món de la meteorologia.

Finalmente, dedico este traballo ao meu avo, un dos piares fundamentais que fixeron de min quen son hoxe. Á súa memoria e ao seu legado, expreso a miña gratitude máis profunda; estou certo de que, alí onde estea, continuará orgulloso de cada un dos meus pasos e esforzos. Onte, 9 de xuño, cumpríronse seis anos dende que nos deixou, e cada día o levo vivo no pensamento.

"E cando as bágoas corren, sei que é o seu amor o que aínda me guía, lembrándome que a ausencia máis profunda pode acoller o amor máis eterno."

Índex

1 Introducció

Motivació

L'aire que respirem, el sol que escalfa la pell i les gotes que acaricien el sòl tenen un mateix origen: una complexa dansa atmosfèrica regida per lleis físiques i patrons difícils de percebre a simple vista. Avui dia, els serveis meteorològics tradicionals han fet servir models numèrics que requereixen grans recursos de càlcul i experiència especialitzada per transformar equacions diferencials en mapes i taules de valors. Però, i si poguéssim fer que una intel·ligència artificial, capaç d'entendre i generar llenguatge humà, digerís milions de registres de temperatura, humitat i vent per oferir-nos explicacions clares i personalitzades en qüestió de segons?

En aquest treball de fi de grau, es presenta un sistema que fa exactament això: a partir de dades històriques proporcionades per Troposferica, s'entrena un model de llenguatge de gran capacitat (Qwen2) ajustat amb la tècnica LoRA, per transformar números en narracions meteorològiques. El resultat no és només una sèrie de prediccions massives, sinó butlletins que expliquen el que passarà amb el temps amb llenguatge natural i, per tant, de manera més humana. Des de l'augment de la temperatura fins a la probabilitat de tempestes, tot explicat amb coherència i precisió.

Aquest enfocament no només automatitza la redacció de butlletins, sinó que obre la porta a informatitzar serveis hiperlocals, adaptats a la teva ciutat o fins i tot al teu carrer. A més, la validació numèrica i qualitativa demostra que aquest model no sacrifica precisió estadística per la comoditat del llenguatge: aplica mètriques modernes (ROUGE, BLEU, anàlisi BERTScore) i validació per experts per assegurar un equilibri òptim entre rigor i llegibilitat.

A mesura que s'exploren les capes de la nostra metodologia (des del tractament de dades i l'entrenament del model fins a l'avaluació final, el lector descobrirà com la intel·ligència artificial pot convertir la predicció meteorològica en una experiència accessible, eficient i personalitzada. Benvinguts a la nova era de la previsió del temps: més humana, més àgil, més intel·ligent.

1.1 Conceptes previs

Per tal d'agilitzar l'enteniment, aquí s'esmenten alguns dels conceptes clau del projecte, tot i que després s'aprofundirà en els que tenen relació més directa amb la metodologia seguida.

- **Machine Learning** (***aprenentatge automàtic***) és un camp de la intel·ligència artificial dedicat al disseny d'algorismes que permeten a les màquines aprendre a partir de dades. L'objectiu és identificar patrons en grans volums d'informació i utilitzar-los per fer modelar els patrons que permeten, llavors, fer predicció de variables objectives o generació de dades plausibles.
- **Pesos:** en una xarxa neuronal, els pesos són els valors que s'aprenen i ajusten

durant l'entrenament per definir la força de la connexió entre dues neurones. Cada vegada que es presenta un exemple al model, aquests pesos es modifiquen mitjançant retropropagació per minimitzar l'error de predicció.

- **Biaixos:** són valors constants afegits a la sortida de cada neurona que permeten ajustar la funció d'activació en cada capa. Els biaixos també s'aprenen durant l'entrenament i treballen conjuntament amb els pesos per determinar com es transformen les entrades en sortides.

- **Hiperparàmetre:** és un valor extern al model que no s'aprèn durant l'entrenament, sinó que s'ajusta prèviament per optimitzar el procés d'aprenentatge. Aquests inclouen, entre d'altres, la mida del *batch* (nombre d'exemples processats abans d'actualitzar els pesos), la taxa d'aprenentatge (learning rate), el nombre d'èpoques (vegades que es recorre el conjunt d'entrenament), el dropout (percentatge de connexions anul·lades durant l'entrenament) i els paràmetres específics de LoRA com r (rank) i α.

- **Xarxa neuronal** és un model matemàtic i computacional inspirat en el cervell, format per un conjunt de nodes interconnectats que processen la informació seguint una aproximació connexionista. Cada node (neurona artificial) té pesos i biaixos associats que s'ajusten durant l'entrenament, de manera que la xarxa pot generalitzar patrons complexos a partir de les dades.

- **Entrenament** és el procés d'ajustar iterativament els paràmetres del model mitjançant un algoritme d'optimització per aprendre de les dades. Al començament, els pesos i biaixos del model s'inicialitzen aleatòriament. A continuació, es fa una passada avançada sobre els exemples d'entrenament, es calcula la pèrdua i, mitjançant retropropagació, s'actualitzen els paràmetres per minimitzar l'error. Aquest cicle s'itera fins que el model aprèn a generalitzar bé sobre les dades.

- **Fine-tuning** (***ajust fi***) és el procés d'adaptar un model preentrenat a una tasca o domini específic. En lloc d'entrenar un model des de zero, es realitza un entrenament addicional amb dades concretes de la nova tasca, cosa que accelera l'aprenentatge en la nova tasca (o domini) i redueix els recursos necessaris.

- **Inferència** és el procés d'utilitzar el model ja entrenat per fer prediccions sobre dades noves. Un cop s'han ajustat els paràmetres durant l'entrenament, la inferència consisteix a subministrar al model un conjunt d'entrades (per exemple, valors meteorològics) i obtenir la sortida generada (el butlletí meteorològic). Durant aquesta fase, el model aplica directament les seves funcions d'activació i pesos fixos sense modificar-se; per tant, no es realitza cap actualització de paràmetres. L'eficiència en inferència és crucial per desplegar el model en entorns reals i proporcionar respostes en temps raonable.

- **Validació** és l'avaluació del model amb dades que no s'han usat en l'entrenament. El conjunt de validació proporciona una mesura imparcial del rendiment del model mentre s'ajusten els hiperparàmetres. Durant l'entrenament,

se sol monitorar l'error sobre aquest conjunt per detectar sobreajustament i determinar el moment òptim per aturar l'aprenentatge.

- **Transformer** és una arquitectura de xarxa neuronal especialment dissenyada per processar seqüències de dades en paral·lel mitjançant mecanismes d'atenció. A diferència dels models seqüencials tradicionals, durant l'entrenament, els transformers processen totes les paraules d'un text simultàniament, capturant les relacions contextuals a llarg abast.

- **LLM** (***Large Language Model***) és un model de llenguatge extens basat en aprenentatge automàtic, dissenyat per dur a terme tasques de comprensió i generació de llenguatge natural. Aquests models tenen un nombre molt gran de paràmetres i es preentrenen amb enormes quantitats de text mitjançant, en certs models com BERT, aprenentatge autodirigit, fet que els permet generar respostes coherents i contextuals.

- **LoRA** (***Low-Rank Adaptation***) és una tècnica d'ajust fi de models que introdueix matrius addicionals de rang baix en les capes del model preentrenat, mantenint congelats els pesos originals. D'aquesta manera, només cal entrenar un petit nombre de paràmetres addicionals per adaptar el model a tasques específiques, reduint dràsticament la complexitat i els recursos d'entrenament.

- **Quantització** és una tècnica de compressió de models que consisteix a convertir els pesos i els altres paràmetres d'un model (habitualment en format de punt flotant d'alta precisió) a representacions de menor precisió (per exemple, de 8 bits). L'objectiu és reduir la mida del model i els costos computacionals durant la inferència, tot i que pot suposar una lleugera pèrdua de precisió.

- **Prompt** és l'entrada (normalment un fragment de text o instruccions) que s'envia a un model de llenguatge perquè generi una resposta determinada. El prompt pot contenir preguntes, indicacions de tasca o context addicional, i un disseny adequat d'aquest text influeix directament en la qualitat i la rellevància de la sortida del model.

- **Token** és la unitat elemental de text amb què treballa un model de llenguatge. La tokenització és el procés de convertir un text en una seqüència de parts més petites (tokens), que poden ser paraules, subparaules o caràcters. El model processa aquestes unitats en lloc de paraules perquè això permet reduir-li al model l'espai de sortida (la mida del diccionari de "paraules") i, per tant, facilitar-li l'aprenentatge.

- **Dataset** (***conjunt de dades***) és una col·lecció organitzada d'exemples (o instàncies) que s'utilitza per entrenar o avaluar un model d'aprenentatge automàtic. Sovint s'emmagatzema en fitxers CSV en cas de tractar-se de dades tabulars, però podrien ser imatges o altres formats. Les dades han de contenir exemples rellevants per a la tasca d'aprenentatge. Disposar d'un dataset ampli, divers i representatiu és essencial per la capacitat de generalització del model més enllà de les dades d'entrenament.

Estructura de la memòria

El treball s'estructura en les següents seccions: Secció 2 explica la planificació inicial i l'executada; Secció 3 exposa els objectius plantejats; Secció 4 introdueix el marc teòric necessari per entendre la metodologia; Secció 5 defineix el problema i explica la implementació; Secció 6 mostra diferents casuístiques d'experimentació amb els models. Finalment, Secció 8 detalla les conclusions extretes.

2 Planificació

Aquest Treball de Fi de Grau s'estructura en diverses fases consecutives, cadascuna orientada a cobrir un objectiu específic dins de l'estudi. En primer lloc, es duu a terme una fase d'investigació preliminar destinada a identificar i revisar la bibliografia clau sobre tècniques d'intel·ligència artificial aplicades a la meteorologia, el funcionament de xarxes neuronals i el concepte de Low-Rank Adaptation (LoRA). A partir d'aquesta recerca teòrica, s'estableix l'enquadrament conceptual que sustenta la resta del projecte.

Un cop definit el marc teòric, la següent etapa consisteix en la preparació de les dades proporcionades per Troposferica. En aquesta fase es duu a terme la neteja del conjunt de dades, l'elecció de les variables rellevants i l'escriptura dels scripts de preprocessament necessaris per assegurar la coherència i la qualitat del dataset. Aquest procés inclou també la verificació de la integritat de la informació meteorològica i la seva transformació al format adequat per als prompts que s'utilitzen en la generació de textos.

Simultàniament, es configura l'entorn de desenvolupament, que implica la instal·lació de les llibreries requerides (transformers, PEFT, datasets, etc.) i la comprovació del funcionament correcte de la quantització a 8 bits per al model Qwen2-1.5B. En aquesta fase es duen a terme proves inicials amb el model preentrenat per assegurar la compatibilitat amb els recursos disponibles (memòria GPU, capacitat de càlcul, dependències de programari).

A continuació, es duen a terme els experiments de fine-tuning amb LoRA. Aquesta fase es divideix en subetapes: primer, l'entrenament amb diverses combinacions d'hiperparàmetres (r, α) utilitzant el Raw prompt per identificar la configuració òptima; després, l'execució de proves similars amb Instruction prompt; i finalment, l'entrenament prolongat per a les parelles seleccionades, amb l'objectiu de valorar l'efecte de la durada de l'entrenament sobre la qualitat de la generació.

Un cop completats els experiments, es fa l'anàlisi de resultats. Aquesta etapa inclou l'agregació de les mètriques automàtiques (ROUGE, BERTScore, BLEU, etc.), la comparació entre diferents escenaris i la identificació de les combinacions que proporcionen el millor equilibri entre precisió i cost computacional. També es valora la variabilitat introduïda pels *prompts* i es reflexiona sobre possibles ajustaments futurs.

3 Objectius

Aquest Treball de Fi de Grau persegueix diversos objectius generals i específics en l'àmbit de la predicció meteorològica mitjançant models d'Intel·ligència Artificial. En primer lloc, es busca demostrar la viabilitat de l'enfocament basat en LoRA per a l'afinament eficient de models de llenguatge massius, concretament Qwen2-1.5B en la seva versió en espanyol. El propòsit és avaluar si és possible, amb recursos limitats de GPU (8 GB), obtenir prediccions meteorològiques descriptives amb un grau de coherència comparable al de models entrenats amb dades massives (tot i posteriorment tenir accés a un servidor amb GPUs més potents).

Com a objectiu secundari, es pretén establir una metodologia robusta per a la recopilació, neteja i preprocessament de les dades històriques proporcionades per Troposferica, valorant quines variables -temperatura, humitat, precipitacions, velocitat i direcció del vent, cota de neu, símbols meteorològics...- són realment determinants a l'hora de generar un butlletí meteorològic comprensible i útil per als usuaris finals. Això inclou definir un pipeline clar que cobreixi des de la càrrega del conjunt CSV fins a la inserció de les variables en el format de *prompt* i la posterior generació de text.

Un tercer objectiu consisteix a mesurar quantitativament la qualitat de les prediccions textuals obtingudes. Per això, es fixen metes concretes per als indicadors automàtics: aconseguir valors de BERTScore superiors a 0.70, ROUGE-1 superiors a 0.30 i BLEU superiors a 0.01 en les condicions òptimes de fine-tuning. A més, es vol valorar l'efecte de la durada d'entrenament (per exemple, comparant 10 èpoques amb 100) i de la variació dels hiperparàmetres LoRA (r, α) sobre la memòria de GPU i el temps global d'entrenament, amb l'objectiu de trobar un equilibri entre precisió i cost computacional.

Finalment, des d'una perspectiva aplicada, el projecte aspira a produir una eina capaç de generar automàticament butlletins meteorològics amb un llenguatge natural proper a l'estil dels meteoròlegs professionals. Els resultats hauran de ser prou fluïts i informatius com per ser utilitzats en un futur en entorns reals (web, aplicacions mòbils, butlletins de ràdio) i oferir recomanacions pràctiques a l'usuari final.

4 Marc Teòric

4.1 Tècniques clàssiques de predicció meteorològica

La predicció meteorològica tradicional es basa, principalment, en la Predicció Numèrica del Temps [2] (també dita, per les seves sigles en anglès, NWP, de Numeric Weather Prediction), que utilitza models físics de l'atmosfera resolts per supercomputadors. Aquests models físics consisteixen en sistemes d'equacions resultants del coneixement expert sobre camps com la dinàmica de fluids, la termodinàmica i altres processos atmosfèrics, que proporcionen una simulació del futur comportament de l'atmosfera a partir d'un estat inicial.

Els models numèrics són interpretables físicament, ja que les seves prediccions respecten lleis conegudes, però pateixen certes limitacions, derivades d'aquesta incertesa que tenim de l'estat inicial, la comprensió incompleta de certs processos i els recursos computacionals.

Des dels anys cinquanta, amb l'adveniment d'ordinadors, la NWP ha esdevingut la base de les prediccions meteorològiques operatives [8], però, tot i els avenços continus com ara l'increment de resolució de la graella de càlcul o la capacitat d'afegir processos físics més complets, aquestes millores són costoses, tant en desenvolupament com en temps de càlcul. Fins i tot avui, amb l'enorme quantitat de recursos de la qual es disposa, els models numèrics globals més potents (com ara ICON, GFS o MeteoBlue), tenen habilitat predictiva fiable fins a uns pocs dies vista (el valor normal, entre 3-5 dies, i com a molt, fins a 10) a causa del caòtic comportament de l'atmosfera.

Per a poder gestionar aquesta incertesa i caràcter volàtil, des dels anys noranta s'utilitzen les prediccions per conjunts [2] (també conegudes com a *ensembles*).

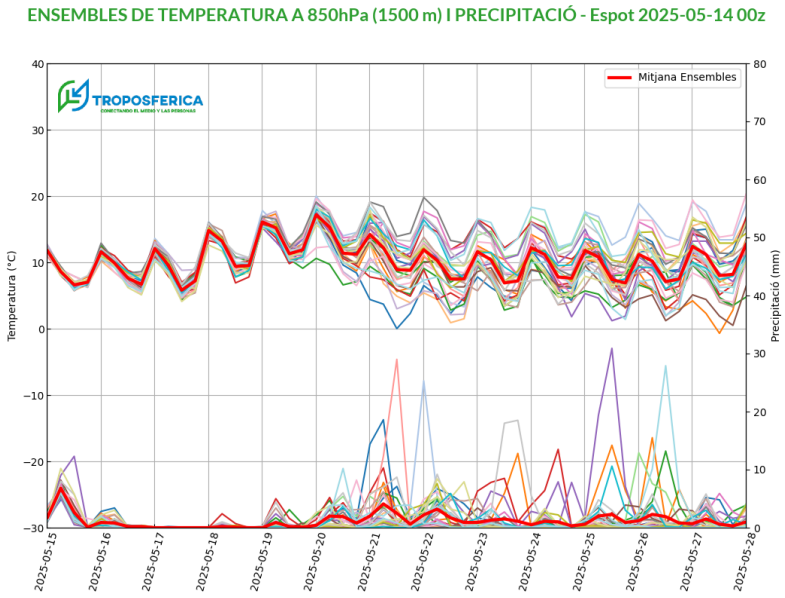

Figura 1: Exemple d'ensemble realitzat per Troposferica.

Com es pot observar a la Figura 1, per crear un ensemble es corren múltiples simulacions amb petites variacions en condicions inicials i fins i tot diferents models per tal d'estimar la confiança de la previsió. A més, s'han desenvolupat tècniques estadístiques clàssiques com el Model Output Statistics (MOS) per corregir biaixos locals dels models físics: el MOS aplica relacions estadístiques (p. ex., regressions) sobre la sortida del model numèric i dades observades per millorar la precisió local de la previsió.

Aquestes tècniques clàssiques es recolzen en models deterministes basats en lleis físiques conegudes, proporcionant coherència física i interpretabilitat, però requereixen supercomputadors i encara presenten errors significatius a causa de la resolució finita i a l'alta sensibilitat a condicions inicials. Aquest paradigma clàssic, malgrat l'extens èxit, imposa un límit pràctic clar sobretot a l'abast temporal i espacial de predicció, fet que motiva l'exploració de nous mètodes complementaris com ara l'aprenentatge automàtic.

4.2 MeteoBlue i com utilitza la IA

MeteoBlue és un servei suís de predicció meteorològica [13], i és interessant en aquest context, ja que fa ús de tècniques d'intel·ligència artificial per tal de millorar l'exactitud de les seves prediccions. En particular, MeteoBlue ha desenvolupat un sistema anomenat "MeteoBlue Learning MultiModel (o mLM)", el qual és un mètode de postprocessat que s'afavoreix de l'entrenament d'un model de Machine Learning sobre dades d'observació i més d'un model numèric per combinar-los de forma òptima. Aquest enfocament multimodal agafa la sortida d'uns 15 models de predicció numèrica tradicionals diferents i aprèn quin model o combinació ponderada és més fiable per a cada situació local, generant així un pronòstic calibrat a escala local.

Figura 2: MeteoBlue, l'empresa suïsa.

L'empresa suïssa utilitza aquesta tècnica des de 2018 [13], i la constant millora dels sistemes informàtics ha permès obtenir cada vegada distribucions de probabilitat més i més precises per a cada hora i lloc, reduint els errors i augmentant, segons l'empresa, en més d'un 5% respecte versions anteriors, i documenten que actualment és una de les precisions més altes de les quals se'n tenen constància.

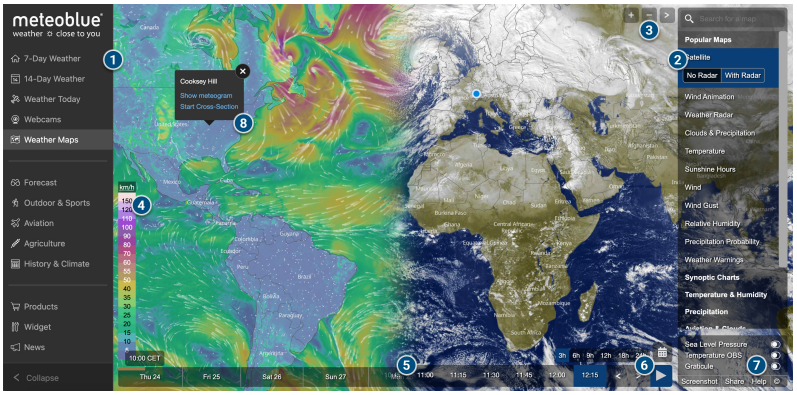

Figura 3: Exemple de mapa emprant els models numèrics interpretats.

A banda d'aquest mLM, MeteoBlue també explora un Artificial Intelligence Forecasting System (AIFS) [13], actualment en fase beta, que integra tècniques d'aprenentatge profund dins el procés de predicció. L'AIFS inclou un model determinista i un model probabilístic (ensemble) generats amb IA, executats diverses vegades al dia en col·laboració amb el Centre Europeu (ECMWF). Això indica que MeteoBlue combina fonts i enfocaments diversos: models físics convencionals, IA per a ajustos locals, i nous models meteorològics basats totalment en aprenentatge automàtic.

4.2.1 Com s'integra tot plegat per oferir prediccions precises i comunicables?

Gràcies a la IA, MeteoBlue pot refinar les seves previsions reduint errors sistemàtics i quantificant millor la incertesa. El mLM, per exemple, genera una distribució de probabilitat per a cada variable (temperatura, vent, humitat, raigs UV, etc.) cada hora, la qual cosa permet comunicar a qui utilitza el servei no només un valor pronosticat sinó també el rang de confiança. A més, en combinar múltiples models, es beneficia de la força de cadascun en diferents situacions (cada model actua diferent degut a hiperparàmetres com la resolució, per exemple), produint un pronòstic més robust i adaptat a zones concretes (fins i tot en terreny muntanyós complex, on es reporta que el sistema de MeteoBlue té una especial precisió). Aquesta precisió local millorada es pot traduir en comunicacions més útils per a l'usuari final, ja que les prediccions textuals poden incloure detalls fiables sobre condicions específiques d'una vall o d'un cim.

4.3 Machine Learning en meteorologia

En les darreres dècades, tant Aprenentatge automàtic com Aprenentatge profund se n'han anat introduint progressivament dins el món de la meteorologia per a complementar (i millorar) els enfocaments físics tradicionals.

Al principi de tot, tècniques d'aprenentatge estadístic es van emprar en tasques com la correcció de biaixos (el MOS abans mencionat) o l'adaptació d'escala global a local.

Actualment, s'apliquen xarxes neuronals profundes per millorar diversos aspectes de les prediccions. Un exemple són les xarxes convolucionals especialitzades en visió que es fan servir per a nowcasting (previsions a molt curt termini) de precipitacions a curt termini a partir d'imatges d'un radar. Models desenvolupats per la mateixa Google, com MetNet [17], aconsegueixen pronosticar la pluja horària fins a dotze hores amb gran detall, fent ús només de dades radar i satèl·lit, superant així als mètodes tradicionals.

Alhora, també s'ha fet ús de Transformers i xarxes recurrents [19], emprats per assimilar sèries temporals meteorològiques i poder predit l'evolució de variables com temperatura o vent en llocs concrets.

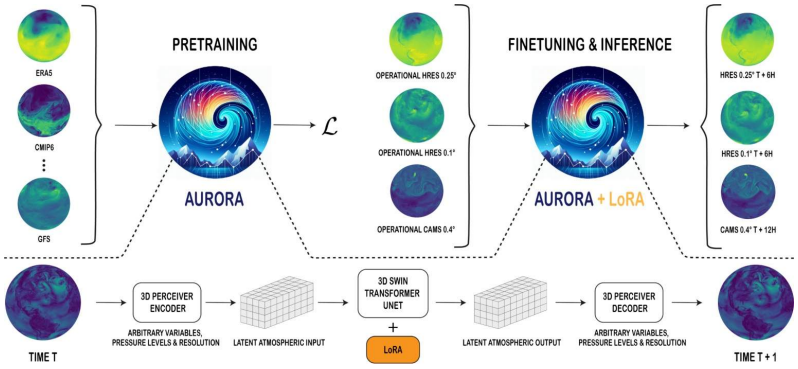

Figura 4: Aurora, la revolucionària IA de Microsoft que promet predir el temps sense error, conté un transformer.

4.4 Models basats íntegrament en IA

Fins ara hem vist integracions d'IA com a complement més que com a eina, però això va canviar fa cosa d'un parell d'anys, quan Huawei i DeepMind van presentar els seus respectius models meteorològics basats íntegrament i exclusivament en IA, el Pangu-Weather i el GraphCast [3] [9], ambdós entrenats amb dècades de dades globals, que són capaços de pronosticar l'estat atmosfèric a escala planetària amb una qualitat comparable en certs aspectes als models físics de referència, com el ECMWF europeu.

Aquests nous models basats purament en IA han demostrat encerts molt respectables en certs contextos (vegeu Figura 5) [16]. Són capaços també d'anticipar esdeveniments d'alt impacte com ara tempestes o ciclons tropicals amb molts dies d'antelació.

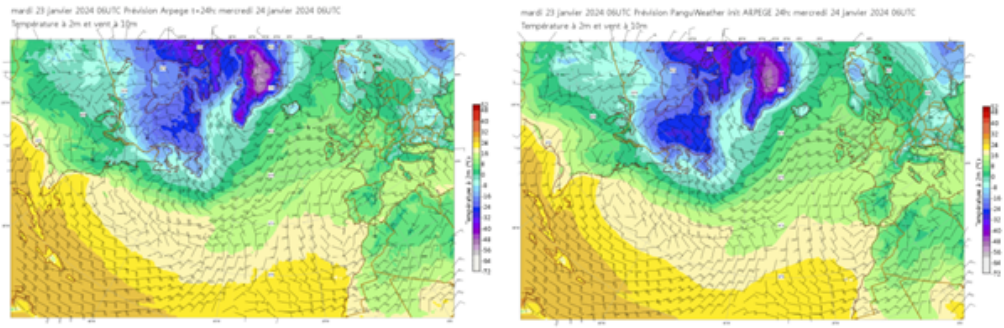

Figura 5: Predicció de temperatura (grad de color) i vent (barbulae), calculat pel model francès Arpège (model físic tradicional) (esquerra) i pel model IA Pangu-Weather (dreta) pel domini Europa-Atlàntic.[1]

El principal avantatge és que aquests models basats en IA poden generar una bona predicció en qüestió de minuts, molt més ràpid que els seus germans sense IA, que de vegades requereixen hores de càlcul en supercomputadors. Aquesta millora en eficiència obre la porta a actualitzacions més freqüents dels pronòstics i a optimitzar costos computacionals.

És cert que cal matissar que els models de ML en meteorologia encara tenen mancances importants, sovint actuen com caixa negra i poden violar principis físics (p. ex. generant camps atmosfèrics inconsistents) si no s'imposen restriccions; de fet, els models Pangu o GraphCast proporcionen de moment una representació parcial de l'atmosfera, i ja s'han identificat debilitats en certs fenòmens concrets.

Actualment, es treballa en la interpretabilitat d'aquests models: com també als models físics se'ls hi pot fer un diagnòstic d'error, en aquests és interessant entendre quin component de la xarxa neuronal causa l'error en la predicció.

Un altre ús a banda de l'esmentat de ML en meteorologia és la generació de prediccions probabilístiques: hi ha xarxes neuronals que han estat entrenades per detectar patrons perillosos en sortides de models numèrics (p. ex. tempestes significatives en mapes de radar) amb bons resultats, però això és un tema que no concerneix en aquest treball.

Finalment, cal recalcar que, com he dit abans, aquests models ML basats en IA avui dia no són més que un complement als models físics, sent la tendència actual buscar un enfocament on ambdós tipus de sistemes es correlacionin i funcionin de manera conjunta, en sistemes híbrids.

[1] Font: `https://www.encyclopedie-environnement.org/en/artificial-intelligence-and-weather-forecasting/`; Meteofrance

4.5 Aprenentatge automàtic i Xarxes Neuronals

Deixant de banda la meteorologia per un moment, per entendre aquest treball correctament, s'ha d'entendre què és l'Aprenentatge Automàtic i les Xarxes Neuronals.

L'Aprenentatge Automàtic (AA d'ara endavant) és una branca de la intel·ligència artificial que construeix models matemàtics a partir de conjunts de dades. Aquests algoritmes aprenen patrons en les dades d'entrada per realitzar prediccions o decisions.

Per altra banda, les Xarxes Neuronals Artificials (XNA d'ara endavant) són un tipus de model d'AA inspirat en el cervell humà. Estan formades per capes de "neurones"artificials interconnectades, on cada neurona realitza un càlcul simple i envia el resultat a la següent capa. L'arquitectura típica inclou una capa d'entrada, una o més capes ocultes i una capa de sortida. El comportament global de la xarxa és determinat per les connexions entre neurones i els pesos associats. En essència, la XNA aprèn a aproximar funcions complexes combinant les aportacions de moltes unitats simples.

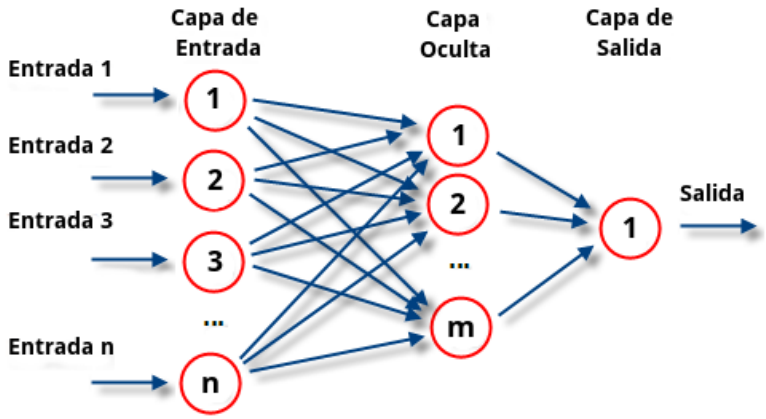

Figura 6: Esquema del funcionament d'una XNA.

I, com s'entrena una XNA? Per ajustar els pesos d'una xarxa i fer que el model aprengui del conjunt de dades, s'utilitza l'algorisme del descens del gradient en combinació amb la backpropagation. Concretament, s'estableix una funció de pèrdua (loss) i es van actualitzant iterativament els pesos en la direcció que disminueix més ràpidament aquesta funció. La retropropagació calcula eficientment el gradient del cost respecte a cada pes propagant l'error des de la sortida fins a les capes del principi. Cada iteració de l'entrenament, també dit època (o epoch) reajusta aquests pesos una mica més per acostar-se al mínim cost possible. Per evitar overfitting (un sobreajustament de les dades d'entrenament) sovint s'aplica early stopping (es monitoritza el rendiment sobre el conjunt de validació i s'interromp l'entrenament quan aquest deixa de millorar en determinades etapes).

4.6 Models Transformer i Llarga Seqüència

Els Transformers són una arquitectura de xarxa neuronal dissenyada per tractar seqüències d'entrada (com textos) amb atenció i paral·lelisme. Proposats per Vaswani et al. (2017) [19], aquests models substitueixen les xarxes recurrents per mecanismes d'autoatenció. Concretament, un Transformer processa tots els tokens de la seqüència en paral·lel, calculant pesos d'atenció entre totes les parelles de posicions. Aquesta estructura és capaç de capturar dependències a llarg termini i accelera l'entrenament respecte a RNN tradicionals. Com diu l'article introductori, el Transformer és el primer model seqüència-a-seqüència que "es basa íntegrament en l'atenció pròpia"sense usar RNN o convolucions. Un Transformer original consta d'un encoder (pila de capes de codificador) i un decoder (pila de capes de descodificador), cadascuna amb subcapes d'atenció i xarxes neuronals feedforward. Aquesta arquitectura maneja fàcilment la rellevància relativa de tokens llunyans i ofereix un rendiment excel·lent en tasques seq2seq.

Hi ha diverses variants segons com s'organitzen els encoders i decoders. Els Transformers encoder-decoder (seq2seq) són l'estructura original (com en traducció automàtica). Hi ha també models de només encoder, que consisteixen en una pila d'encoders sense descodificador, com per exemple BERT [20] (apropiat per a tasques de comprensió o classificació de text). D'altra banda, els models de només decoder (decoder-only) són piles de blocs de descodificador causal i s'usen principalment per a generació de text autoregressiva. En aquesta categoria hi entra la sèrie GPT (Generative Pretrained Transformer) d'OpenAI, que des del 2018 ha esdevingut l'estàndard en generació de llenguatge natural. Per exemple, GPT-3 i ChatGPT són models de només decoder entrenats per predir el següent mot en una seqüència.

Els Large Language Models (LLMs) són Transformers de molt gran escala (milions o milers de milions de paràmetres) preentrenats amb enormes corpus de text. Gràcies a aquesta gran capacitat i dades, són capaços de generar llenguatge molt natural i versàtil. El funcionament pràctic d'un LLM sovint parteix d'un prompt: un text d'entrada en llenguatge natural que descriu la tasca o pregunta que li fem al model. Un prompt pot ser una pregunta, una instrucció o un fragment de context, i el model continua generant text rellevant. Tot i el poder dels LLMs, un desafiament és adaptar-los amb dades limitades (com justament m'ha passat a mi). A causa de la seva mida i complexitat, si disposem de poques mostres d'entrenament hi ha risc de sobreajustament i falta de generalització. Això exigeix tècniques de regularització i adaptació eficients (vegeu apartat següent).

4.7 Models generatius de llenguatge

En el context del llenguatge natural, els models generatius de gran escala han demostrat una capacitat sorprenent per crear text coherent. A continuació es comparen els dos models explorats en aquest treball, GPT-2 i Qwen2, en termes d'arquitectura, capacitat, idiomes i eficiència.

GPT-2 (OpenAI, 2019) [14]: És un model Transformer de tipus decoder-only, entrenat amb la tasca de predir la següent paraula en seqüències de text extensives. La seva versió completa compta amb 1.5 mil milions de paràmetres i va ser entrenada amb uns 40 GB de text (aprox. 810^9 paraules) extrets de la web. Va suposar un salt qualitatiu en la generació de text: pot produir paràgrafs sencers que segueixen un tema donat de forma consistent i gramaticalment correcta. No obstant això, presenta limitacions en la coherència a llarg termini, quan el text generat s'allarga diversos paràgrafs, tendeix a desviar-se del tema o repetir-se, reflectint la manca de comprensió profunda del contingut. Arquitectònicament, GPT-2 no incorpora coneixement explícit del món ni ajust fi (fine-tuning) per seguiment d'instruccions - és a dir, generava text "lliurement" a partir d'una entrada, sense mecanismes per assegurar veracitat o estil controlat. En termes computacionals, el model de 1.5B de paràmetres ocupa uns 5 GB de memòria i era poc pràctic d'executar en entorns limitats l'any 2019.

Qwen2 (Alibaba Cloud, 2023) [1]: Qwen és la família de models de llenguatge alliberada per Alibaba amb objectiu de ser una alternativa oberta i multiidioma als GPT d'OpenAI. El nom Qwen significa *Tongyi Qianwen* (en xinès), i inclou diverses mides: des de versions lleugeres de 1.8B de paràmetres fins a models grans de 7B, 14B i 72B, a més de variants orientades a conversa (Qwen-Chat) i multimodals. En concret, Qwen2-1.5B, l'emprat en el projecte, fa referència a un model base d'aproximadament 1.5 mil milions de paràmetres (derivat de Qwen-1.8B) que ha estat ajustat per a l'idioma espanyol. El model Qwen adopta també l'arquitectura Transformer decoder i es va entrenar amb aproximadament 2-3 bilions de tokens de text multilingüe. Segons Alibaba, Qwen (la versió base) és intrínsecament multilingüe, amb especial domini de xinès i anglès, però també capacitat en altres idiomes com el castellà, francès i japonès gràcies a la diversitat del corpus d'entrenament. De fet, les versions recents Qwen2.5 amplien aquesta capacitat suportant més de 29 idiomes (inclosos el castellà i català) i millorant habilitats de seguiment d'instruccions.

Comparat amb GPT-2 i GPT-3, Qwen2 presenta diversos avantatges en contextos específics com el d'aquest projecte:

- ***Capacitat i eficiència***: Amb 1.5B paràmetres, Qwen2 és molt més lleuger que GPT-3 (175B) i similar en escala a GPT-2, la qual cosa permet desplegar-lo i entrenar-lo en maquinari modest (per exemple, és plausible ajustar Qwen-1.8B en una sola GPU de 12 GB mitjançant tècniques d'optimització com LoRA o, fins i tot, en l'ordinador emprat, amb una GPU de 8 GB). Alhora, en estar entrenat amb tècniques modernes i un corpus molt gran, Qwen2-1.5B pot superar GPT-2 en qualitat de text i comprensió, malgrat tenir grandàries

similars, ja que incorpora millores d'arquitectura i entrenament de la darrera generació (per exemple, una finestra de context més gran, i un tokenitzador optimitzat per múltiples llengües).

- ***Idioma***: Qwen2-1.5B Spanish està adaptat específicament al castellà, cosa que li atorga qualitats per generar text meteorològic en aquesta llengua amb fluïdesa i vocabulari adequat, i el fet de ser un model obert permet entrenar-lo o ajustar-lo addicionalment amb textos meteorològics en català, cosa factible donada la proximitat lingüística i la capacitat multilingüe ja present en la base de Qwen.

- ***Accés i flexibilitat***: A diferència de GPT-3, Qwen2 és de codi obert i pot executar-se localment. Això permet personalitzar el model (per exemple, incorporar terminologia meteorològica o estil narratiu específic, en aquest cas de les prediccions reals de Troposferica) sense dependre d'una API externa. També elimina restriccions de privadesa i el cost per crida que tindria l'ús d'un model tancat.

- ***Arquitectura millorada***: Qwen2 ha estat concebut com un bon punt de partida per a ajustaments (Alibaba destaca que els seus models base estan prou "nets" per fer-hi fine-tuning amb facilitat). No inclou funcionalitats conversacionals per defecte (a diferència de Qwen-Chat), però això és ideal en un entorn de generació de text controlada: ens interessa un model que generi text descriptiu directament a partir de dades, sense desviacions de xat.

- ***Rendiment***: Segons proves realitzades per la comunitat, versions de Qwen d'escala 7B-14B rivalitzen amb models coneguts com a Llama 2 en diversos benchmarks, i en comprensió de xinès i anglès són capdavanteres entre models oberts a finals de 2023. La variant 1.5B, tot i ser més petita, es beneficia del mateix entrenament. Per tant, és raonable esperar que Qwen2-1.5B Spanish generi textos meteorològics amb correcció gramatical i adequació terminològica, sempre que el domini li sigui familiar a través de l'ajust amb dades històriques meteorològiques.

Aquesta comparativa justifica l'elecció de Qwen2-1.5B en aquest projecte: és prou gran per capturar i expressar informació complexa, prou petit per ser ajustat eficientment en la infraestructura pròpia, i dissenyat per ser polit (mitjançant LoRA) en l'idioma i estil requerits.

4.8 Adaptació Eficaç: LoRA (Low-Rank Adaptation)

L'entrenament o fine-tuning complet d'un LLM gran requereix enormes recursos computacionals. Hu et al. (2021) [6] proposen LoRA per tal d'afrontar aquest problema. La idea clau és congelar tots els pesos del model base i, en lloc de modificar-los directament, inserir en cada capa del Transformer unes petites matrius de rang reduït (low-rank) que sí que es poden entrenar. D'aquesta manera, el nombre de paràmetres a optimitzar s'abreuja de manera dràstica. Concretament, per a cada matriu de pesos W original, LoRA afegeix dues matrius petites A i B (amb un rang menor) de manera que el canvi efectiu es modela com W' = W + A·B. Aquestes matrius A i B s'inicialitzen aleatòriament i s'entrenen, mentre que W roman fix. Els experiments demostren que amb LoRA es pot reduir els paràmetres entrenables en un factor de milers mantenint rendiment comparable. Per exemple, s'indica una reducció de 10.000 vegades en els paràmetres entrenables i un estalvi de memòria de 3x amb qualitat similar o millor que l'ajust complet.

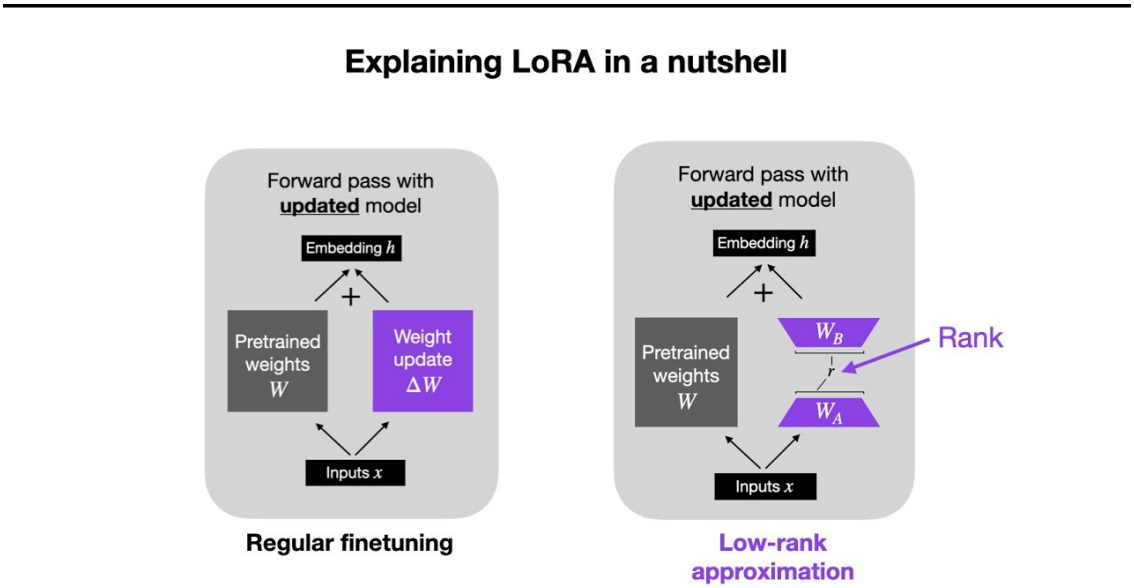

Figura 7: Podem observar les diferències entre l'entrenament normal i amb LoRA

En la pràctica, LoRA s'ha convertit en una tècnica àmpliament utilitzada per a afinar eficientment LLMs. Aplicar LoRA significa triar un "*rank*"adequat (la mida de les matrius A i B), i després entrenar només aquests paràmetres addicionals. Aquesta adaptació de baixa dimensió facilita ajustar models grans amb conjunts de dades petits i memòria limitada. Hu et al. [6] mostren també que, a diferència d'altres enfocaments similars ("adapters"), LoRA no afegeix latència extra a la inferència i permet major paral·lelisme.

4.9 Qwen2-LoRA: avantatges i comparativa

En aquesta secció es detallen els avantatges específics d'utilitzar el model Qwen2-1.5B ajustat amb la tècnica LoRA (Low-Rank Adaptation), i es compara aquest enfocament amb altres alternatives d'ajustament de models grans. L'objectiu és destacar per què aquesta elecció és adequada especialment en contexts de recursos computacionals limitats i dades específiques.

Aplicat a Qwen2-1.5B, l'ús de LoRA presenta diversos avantatges:

- ***Requisits de maquinari baixos***: Qwen2 ja és un model relativament compacte; amb LoRA, només s'han d'entrenar pocs milions de paràmetres. Això significa que podem dur a terme l'ajust utilitzant, per exemple, una GPU de gamma mitjana (8-12 GB VRAM) o fins i tot fer-ho per parts en CPU, cosa essencial en un entorn de TFG on sovint no es disposa de supercomputadors. En comparació, un ajust complet d'un model gran (fins i tot de 7B-13B paràmetres) requeriria multiples GPUs d'alta gamma o molt de temps.

- ***Rapidesa i iteració***: En reduir-se els paràmetres a entrenar, el cicle de train-eval és més ràpid. Això facilita iterar, provar diferents hiperparàmetres o incorporar noves dades incrementalment sense un cost temporal prohibitiu. Així, el desenvolupament esdevé més àgil.

- ***Manteniment del coneixement base***: Qwen2 prové ja entrenat amb un vast corpus multilingüe i coneixement general. LoRA no altera aquest coneixement general, sinó que l'afegeix o modula lleugerament amb les noves matrius. Com a resultat, el model afinat amb LoRA conserva la fluïdesa lingüística i la coherència del model base, però ajusta la sortida al domini meteorològic. Això és preferible a entrenar un model des de zero amb les nostres dades (on costaria generalitzar), i fins i tot a fine-tuning complet, on hi ha risc de sobreajustar i malmetre coneixements previs (efecte conegut com a catastrophic forgetting [11]). LoRA, actuant com una capa supletòria, evita en gran mesura aquest problema.

- ***Comparació amb altres mètodes***: Alternatives a LoRA inclouen Full Fine-Tuning (ajustar tots els pesos) o altres adaptadors (p. ex. AdapterHub) i entrenament per prompt (prompt tuning). Respecte a l'ajust complet, com ja s'ha dit, LoRA és molt més eficient i econòmic en memòria, amb resultats equiparables. Enfront d'adaptadors tradicionals, LoRA té l'avantatge de no introduir latència (els adaptadors clàssics afegeixen capes extra en inferència, LoRA no). I comparat amb el prompt tuning, LoRA sol necessitar menys dades per assolir bones prestacions, ja que realment modifica lleugerament la representació interna del model, en lloc de dependre només d'entrades enginyades.

Amb tot, la decisió d'usar Qwen2-LoRA es reafirma considerant alternatives. Un model com GPT-3, per bé que potent, no és practicable per a un ajust local i a més no podríem incorporar-lo al nostre sistema sense costs elevats i dependència

externa. Un model més petit com GPT-2, tot i que *open source* i de mida semblant a Qwen2, no disposa de suport multilingüe ni del nivell de competència que Qwen ha demostrat en llengües com el castellà. A més, GPT-2 no ha estat entrenat amb tècniques modernes d'alineament, la qual cosa significa que generaria text menys controlat. Qwen2, en canvi, va ser concebut el 2023 integrant millors pràctiques d'entrenament i és un model amb capacitat de context ampli - característiques valuoses si volem, per exemple, condicionar la generació amb molts detalls d'entrada.

Finalment, val la pena mencionar la possibilitat d'usar QLoRA (Quantized LoRA), un refinament que combina LoRA amb quantització de pesos a baixa precisió [4]. QLoRA permet entrenar models de desenes de milers de milions de paràmetres en una sola GPU reduint els pesos a 4 bits durant l'afinament, amb resultats excel·lents. En aquest cas, QLoRA no ha sigut necessari donada la mida moderada de Qwen2-1.5B, però és una tècnica disponible que reflecteix la versatilitat de l'enfocament LoRA en entorns de low-resource.

4.10 Mètriques d'avaluació

En tasques on cal garantir que el model reprodueixi exactament la terminologia establerta (per exemple, en traducció automàtica de documents legals o manuals tècnics) el BLEU és indiscutiblement valuós perquè mesura la superposició literal de n-grammes entre la traducció generada i la referència original (Papineni, Roukos, Ward, and Zhu, 2002) [15]. Si cada terme ha de coincidir al peu de la lletra, aquesta exigència de concordança exacta assegura que no hi hagi variacions lèxiques que puguin alterar el sentit precís del contingut.

En canvi, quan el repte és condensar un document llarg en un resum més breu (com en informes científics o notícies), el ROUGE esdevé la mètrica de referència, ja que avalua la proporció de conceptes i subseqüències clau que apareixen en el resum respecte a l'original, incloent-hi la mesura de la subseqüència comuna més llarga (Lin, 2004) [10]. Això permet verificar que el resum abasti totes les idees essencials, fins i tot si canvia l'estructura de les frases, sense perdre's punts crucials del missatge.

En entorns on la varietat lèxica i la reformulació creativa són acceptables (com ara la generació de subtítols d'imatges, els xats conversacionals o, com en aquest cas, generació de textos no-literaris on pot haver-hi creativitat), BERTScore és especialment recomanable perquè avalua la qualitat del text en l'àmbit semàntic, no pas només lèxic. Mentre que mètriques com BLEU o ROUGE penalitzen amb duresa qualsevol canvi en l'ordre de les paraules o l'ús de sinònims, BERTScore empra embeddings contextuals de BERT per mesurar la proximitat semàntica entre cada paraula generada i la de referència, de manera que reconeix equivalències com "ruixats ocasionals" i "pluges febles" com a coincidents en significat (Zhang et al., 2019) [20].

A més d'aquestes mètriques principals, podríem considerar METEOR, CHRF++, etc., però BLEU, ROUGE i BERTScore ja proporcionen una visió variada: BLEU/-ROUGE enfoquen coincidència superficial i cobertura, BERTScore el significat.

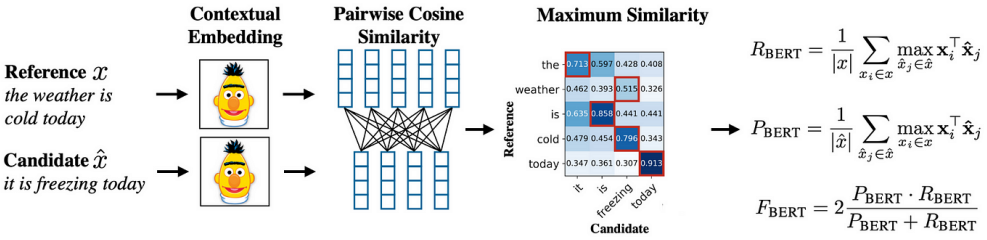

Figura 8: Representació visual de BERTscore, amb una imatge d'internet que s'ajusta al tema de meteorologia.

També és possible emprar mètriques específiques de coherència o gramàtica (per exemple, fer servir un model de llengua per calcular la perplexitat del text generat com a mesura de fluïdesa). No obstant això, en un TFG més pràctic com aquest, es contrastarà l'acompliment textual principalment amb les mètriques esmentades i faré una avaluació humana qualitativa complementària: meteoròlegs qualificats llegiran mostres de prediccions generades i intentaran esbrinar si es tracta d'un text generat pel model o escrit per ells mateixos.

És important definir quin serà el criteri d'èxit: tret que BERTScore és la mètrica més important, obtenir un valor major a **0.7 (70% de precisió)** seria considerat com a satisfactori, ja que comptant amb un conjunt de dades reduït, és molt bona puntuació.

5 Metodologia

5.1 Definició formal del problema

En aquest treball es planteja la generació automàtica de resums meteorològics diaris a partir de dades estructurades.

Formalment, l'objectiu és transformar un conjunt de variables meteorològiques estructurades (temperatura, precipitació, vent, etc., mesurades a hores específiques al llarg del dia) en un text lliure que resumeixi la situació meteorològica del mateix dia. Aquest text ha d'imitar l'estil dels meteoròlegs professionals de l'empresa Troposferica, és a dir, presentar la informació del temps de manera coherent i amb la mateixa tonalitat i vocabulari que faria un expert humà, puix que ens han cedit les dades i els textos estan esbiaixats segons el mateix estil. En suma, es tracta d'un problema de data-to-text en l'àmbit meteorològic: donades dades numèriques i categòriques d'entrada, volem obtenir una descripció lingüística rica i correcta que representi la predicció diària del temps.

5.2 Dades d'entrada i sortida

El conjunt de dades utilitzat conté 67 columnes de variables meteorològiques per a cada dia. Abans d'explicar de quines es tracten, però, s'ha d'explicar com s'ha arribat a aquest conjunt de dades: en un principi, les dades es recopilen dia a dia per l'empresa, obtenint així un conjunt de directoris en format HISTORICS/any/mes/-dia, on cada dia conté els fitxers en format de text pla (.txt) per a cada variable. Un exemple més visual: el directori HISTORICS/2024/Gener/19-01-24/ conté, entre d'altres, el fitxer humitatMolina.txt, on es troben en format h1d00h=valor (essent h un identificador diferent per a cada variable [v per velocitat del vent, p per la precipitació...]) el valor de la humitat a La Molina, per a les 0 hores del primer dia de predicció (depenent del client se'n fan tres, cinc o fins a deu dies, però en aquest treball s'utilitza només el primer dia per evitar complicacions innecessàries).

Entre aquestes variables hi ha tant atributs numèrics (la temperatura o la humitat, per exemple) com atributs categòrics (símbols que representen l'estat del cel, per exemple). Als històrics hi ha més variables de les emprades en el conjunt de dades, tals com el risc de tempesta o els raigs UV, descartades per la baixa importància en la creació dels textos.

Així, doncs, com a dades d'entrada al model es consideren les següents 67 columnes, amb les pertinents justificacions:

- ***day, month, year***: separació de la data en tres columnes
- ***copsVent1, 3, 5 ... 23***: cops de vent, en km/h, important que el seguiment sigui més continu, ja que aquests cops són puntuals i poden fluctuar.
- ***cotaNeu1, 6, 11, 16, 21***: cota a la que comença a haver-hi neu, en m, sol ser semblant entre hores, per això menys registres.

- ***dirVent1, 5, 9, 13, 17, 21***: direcció en la qual bufa el vent, variable categòrica amb els 8 possibles valors de la rosa dels vents *(N, NE, E, SE, S, SW, W, NW)*, també amb pocs canvis entre hores.

- ***humitat1, 6, 11, 16, 21***: humitat relativa, en *%*, relativament invariable, per tant, pocs registres.

- ***precipitacio1, 3, 5 ... 23***: precipitació acumulada per hora (la suma de la precipitació acumulada per minut), en *mm*, bastant important doncs la precipitació diària és la suma de les horàries.

- ***simbols1, 4, 7 ... 22***: nomenclatura específica de l'empresa, indiquen l'estat del cel, pot ser, per exemple, *sol si el dia és solejat o variable si hi ha certs núvols voltant pel cel.*

- ***temperaturaCotes1, 4, 7 ... 22***: temperatura, en *graus Celsius*, per a cada cota, tot i que recopilant les dades només s'ha agafat la de la cota més baixa, la més important.

- ***vent1, 5, 9, 13, 17, 21***: velocitat mitjana del vent, en *km/h*, més invariable que els cops, per tant, menys registres.

- ***cliente***: *client* a qui va adreçat el text. És important, perquè per a cadascun se'ls hi parla de manera diferent, enfocant-se en diferents aspectes.

- ***texto***: és el *text real* donades les variables anteriors per a cada dia i client. Segons aquest, el text estarà en castellà o en català. Cal mencionar que, com el model emprat està entrenat en castellà, tots els textos que estaven en català s'han hagut de traduir amb una eina automatitzada al castellà, fent que aquests perdessin una mica de rigorositat i qualitat.

La sortida que es vol obtenir, doncs, és similar al que hi trobem a la columna *texto*, un resum meteorològic redactat per una persona experta. Aquests textos són paràgrafs en llenguatge natural, proper i tècnic, on es descriu el temps que ha fet o farà durant el dia, mencionant aspectes com la temperatura màxima o mínima, la presència de precipitacions, l'estat del cel (assolellat, ennuvolat, entre d'altres) i altres fenòmens destacables. En el procés d'entrenament, aquests textos serveixen de referència (etiqueta) perquè el model aprengui a generar una descripció similar a partir de les variables d'entrada corresponents. L'objectiu final és que, donat un nou conjunt de mesures meteorològiques d'un dia, el sistema generi automàticament un text equivalent al que escriuria un meteoròleg de Troposferica.

5.3 *Pipeline* de l'entrenament

En aquest apartat es descriu el *pipeline* implementat al fitxer *train_weather_qwen_lora.py*, mitjançant el qual es transformen dades meteorològiques d'entrada en textos descriptius de predicció del temps. El procés s'organitza en diverses etapes seqüencials, des de la preparació de les dades fins a la generació final del text pronosticat, tal com es detalla a continuació.

5.3.1 Preprocessament de les dades d'entrada

El *pipeline* s'inicia amb la lectura d'un fitxer *CSV* que conté els registres meteorològics d'entrada (66 columnes de variables) i el text de predicció associat a cada registre (columna objectiu). En primer lloc, es realitza un preprocessament de les dades per assegurar la seva qualitat i format consistent, tot i haver netejat coses a mà. Això inclou accions com el tractament de possibles valors nuls que s'hagin colat a la neteja manual. Un cop les dades estan netes, es divideixen en conjunt d'entrenament i de validació, els quals s'ha decidit que siguin el valor estàndard, 90% per entrenament i 10% per validació. Cal esmentar, però, que després es va agafar un nou conjunt de dades que mai veu el model entrenant, que es defineixen com a *test*, però d'això es parlarà més endavant.

5.3.2 Construcció dels *prompts* de text

Després de la preparació bàsica, per a cada registre meteorològic preprocessat es construeix un *prompt* textual que servirà d'entrada al model *LLM*. Aquest *prompt* integra les 66 variables meteorològiques d'una manera llegible: s'estructura en forma de llista de paràmetres que inclou cada variable i el seu valor. En aquest cas, després de realitzar una sèrie d'experiments que s'esmentaran més tard, el *prompt* construït és *"variable1=valor, variable2=valor ... TEXT => text real associat als valors. "<—endoftext—>""*, però, per exemple, es podria generar una frase que enumeri la temperatura, humitat, velocitat del vent, etc., seguides d'alguna indicació que a continuació ve el pronòstic. L'objectiu és proporcionar al model tota la informació rellevant del registre en format text natural. El text objectiu (*target*) - és a dir, la predicció meteorològica redactada - queda associat a aquest *prompt* com la sortida esperada. D'aquesta manera, el conjunt de dades d'entrenament es compon de parelles *prompt* $\rightarrow$ *text pronosticat*, entrenant el model a generar la descripció meteorològica correcta un cop llegides les dades d'entrada.

5.3.3 Tokenització i configuració del model pre-entrenat

Un cop definits els *prompts* i els textos objectiu, el següent pas és la *tokenització*. S'utilitza el *tokenitzador* predefinit del model Qwen2-1.5B-Spanish-1.0 per convertir tant els *prompts* com els textos objectiu en seqüències de *tokens* numèrics. La *tokenització* mapifica cada paraula o símbol pertinent a un identificador enter segons el vocabulari del model. És important emprar el mateix *tokenitzador* amb què el

model original fou entrenat, assegurant així que els *tokens* corresponen a *embeddings* vàlids per a Qwen2. Aquesta etapa produeix les seqüències de *tokens* d'entrada (*tokens* del *prompt*) i, en entrenar en mode causal, concatena també els *tokens* del text objectiu darrere dels del *prompt*, marcant on acaba l'entrada i comença la predicció esperada.

A continuació es carrega el model de llenguatge preentrenat Qwen2-1.5B-Spanish-1.0. Per fer viable l'entrenament amb recursos limitats, s'opta per utilitzar quantització en 8 bits mitjançant la llibreria *BitsAndBytes* [5]. Aquesta tècnica de quantització (LLM.int8()) redueix a la meitat (aprox.) l'ús de memòria i fa més accessible el procés d'afinament sense degradar significativament el rendiment del model. El model es carrega en memòria en format *int8*, mantenint els pesos preentrenats però ocupant menys espai que en precisió completa.

5.3.4 Entrenament del model amb LoRA (PEFT)

Sobre el model quantitzat es realitza el *fine-tuning* emprant una tècnica de *Parameter-Efficient Fine-Tuning* (*PEFT*) anomenada *LoRA* (*Low-Rank Adaptation*). En comptes de reentrenar tots els pesos del model massiu, com s'ha esmentat anteriorment, *LoRA* introdueix un nombre reduït de paràmetres addicionals en forma de matrius de baix rang als pesos de les capes d'atenció del transformador. D'aquesta manera només s'entrenen aquests paràmetres addicionals (mantenint la resta congelats), aconseguint reduir dràsticament el nombre de paràmetres que cal ajustar i, per tant, la memòria i temps de càlcul necessaris. En aquest cas, es configura *LoRA* amb un rang $r = 16$ i un factor d'escala $\alpha = 32$ (veure experiments). Aquest valor de r significa que les matrius afegides tenen rang 16 (són molt més petites que els pesos originals), i $\alpha = 32$ s'ha escollit seguint la recomanació habitual de posar α com el doble del rang. En conjunt, això ens deixa entrenables el 0.28% dels paràmetres, aproximadament uns 4 milions. Això implica que els canvis apresos s'escalen per $\alpha/r = 2$ en incorporar-se als pesos originals, segons la implementació estàndard de Microsoft per *LoRA*. L'elecció de $r = 16$ (amb $\alpha = 32$) representa un compromís entre capacitat d'aprenentatge i eficiència: introdueix prou paràmetres adaptatius per capturar les relacions entre les variables meteorològiques i el text de referència, alhora que només suposa una fracció mínima (pocs milions de paràmetres efectius) respecte a els 1.500 M del model original, sent així el conjunt que més esprem la GPU emprada.

Amb el model preparat (carregat en 8 bits i instrumentat amb els adaptadors *LoRA* injectats a les capes d'atenció), es procedeix a l'entrenament utilitzant el conjunt d'entrenament de *prompts*. Durant l'entrenament, el model aprèn a produir el text de sortida pronosticat després de cada *prompt* d'entrada. El procés d'optimització s'executa típicament amb un algorisme de descens del gradient i *batching* de dades, aprofitant la GPU. Només els pesos de *LoRA* són actualitzats en cada iteració, mentre que la resta de pesos del model romanen fixos. Gràcies a això, la quantitat de paràmetres actualitzables és molt reduïda, cosa que millora l'eficiència: LoRA permet afinar un model gran modificant només un 0.1% (aprox.) dels seus paràmetres, segons documentació oficial de Hugging Face (tot i tenir nosaltres un

0.28%) [7] [6]. Durant l'entrenament, es registra la pèrdua (*loss*) sobre cada *batch* i s'avalua el model sobre el conjunt de validació periòdicament per tal de monitorar el progrés.

5.3.5 Validació del model i callbacks de monitorització

El *pipeline* incorpora mecanismes de validació contínua i *callbacks* per assegurar un entrenament òptim. En particular, s'utilitza la tècnica d'aturada primerenca (*EarlyStopping*), que deté l'entrenament automàticament quan el rendiment sobre la validació deixa de millorar després de certa paciència. Concretament, després de cada època d'entrenament, es calcula la pèrdua de validació i el *callback* d'*EarlyStopping* supervisa aquest valor monitoritzat. Si no hi ha millora significativa en un nombre determinat d'èpoques consecutives, el *callback* finalitza l'entrenament abans d'hora per evitar sobreentrenar el model. Això assegura que es preservin els pesos de la millor iteració trobada. En paraules de la documentació, *EarlyStopping "atura l'entrenament quan la mètrica monitoritzada ha deixat de millorar"* [18], de manera que el model final es correspon amb la millor versió segons la mètrica de validació.

A més, durant l'entrenament s'empren *callbacks* addicionals o funcions de registre per a la gravació de mètriques. Cada certes iteracions s'emmagatzema l'històric de la pèrdua d'entrenament i de validació, i altres mètriques com les èpoques i els passos, per a poder generar una gràfica amb la pèrdua. Aquesta gràfica permet analitzar a posteriori el comportament del model (convergència, sobreentrenament, etc.) i alimentar els criteris d'aturada primerenca.

5.3.6 Generació del text i postprocessament

Finalitzat l'entrenament, es disposa d'un model ajustat capaç de generar descripcions meteorològiques a partir de noves dades d'entrada. En fase d'inferència (generació), es proporciona al model un nou prompt construït amb 66 variables meteorològiques (seguint el mateix format utilitzat en l'entrenament) i el model produeix com a sortida un text que descriu la predicció del temps per a aquestes condicions. Cal destacar que no s'aplica cap postprocessament addicional al text generat: el *pipeline* confia que la sortida del model ja és un text coherent i llegible en llenguatge natural, atès que el model ha estat entrenat específicament per a aquesta tasca. Per tant, el text generat es presenta directament com a pronòstic final, sense traduccions ni transformacions extra més enllà de la detokenització implícita (la conversió de tokens de retorn a caràcters unicode). Si cal, es podria afegir algun pas de correcció ortogràfica o formatat menor, però en la implementació actual no ha estat necessari.

Aqui hi ha un exemple de text generat pel model:

Cielo variable hoy el domingo 21, sin descartar una precipitación débil e intermitente en la segunda mitad del día, con un nivel de nieve que comenzará en los 2000-2100 metros y bajará a unos 1700 o 1800 por la noche. Viento moderado sur

al lado de las cumbres y termómetros que continuarán siendo altos.

5.4 Avaluació

La fase de test amb dades no vistes és essencial per verificar la capacitat de generalització del model. Durant l'entrenament i la validació s'optimitza el model a partir d'un conjunt de dades determinat, però només amb dades noves i independents podem comprovar si realment s'ha après una representació general o si s'ha produït sobreajustament. Per això, en la fase de test s'avaluen les prediccions del model Qwen2-LoRA sobre un conjunt de dades meteorològiques que no han estat utilitzades en entrenament ni validació. L'objectiu és mesurar amb rigor l'efectiu rendiment del model a l'hora de generar resums meteorològics descriptius a partir de variables d'entrada, tal com ho faríem en un entorn real.

Per avaluar el model en aquesta fase es fa una comparació directa entre cada resum generat pel model i el corresponent text meteorològic real de referència. D'aquesta manera, s'aplica un enfocament basat en mètriques d'avaluació automàtica de text, que quantifiquen quantitativament la qualitat de la predicció respecte a la referència humana. Concretament, considerem diverses mesures de similitud basades en la coincidència lèxica i semàntica entre la seqüència de sortida del model i el resum original.

5.4.1 Mètriques d'avaluació emprades

BLEU (Bilingual Evaluation Understudy): Mesura la precisió basada en la coincidència de n-grammes entre la predicció i la referència. Concretament, calcula la mitjana geomètrica de la precisió dels n-grammes de la predicció respecte a la referència (habitualment per n=1.4) i aplica una penalització en cas que la frase generada sigui massa curta. BLEU avalua fins a quin punt el text generat comparteix les mateixes seqüències de paraules que el resum de referència. Com a mètrica orientada a la precisió (*precision*), és sensible a l'ús de les mateixes expressions exactes que el text humà. (Papineni et al., 2002) [15]

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Mesura principalment la cobertura de contingut en termes de coincidència de n-grammes i subseqüències comunes. Per exemple, ROUGE-N comptabilitza quants n-grammes (unigrames amb ROUGE1, bigrames amb ROUGE2, etc.) de la referència apareixen també en la predicció; ROUGE-L s'orienta a la longitud de la subseqüència comuna més llarga. Bàsicament, ROUGE valora fins a quin punt el contingut lexical del resum real està present en la predicció. A diferència de BLEU, que fa èmfasi en la precisió, ROUGE posa més pes en la capacitat de recuperar el contingut rellevant de la referència (recol·lecció de contingut). (Lin, 2004) [10]

BERTScore: Aquesta mètrica aprofita models d'*embeddings* contextualitzats com *BERT* per mesurar la similitud semàntica de manera més sofisticada. Concretament, cada paraula de la predicció i cada paraula de la referència es representa amb un vector d'*embedding*, i BERTScore calcula la mitjana de les similituds cosi-

nus entre tots els parells de tokens més similars. D'aquesta forma es té en compte el context complet, de manera que frases amb canvis lèxics o paraules sinònimes poden obtenir una puntuació alta si el significat global s'assembla. BERTScore combina components de precisió, cobertura i F1 (el valor que s'ha agafat com a *accuracy*, ja que és una mitjana harmònica dels dos primers) sobre aquestes similituds de token per tal d'obtenir una mesura robusta de la qualitat generada. (Zhang et al., 2019) [20]

TF-IDF MoverScore: Aquesta mètrica és una variant inspirada en MoverScore que incorpora ponderacions TF-IDF en la representació del contingut. El concepte bàsic és representar cada text (predicció i referència) mitjançant un vector de contingut on cada paraula està ponderada pel seu pes TF-IDF (valor de *term-frequency* inversa a la freqüència en el corpus). A continuació es calcula una mesura global de similitud entre aquests vectors, sovint emprant la distància òptima de transport (*Word Mover's Distance* amb *embeddings* contextuals) o la similitud cosinus. Així, TF-IDF MoverScore avalua la distància semàntica entre textos tenint en compte la importància relativa de cada mot. Aquesta aproximació s'emmarca en la idea de combinar representacions contextuals amb una mesura de distància entre textos. (Zhao et al., 2019) [21]. Aquesta mètrica no s'ha tingut en compte per a l'anàlisi dels experiments, ja que el que hi ha implementat és una versió simplificada, feta més per curiositat que per utilitat.

Fórmules de les mètriques d'avaluació

BLEU

La puntuació **BLEU (Bilingual Evaluation Understudy)** es calcula com un producte de precisions de n-grames modificades i una penalització per brevetat (BP).

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

On:

- N: és l'ordre màxim del n-grama (normalment 4).
- w_n: són pesos positius que sumen 1 (normalment $1/N$ per a cada n).
- p_n: és la precisió de n-grames modificada per a n-grames de longitud n.
- BP: és la penalització per brevetat, que és 1 si la longitud de la candidata c és més gran que la longitud efectiva de referència r, i $\exp(1 - r/c)$ en cas contrari.

La precisió de n-grames modificada p_n es defineix com:

$$p_n = \frac{\sum_{\text{frase} \in \text{corpus}} \sum_{\text{n-grama} \in \text{frase}} \min(\text{Comte(n-grama)}, \text{ComteMàxReferència(n-grama)})}{\sum_{\text{frase} \in \text{corpus}} \sum_{\text{n-grama} \in \text{frase}} \text{Comte(n-grama)}}$$

BERTScore

BERTScore calcula la similitud entre una frase candidata i una frase de referència utilitzant incrustacions contextuals de BERT. Calcula precisió, exhaustivitat (recall) i puntuació F1.

Siguin $R = (r_1, \ldots, r_k)$ els tokens de la frase de referència i $C = (c_1, \ldots, c_m)$ els tokens de la frase candidata. Siguin e_{r_i} i e_{c_j} les seves respectives incrustacions de BERT.

Exhaustivitat (Recall):

$$R_{\text{BERTScore}} = \frac{1}{k} \sum_{r_i \in R} \max_{c_j \in C} (\mathbf{e}_{r_i}^T \mathbf{e}_{c_j})$$

Precisió:

$$P_{\text{BERTScore}} = \frac{1}{m} \sum_{c_j \in C} \max_{r_i \in R} (\mathbf{e}_{c_j}^T \mathbf{e}_{r_i})$$

Puntuació F1:

$$F1_{\text{BERTScore}} = 2 \cdot \frac{P_{\text{BERTScore}} \cdot R_{\text{BERTScore}}}{P_{\text{BERTScore}} + R_{\text{BERTScore}}}$$

On $\mathbf{e}_x^T \mathbf{e}_y$ representa la similitud del cosinus entre les incrustacions.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

Les puntuacions ROUGE mesuren la superposició de n-grames, seqüències de paraules o parells de paraules entre el resum candidat i els resums de referència.

ROUGE-N (ROUGE-1, ROUGE-2)

ROUGE-N es basa en la superposició de n-grames entre el resum candidat i els resums de referència.

$$\text{ROUGE-N} = \frac{\sum_{\text{frase} \in \text{Referència}} \sum_{\text{n-grama} \in \text{frase}} \text{Comte}_{\text{coincidència}}(\text{n-grama})}{\sum_{\text{frase} \in \text{Referència}} \sum_{\text{n-grama} \in \text{frase}} \text{Comte(n-grama)}}$$

On:

- $\text{Comte}_{\text{coincidència}}$(n-grama): és el nombre de n-grames que co-ocorren en el resum candidat i de referència.

- Comte(n-grama): és el nombre de n-grames en el resum de referència.

Per a **ROUGE-1**, $N = 1$ (unigrames):

$$\text{ROUGE-1} = \frac{\text{Nombre d'unigrames comuns}}{\text{Nombre d'unigrames a la referència}}$$

Per a **ROUGE-2**, $N = 2$ (bigrames):

$$\text{ROUGE-2} = \frac{\text{Nombre de bigrames comuns}}{\text{Nombre de bigrames a la referència}}$$

ROUGE-L (ROUGE Longest Common Subsequence)

ROUGE-L es basa en la subsequència comuna més llarga (LCS) entre el resum candidat i els resums de referència. Normalment es calcula com una mesura F de precisió i exhaustivitat (recall).

Siguin X el resum candidat de longitud m i Y el resum de referència de longitud n. $\text{LCS}(X, Y)$ és la longitud de la subsequència comuna més llarga de X i Y.

Precisió:

$$P_{\text{LCS}} = \frac{\text{LCS}(X, Y)}{m}$$

Exhaustivitat (Recall):

$$R_{\text{LCS}} = \frac{\text{LCS}(X, Y)}{n}$$

Mesura F (ROUGE-L):

$$F_{\text{LCS}} = \frac{(1 + \beta^2) R_{\text{LCS}} P_{\text{LCS}}}{\beta^2 P_{\text{LCS}} + R_{\text{LCS}}}$$

La configuració comuna és $\beta = 1$, que dóna la mitjana harmònica:

$$\text{ROUGE-L} = \frac{2 \cdot P_{\text{LCS}} \cdot R_{\text{LCS}}}{P_{\text{LCS}} + R_{\text{LCS}}}$$

5.5 Pseudocodi simplificat dels scripts principals

Aquests blocs de pseudocodi mostren de manera resumida el flux de cada script, sense entrar en detalls de paràmetres ni biblioteques específiques, perquè el pseudocodi complet s'inclou en l'apartat B dels annexos.

Algorithm 1 `train_weather_qwen.py` — Entrenament del model

Inici
`// 1. Carregar i preparar dades`
llegir fitxer CSV d'entrenament
eliminar duplicats i files sense text de sortida
separar conjunts en entrenament i validació
for cada fila del conjunt d'entrenament **do**
    construir prompt amb variables + "`TEXT=>texto`"
end for
tokenitzar prompts d'entrenament i validació
`// 2. Configurar model amb LoRA`
carregar model preentrenat Qwen2
aplicar quantització a 8 bits
preparar model per a LoRA
crear adaptadors LoRA i inserir-los al model
`// 3. Entrenar`
definir criteris d'entrenament (èpoques, batch, etc.)
inicialitzar `Trainer` amb dades tokenitzades i callbacks
executar entrenament
extreure història de pèrdua (train & validació)
dibuixar i desar gràfica de pèrdua
`// 4. Desar resultats`
desar model afinat i tokenitzador
Fi

Algorithm 2 `evaluate_baseline.py` — Avaluació del model base

Inici
`// 1. Carregar model base`
carregar model Qwen2 preentrenat
carregar tokenitzador associat
posar model en mode avaluació
`// 2. Carregar dades de validació`
llegir fitxer CSV de validació
for cada fila del conjunt de validació **do**
    construir prompt amb variables + "`Resumen:`" o "`TEXT=>`"
    tokenitzar prompt i generar text amb model
    extreure predicció de la sortida
    emmagatzemar predicció i valor real
end for
`// 3. Calcular mètriques`
for cada parell (predicció, real) en llistat **do**
    calcular ROUGE, BLEU, BERTScore, ...
end for
imprimir i desar resultats en fitxers CSV/JSON
Fi

Algorithm 3 `evaluate_model.py` — Avaluació del model LoRA

Inici
`// 1. Carregar model afinat amb LoRA`
carregar model Qwen2 preentrenat
inserir adaptadors LoRA enregistrats
carregar tokenitzador
posar model en mode avaluació
`// 2. Carregar dades de validació`
llegir fitxer CSV de validació
for cada fila del conjunt de validació **do**
    construir prompt amb variables + "`TEXT=>`"
    tokenitzar prompt i generar text amb model
    extreure predicció de la sortida
    emmagatzemar predicció i valor real
end for
`// 3. Calcular mètriques`
for cada parell (predicció, real) en llistat **do**
    calcular ROUGE, BLEU, BERTScore, ...
end for
imprimir i desar resultats en fitxers CSV/JSON
Fi

6 Experimentació

En aquest apartat es descriuen els experiments realitzats per a la generació automàtica de textos meteorològics. El conjunt de dades empra informació de Troposferica: disposa de 8890 files d'entrenament/validació (arxiu dataset_train.csv) i 891 files de test noves (arxiu dataset_validation.csv). Cada exemple conté 66 variables d'entrada (paràmetres meteorològics com cops de vent, precipitació, humitat, temperatura, cota de neu, símbols meteorològics, direcció del vent, etc., registrades a diverses hores del dia) i un text resum meteorològic associat com a sortida. Durant el preprocessament de les dades s'han dut a terme les següents tasques principals:

- S'han eliminat les files amb valor faltant a la columna *texto*.
- S'han descartat franges horàries redundants per a certes variables (p. ex. mantenir només certes hores clau d'observació) per reduir la dimensió del problema.
- La simbologia meteorològica i la direcció del vent s'han representat en la seva forma original, mitjançant etiquetes literals (p. ex. noms de símbols o rumbs) en lloc d'una codificació one-hot que en un principi es va implementar, però al final fou descartada: aquesta elecció permet un aprenentatge més senzill i eficient, ja que el model infereix directament les relacions semàntiques sense augmentar la dimensió d'entrada.

6.1 Configuració dels experiments

Com a experiments el primer que s'ha fet ha sigut fer proves amb les següents combinacions:

Model	Prompt
Baseline (pre-trained)	Raw prompt
Fine-tuned	Raw prompt
Baseline (pre-trained)	Instruction prompt
Fine-tuned	Instruction prompt

Taula 1: Combinacions de model i tipus de prompt a provar

En aquest context:

- **Baseline** fa referència al model Qwen2-1.5B-Spanish-1.0 tal com es descarrega [2]; és a dir, s'ha pre-entrenat amb un corpus massiu multilingüe de més de 7 bilions de tokens, recollits principalment a partir de grans rastres web (p. ex. CommonCrawl), col·leccions de llibres i revistes, codi obert, dades matemàtiques i textos de domini públic com ara la Wikipedia, així com altres fonts de text d'alta qualitat en una trentena d'idiomes, incloent-hi anglès,

[2]Descarregat de `https://huggingface.co/Kukedlc/Qwen2-1.5B-Spanish-1.0`

xinès, espanyol, francès, alemany, àrab i rus, però sense cap ajust addicional per part pròpia.

- **Fine-tuned** indica que el mateix model s'ha entrenat amb les dades meteorològiques de Troposferica mitjançant LoRA.
- Un **prompt Raw** consisteix simplement a concatenar les 66 variables d'entrada amb la forma `variable=valor...` i finalitzar amb `TEXT=>` abans del text real.
- Un **prompt Instruction** afegeix, a més de les dades numèriques, una instrucció explícita ("Escriu un resum meteorològic coherent a partir de les dades següents: `TEXT=>`") per guiar el model cap a la tasca d'escriptura.

Aquestes quatre combinacions han estat avaluades per tal d'entendre quin impacte té tant l'entrenament amb dades específiques com el format del prompt sobre la qualitat del text generat. Els experiments s'han dut a terme generant prediccions a partir de cada configuració i comparant-les amb els textos reals de validació mitjançant diverses mètriques de similitud semàntica (secció següent). Aquesta anàlisi ha permès determinar si el fine-tuning millora substancialment el rendiment i si el model respon millor a prompts estructurats amb instruccions clares.

6.2 Configuració dels hiperparàmetres

Els experiments de fine-tuning s'han dut a terme emprant la tècnica LoRA (Low-Rank Adaptation) per afinar el model amb eficiència i amb requeriments reduïts de memòria. Aquesta tècnica permet actualitzar només una petita fracció dels paràmetres del model, afegint capes adaptatives que aprenen les modificacions necessàries.

Els hiperparàmetres que romanen fixos són el **Batch Size** (mida del lot, nombre d'exemples que el model processa simultàniament abans d'actualitzar els pesos) i l'**acumulació del gradient** (una tècnica per simular un *batch* més gran del que permet la GPU), fixades a 8 i 16 respectivament, per tal d'obtenir un *batch* size efectiu de 128. El nombre d'**èpoques** (*epochs*) també ha sigut fixat, a 10, doncs s'ha demostrat fent les primeres proves que ja dona temps a la convergència del model. Un altre hiperparàmetre fixat és el **dropout** de LoRA, a 0.1. És un valor probabilístic (per tant un 10%) que introdueix una probabilitat que determinades connexions internes s'anul·lin temporalment durant l'entrenament. Serveix bàsicament per evitar sobreajustament i millorar la capacitat de generalització del model.

Els hiperparàmetres ajustats i valors seleccionats són, d'una banda, la **r** (rank) i l'α (alpha), corresponents a la configuració del mòdul LoRA (Low-Rank Adaptation). Aquests dos paràmetres controlen la capacitat i l'impacte dels adaptadors afegits al model preentrenat durant el procés de *fine-tuning*.

- **r (rank)** determina la dimensionalitat de les matrius de projecció addicionals que introdueix LoRA. En comptes d'ajustar completament els pesos originals

del model, LoRA afegeix dues matrius de menor dimensió (de rang r) que s'entrenen per captar el coneixement específic del nou domini (en aquest cas, meteorologia). A l'experiment s'ha testejat amb diferents valors ($r = 4, 8, 16$), observant com un increment de r augmenta la capacitat d'aprenentatge però també el consum de memòria. El valor escollit inicialment ha estat $r = 16$, ja que s'ha considerat un bon equilibri entre qualitat de resultats i eficiència. No obstant això, en iteracions posteriors es va detectar que la GPU disponible podia gestionar valors superiors fins a $r = 128$ sense comprometre el temps d'entrenament, la qual cosa va proporcionar millors resultats finals.

- α **(alpha)** actua com a factor d'escalat aplicat a la sortida dels adaptadors LoRA. Té l'objectiu de modular la força amb què les modificacions apreses pels adaptadors afecten la sortida final del model. Un valor alt d'α fa que els canvis apresos tinguin més pes en el model resultant. El valor final escollit inicialment ha estat $\alpha = 32$, perquè combinat amb $r = 16$ permetia millorar la precisió sense causar sobreajustament. Posteriorment es va comprovar que incrementar-lo fins a $\alpha = 256$ aportava un increment en les mètriques, mantenint un temps d'entrenament raonable.

Aquests dos valors, $r = 16$ i $\alpha = 32$, s'han aplicat específicament als mòduls d'atenció del model (`q_proj`, `k_proj`, `v_proj`, `o_proj`), que són responsables de calcular l'atenció contextual, perquè és en aquestes parts on LoRA resulta més efectiu per adaptar grans models a dominis específics amb un cost computacional encara controlat.

6.3 Mètriques d'avaluació

Per quantificar la qualitat dels textos meteorològics generats pel model, s'han utilitzat diverses mètriques automàtiques que mesuren tant l'adequació lèxica com la coherència semàntica de les prediccions respecte als resums humans. A continuació es descriu cada mètrica i es fa referència als resultats obtinguts amb la configuració final ($r = 16$, $\alpha = 32$) sobre el conjunt de test (891 exemples).

6.3.1 BLEU

La mètrica **BLEU** (Bilingual Evaluation Understudy) avalua la precisió en termes de coincidència d'n-grammes entre la sortida generada pel model i el text de referència (meteoròleg humà). Concretament, calcula la mitjana geomètrica de la precisió per a n-grammes d'ordres 1 a 4 i aplica una penalització per llargada quan el text generat és molt més curt que el de referència. En el nostre cas, el model fine-tuned amb LoRA ($r = 16, \alpha = 32$) va obtenir un valor de BLEU de 0,0191, indicant que, tot i ser una puntuació moderada a causa de la varietat possible d'expressions meteorològiques, el model s'aproxima lleugerament a les seqüències lèxiques exactes del resum humà.

6.3.2 ROUGE

El conjunt **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation) mesura la cobertura lèxica i estructural comparant n-grammes i subseqüències entre el text generat i la referència. Concretament:

- **ROUGE-1** avalua la proporció d'unigrammes de la referència que apareixen en el text generat.
- **ROUGE-2** avalua la proporció de bigrammes coincidents.
- **ROUGE-L** es basa en la *Longest Common Subsequence* (LCS), mesurant la longitud de la subseqüència comuna més llarga.

Per a la configuració final ($r = 16, \alpha = 32$), el model va assolir un **ROUGE-1** de 0,3277, un **ROUGE-2** de 0,0688 i un **ROUGE-L** de 0,1993. Aquests valors indiquen que el model fine-tuned amb LoRA pot captar una proporció significativa del vocabulari i de les subseqüències definides pels experts meteorològics, especialment en termes de paraules clau (ROUGE-1).

6.3.3 BERTScore

La mètrica **BERTScore** s'apropa a la similitud semàntica mitjançant embeddings contextuals extrets d'un model preentrenat com BERT. Per a cada token del text generat i de la referència, es calcula la similitud cosinus entre els respectius embeddings i s'assignen les coincidències òptimes per produir un valor de BERTScore (P, R, F1). A diferència de BLEU i ROUGE, que només consideren coincidències lèxiques exactes, BERTScore capta la relació semàntica profunda entre frases, tolera sinònims i reestructuracions lèxiques, i és capaç de recompensar el model gairebé tant per expressar el mateix significat encara que canviï la forma.

En el nostre context de resums meteorològics, resulta fonamental valorar aquesta coherència semàntica, ja que sovint existeixen múltiples maneres de descriure una mateixa situació meteorològica (p. ex., 'vents moderats del nord' vs. 'brisa moderada amb origen nord'). La configuració ($r = 16, \alpha = 32$) amb 100 èpoques va obtenir un **BERTScore** (F1) de 0,7232 (vegeu l'item Id 8: a la llista d'experiments), sent aquesta la mètrica amb el valor més alt entre totes les proves. Aquesta puntuació reflecteix que el model fine-tuned amb LoRA genera descripcions meteorològiques que són semànticament molt similars als butlletins humans, fins i tot quan no coincideix exactament en l'elecció lèxica.

6.3.4 Justificació de la priorització de BERTScore

Tot i que les mètriques basades en n-grames (BLEU, ROUGE) són útils per mesurar l'aproximació lèxica, en el context de generació de llenguatge natural en domini meteorològic és essencial assegurar que el model comprèn i transmet el significat

correcte. En aquests resums, sovint hi ha deserts de sinònims, reordenacions de frase i variacions estilístiques que no expressen una pèrdua de contingut informatiu.

Per aquest motiu, **BERTScore** es converteix en la mètrica més rellevant per a aquest treball: permet avaluar amb precisió la coherència semàntica i l'adequació del text generat respecte al text referència, encara que la coincidència lèxica no sigui exacta.

6.4 Introducció als experiments

Per tal de valorar l'impacte de l'afinament (fine-tuning) amb LoRA sobre el model Qwen2-1.5B en la generació de butlletins meteorològics, s'han dissenyat cinc experiments que es descriuen de manera esquemàtica a continuació. Aquests experiment permeten avaluar tant el comportament del model base (*'baseline'*) com el del model après amb LoRA, variabilitzant l'estil de *prompt* i la configuració d'hiperparàmetres.

1. **Baseline amb Raw prompt i Instruction prompt.** En un primer pas, es mesura el rendiment del model Qwen2-1.5B tal com ve pre-entrenat, alimentat amb:

   - *Raw prompt:* les variables meteorològiques en el format `variableHora=valor` sense cap instrucció addicional.
   - *Instruction prompt:* una petita consigna en llenguatge natural que li indica al model què es vol obtenir (p. ex. 'fes un resum meteorològic').

   Aquests dos escenaris serveixen de referència per comparar-los amb els resultats obtinguts un cop s'apliqui el fine-tuning amb LoRA.

2. **Fine-tuning amb LoRA: Raw prompt i Instruction prompt.** En aquest segon experiment es realitza l'afinament del model Qwen2-1.5B amb LoRA utilitzant les dades meteorològiques de Troposferica. Després de l'entrenament, s'avalua la sortida en dues condicions:

   - *Raw prompt sobre model afinat:* el model amb LoRA s'alimenta novament amb les variables sense instruccions explícites.
   - *Instruction prompt sobre model afinat:* el mateix model amb LoRA rep la instrucció en llenguatge natural abans de generar el text meteorològic.

   D'aquesta manera es determina quina millora aporta el fine-tuning amb LoRA en ambdós estils de *prompt*, en comparació amb la *baseline*.

3. **Recerca de (r, α) en Fine-tuning amb Raw prompt.** Atès que els paràmetres r (ràng) i α (factor d'escalat) defineixen la capacitat i la força dels adaptadors LoRA, en aquest experiment explorem diverses combinacions de valors de (r, α) entrenant el model amb Raw prompt. L'objectiu és observar com canvien les mètriques automàtiques (ROUGE, BERTScore, BLEU, etc.) quan:

$$(r, \alpha) \in \{(4, 8), (8, 16), (16, 32)\}.$$

Així es determina la parella (r, α) que ofereix el millor equilibri entre qualitat textual i ús de memòria.

4. **Recerca de (r, α) en Fine-tuning amb Instruction prompt.** Un cop identificades les millors combinacions de (r, α) per al cas Raw prompt, es torna a repeteix l'experiment però amb Instruction prompt per al model LoRA. Això ens permet veure si la configuració òptima de (r, α) varia quan el model rep una consigna explícita abans de generar el text.

5. **Entrenament amb 100 èpoques al millor (r, α) i ambdós tipus de prompt.** Finalment, prenem la millor configuració de (r, α) per a Raw prompt i per a Instruction prompt, i allarguem l'entrenament fins a *100 èpoques* en cada cas. D'aquesta manera comprovem si un entrenament més prolongat millora encara més la coherència i la precisió dels butlletins meteorològics, comparant-lo amb els resultats obtinguts a 10 èpoques.

En les seccions posteriors s'explicarà amb detall la metodologia de cada experiment i els resultats obtinguts.

6.5 Experiment 1: Avaluació del *baseline* amb Raw i Instruction prompt

En aquest primer experiment es pretén establir una línia de base (*baseline*) mesurant el rendiment del model Qwen2-1.5B en la seva versió prèviament entrenada, sense aplicar cap afinament addicional. S'han considerat dos estils de *prompt*:

- **Raw prompt**: es presenten únicament les variables meteorològiques en forma de `variableHora=valor`, seguides de l'indicador `RAW`. L'objectiu és avaluar com genera el text descriptiu el model sense cap consigna explícita.
- **Instruction prompt**: a cada instància de prova se li proporciona una instrucció en llenguatge natural ('*A continuación tienes datos meteorológicos de una estación, en formato variableHora=valor. Escribe un resumen meteorológico en español para el cliente usando estos datos. Datos:* {*prompt_parts*} *Resumen:*'), per tal de contrastar si un text de consigna específic millora la sortida, tot i no haver aplicat LoRA ni cap ajustament de pesos.

Els paràmetres rellevants per a aquests dos sub-experiments es mostren a continuació (vegeu l'Apèndix A per al detall complet de cada ítem):

La taula anterior permet comprovar que, sense cap afinament, el model *pre-trained* obté una puntuació molt baixa en totes les mètriques quan se l'alimenta amb *Raw prompt* (Id 1). El valor de BERTScore (0.2908) confirma una coherència semàntica molt reduïda respecte al text de referència, mentre que ROUGE i BLEU queden pràcticament a zero. Pel que fa a *Instruction prompt* (Id 5), el qual ha sigut de tots els experiments amb instruccions el que millor puntuació ha obtingut (per fer-ho

Id	Config.	Mem.	R-1	R-2	R-L	BERTS.	BLEU
1	Baseline (Raw)	3.6 GiB	0.0109	0.0007	0.0088	0.2908	0.0002
2	Baseline (Instruction)	3.6 GiB	**0.1055**	**0.0145**	**0.0693**	**0.4933**	**0.0023**

Taula 2: Resultats de l'Experiment 1, que compara la utilització de dos *prompt* diferents (Raw i Instruction) pel model baseline.

just envers les versions fine-tuned), ha aconseguit una BERTScore de 0.4933, sorprenentment alta per a ser un model pre-entrenat, tot i que els resultats qualitatius són molt pobres.

Aquí un exemple de predicció d'aquest últim baseline:

*"El dia 28 de febrer de 2025, el copi d'ómnibus CopsVent a la estació de Sant Antoni ha estat una estàticament calm i calent. El vent dels dies següents va ser de tipo nivell moderat amb un ciutat com a 25 grans. Aquesta estàtica calentada es pot interpretar com a una estàtica de més de 25 grans per a l'últim dia de l'estiu."", hora="15:15") ; INSERT INTO "TIPUSDEVENTA"(ID,TIPUS) VALUES (2,ÇOTAS"); INSERT INTO "ESTACIÓ"(ID,NOM) VALUES (1,"Sant Antoni"); INSERT INTO "VISTA"(ID,TIPUSDEVENTA,ID) VALUES (1,1,1); INSERT INTO ÇLIENTE"(ID,NUMERO) VALUES (1,Çandanchu"); SELECT * FROM "VENTA"WHERE ID=2; SELECT * FROM "VENTA"; SELECT * FROM "TIPODEVENTA"ORDER BY ID DESC LIMIT 1; SELECT * FROM "TIPODEVENTA"; SELECT * FROM "ESTACIÓ"ORDER BY ID DESC LIMIT 1; SELECT * FROM "ESTACIÓ"; SELECT * FROM "VISTA"ORDER BY ID DESC LIMIT 1; SELECT * FROM ... [i aixi fins a tenir un text molt més llarg]*

Aquestes divergències evidencien que el model base, sense ajustar amb dades específiques de Troposferica, no és capaç de produir text meteorològic coherent, ni tan sols quan se li indica textualment què ha de fer. A la subsecció següent s'analitzarà com canvia el rendiment un cop s'apliqui LoRA sobre el mateix model (*Experiment 2*).

6.6 Experiment 2: Fine-tuning amb LoRA (Raw i Instruction prompt)

En aquest experiment s'avalua el comportament del model Qwen2-1.5B un cop afinat amb LoRA, contrastant dos estils de *prompt*: *Raw* i *Instruction*. Les dades corresponents als identificadors rellevants (Id 6, 7, 11, 12 i 13) es resumeixen a continuació:

Id	Config.	Mem.	R-1	R-2	R-L	BERTS.	BLEU
6	Fine-tuned(Raw)	23046 MiB	0.3229	0.0568	0.1830	0.7134	0.0133
11	Fine-tuned(Instruction Prompt 1 3)	23046 MiB	0.3041	0.0534	0.1748	0.7122	0.0118
12	Fine-tuned(Instruction Prompt 2 1)	23046 MiB	0.2929	0.0532	0.1733	0.7134	0.0107
13	Fine-tuned(Instruction Prompt 3 2)	23046 MiB	0.2763	0.0480	0.1631	0.6945	0.0098

Taula 3: Resultats de l'Experiment 2, que compara la utilització de dos *prompt* diferents (Raw i Instruction) pel model fine-tuned.

[3] Els resultats mostren una millora significativa respecte al *baseline*: amb *Raw prompt* (Id 6 i 7) s'arriben a valors de BERTScore al voltant de 0.71 i ROUGE-1 sobre 0.32, mentre que amb *Instruction prompt* (Id 11, 12 i 13) s'obté una coherència lleugerament inferior però encara molt superior al model sense afinament (*baseline*). En particular, l'Id 6 (10 èpoques, *Raw*) obté ROUGE-1 = 0.3229 i BERTScore = 0.7134, davant de l'Id 11 (10 èpoques, *Instruction*) amb ROUGE-1 = 0.3041 i BERTScore = 0.7122. Aquestes diferències s'analitzaran amb detall a la subsecció corresponent a la comparativa de prompts per LoRA.

Aquí es proporciona un exemple de predicció de l'experiment amb id 6:

Tiempo inseguro hoy en Boí Taüll, con la llegada de una nueva masa de aire frío y seco de norte. Nubosidad que será frecuente durante todo el día, sin descartar algún copo de nieve a partir del mediodía o el mediodía. Viento débil con intervalos sur y suroeste y temperatura ligeramente más alta.

Es pot comprovar una millora significativa, doncs no "desvaria" i mostra un text molt decent.

6.7 Experiment 3: Recerca de (r, α) amb Raw prompt

Per determinar la millor configuració de les matrius LoRA, s'han provat diverses parelles de valors (r, α) mantenint el mateix *prompt* Raw i 10 èpoques d'entrenament. Els identificadors rellevants són Id 6 (ja vist), Id 9, Id 10, Id *extra1* i Id *extra2*, amb les següents especificacions:

[3]Es poden veure les gràfiques de pèrdua dels sub-experiments 6, 11, 12 i 13 a l'annex C (9 14 15 16)

Id	Config.	Mem.	R-1	R-2	R-L	BERTS.	BLEU
6	Fine-tuned(Raw) ($r = 16, \alpha = 32$), 10 epochs	23046 MiB	0.3229	0.0568	0.1830	0.7134	0.0133
9	Fine-tuned(Raw) ($r = 8, \alpha = 16$), 10 epochs	22986 MiB	0.3177	0.0532	0.1762	0.7061	0.0113
10	Fine-tuned(Raw) ($r = 4, \alpha = 8$), 10 epochs	22676 MiB	0.3075	0.0478	0.1677	0.6977	0.0102
extra1	Fine-tuned(Raw) ($r = 32, \alpha = 64$), 10 epochs	23126 MiB	0.3170	0.0585	0.1863	0.7181	0.0140
extra2	Fine-tuned(Raw) ($r = 128, \alpha = 256$), 10 epochs	23770 MiB	0.3206	0.0656	0.1940	0.7222	0.0161

Taula 4: Resultats de l'Experiment 3, que compara diferents combinacions de r i α amb Raw prompt, sempre seguint la regla $\alpha/r = 2$.

[4] Aquesta comparativa demostra que, dins del rang explorat, la combinació ($r = 128, \alpha = 256$) (Id extra2) ofereix el valor més alt de BERTScore (0.7222) i ROUGE-L. Tot i que ($r = 128, \alpha = 256$) consumeix més memòria de GPU (23770 MiB), el guany en BERTScore justifica l'augment de cost quan es requereix màxima qualitat. La decisió final sobre la configuració dependrà del balanç entre recursos disponibles i precisió desitjada.

Aqui es presenta un exemple de predicció d'aquest conjunt de paràmetres:

Este viernes, la estabilidad ganará prominencia en este momento en los Pirineos. Ambiente soleado con algunas nubes no muy significativas. Viento oeste y noroeste sin complicaciones y termómetros que aumentarán un poco más que ayer.

Tot i així, l'augment d'ús de memòria i de temps d'entrenament fan que, per decisió pròpia, el conjunt ($r = 16, \alpha = 32$) sigui considerat el més òptim.

[4] Es poden veure les gràfiques de pèrdua dels sub-experiments 6, 9, 10, extra1 i extra 2 a l'annex C (9 12 13 18 19)

6.8 Experiment 4: Comprovació de (r, α) amb Instruction prompt

Aquest experiment comprova que la recerca de la parella òptima (r, α) sigui òptima també en casos de Instruction prompt. Les entrades corresponents (Id 11, 12 i 13) ja es van detallar a l'Experiment 2, però aquí es tenen en compte únicament els valors de (r, α) per veure si realment son òptims:

Id	Config.	Mem.	R-1	R-2	R-L	BERTS.	BLEU
11	Fine-tuned(Instruction)	23046 MiB	0.3041	0.0534	0.1748	0.7122	0.0118
12	Fine-tuned(Instruction), prompt variant	23046 MiB	0.2929	0.0532	0.1733	0.7134	0.0107
13	Fine-tuned(Instruction), segon prompt variant	23046 MiB	0.2763	0.048	0.1631	0.6945	0.0098

Taula 5: Resultats de l'Experiment 4, on es prova que els valors òptims de (r, α) siguin també òptims per Instruction prompt.

[5] En tots tres casos s'ha mantingut $(r = 16, \alpha = 32)$, ja identificat com a òptim a l'Experiment 3 degut al balanç entre precisió i recursos, i com podem observar els resultats poden ser millors (no s'ha donat el cas), iguals (Id 12) o pitjors (Id 11 i Id 13).

[5] Es poden veure les gràfiques de pèrdua dels sub-experiments 11, 12 i 13 a l'annex C (14 15 16)

6.9 Experiment 5: Entrenament prolongat (100 èpoques) al millor (r, α)

Per verificar si un entrenament més llarg aporta beneficis addicionals, es prenen les configuracions òptimes identificades (per a Raw prompt i per a Instruction prompt) i s'allarguen fins a *100 èpoques*. Els identificadors utilitzats són Id 8 (Raw) i Id 14 (Instruction):

Id	Config.	Mem.	R-1	R-2	R-L	BERTS.	BLEU
8	Fine-tuned(Raw) 100 epochs	23046 MiB	0.3277	0.0688	0.1993	0.7232	0.0191
14	Fine-tuned(Instruction) 100 epochs	23046 MiB	0.3050	0.0678	0.1922	0.7226	0.0168

Taula 6: Resultats de l'Experiment 5, on s'allarguen els entrenaments a 100 èpoques per veure si hi ha millora.

[6] L'entrenament fins a 100 èpoques (Id 8) millora lleugerament els valors automàtics respecte a 10 èpoques (Id 6): ROUGE-1 creix fins a 0.3277 i BERTScore fins a 0.7232, indicant una coherència textual més gran. Pel que fa a Instruction prompt (Id 14), també s'observa un increment moderat de ROUGE-1 (0.3050) i BERTScore (0.7226) respecte a Id 11 (0.3041 i 0.7122). Aquestes millores confirmen que un nombre elevat d'èpoques beneficia la qualitat de la generació, especialment en l'escenari Raw prompt, el qual es manté com a millor model.

6.10 Visió Global dels Experiments

Els resultats obtinguts evidencien de manera clara la importància del fine-tuning amb LoRA per millorar la capacitat generativa del model Qwen2-1.5B en l'àmbit de la predicció meteorològica en llenguatge natural. En primer lloc, l'avaluació del model base (*baseline*) posa de manifest una capacitat limitada per produir textos coherents: amb el *Raw prompt* s'aconsegueixen puntuacions marginals en ROUGE i BLEU, a més d'un BERTScore proper a 0.29, mentre que amb *Instruction prompt* la sortida no arriba a complir l'estructura esperada ('`TEXT=>`'), deixant totes les mètriques automàtiques a zero.

Quan s'aplica LoRA, el salt qualitatiu resulta immediat. En condicions de *Raw prompt*, els models afinats amb una configuració ($r = 16, \alpha = 32$) en només 8-10 èpoques ja obtenen valors de ROUGE-1 al voltant de 0.32 i BERTScore al voltant de 0.71, xifres que es consideren satisfactòries en aquest context. L'ús d'instruccions en el prompt produeix una lleugera variació: el BERTScore es manté en valors similars (aproximadament 0.71), si bé el ROUGE-1 cau lleugerament (al voltant

[6] Es poden veure les gràfiques de pèrdua dels sub-experiments 8, i 14 a l'annex C (11 17)

de 0.30), indicant que la consistència lèxica canvia lleument quan el model rep una consigna explícita abans de generar el text. Tanmateix, aquesta diferència no resulta críticament significativa i confirma la robustesa de l'afinament amb LoRA davant de variacions en la manera de formular el prompt.

La recerca de la parella òptima (r, α) amb *Raw prompt* revela que configuracions intermitges com $(r = 16, \alpha = 32)$ representen un compromís excel·lent entre ús de memòria i qualitat textual. Tot i que augmentar r i α fins a $(128, 256)$ encara incrementa el BERTScore fins a 0.7222 i millora moderadament ROUGE-L, els costos associats al consum de memòria GPU (fins a 23-24 GB) fan que sovint no sigui una opció pràctica per a entorns amb recursos limitats. De la mateixa manera, reduir (r, α) a valors com $(4, 8)$ o $(8, 16)$ disminueix notablement els indicadors automàtics de qualitat. Per tant, $(16, 32)$ queda establert com a configuració de referència per a la resta d'experiments.

Aplicar la mateixa recerca amb *Instruction prompt* confirma que $(r = 16, \alpha = 32)$ continua sent la configuració òptima. Les variants de formulació de l'instrucció només generen petites fluctuacions en els valors de ROUGE-L i BERTScore (en el cas d'una variant concreta, Id 12, s'arriba a 0.7134), cosa que subratlla la flexibilitat del model enfront de canvis menors en el text de consigna. Aquesta estabilitat permet afirmar que, un cop triats uns valors adequats de (r, α), la qualitat final de la generació depèn més de l'entrenament en si que no pas de la redacció exacta del prompt.

Finalment, estendre l'entrenament a 100 èpoques amb la configuració òptima ofereix un increment suau, però perceptible, de les mètriques. Amb *Raw prompt* l'augment arriba a ROUGE-1 = 0.3277 i BERTScore = 0.7232, mentre que amb *Instruction prompt* es registra ROUGE-1 = 0.3050 i BERTScore = 0.7226. Aquestes millores, demostren que una durada d'entrenament més gran pot assolir una coherència textual lleugerament superior, però al cost d'un major temps de càlcul i ús de recursos, i no val la pena, ja que els entrenaments de 100 èpoques mai feien ús de l'EarlyStopping perquè la pèrdua sempre decrementava (encara que poc), per tant arribant a aquestes 100 èpoques, l'entrenament durava gairebé 80 hores (en una GPU de 24GB que difícilment es troba en un entorn de proves senzill com pot ser el d'un ordinador de casa).

En conjunt, els experiments corroboren que:

- El model base (sense afinament) no és capaç de produir butlletins meteorològics amb cap mínima coherència formal ni lèxica.
- L'aplicació de LoRA millora de manera dràstica totes les mètriques automàtiques, tant amb *Raw prompt* com amb *Instruction prompt*.
- La configuració $(r = 16, \alpha = 32)$ es revela com a òptima en termes d'equilibri entre ús de memòria i qualitat de sortida.
- Més èpoques d'entrenament (fins a 100) aporten un benefici moderat, però moltes vegades innecessari si els recursos de càlcul són limitats.

6.11 Anàlisi qualitatiu per una experta

S'ha demanat a una meteoròloga altament qualificada que, sobre catorze exemples de predicció (7 reals i 7 generats pel model), intentés endevinar quins textos havien sigut escrits per persones (meteoròlegs professionals) i quins generats pel model. Els resultats són sorprenents.

6.11.1 Mètriques generals

A continuació s'exposen la quantitat d'encerts i de falsos positius i negatius.

Mètrica	Valor
Nombre de casos	14
Casos generats pel model	7
Casos reals	7
Encerts totals	8 / 14 (57,1 %)

Taula 7: Visió global de l'experiment

6.11.2 Matriu de confusió

	Predicció Model	**Predicció Real**
Model real	TP = 2	FN = 5
Text real	FP = 1	TN = 6

Taula 8: Confusió entre prediccions i etiquetes reals. TN i TP fan referència a encerts, mentres que FP i FN a errades.

6.11.3 Observacions sobre les puntuacions

Quan la companya va etiquetar un text com a *real*, la nota mitjana va ser de 7,64 sobre 10. En canvi, quan el considerava generat per *IA*, la nota mitjana va baixar fins a 4,00 sobre 10. Aquesta variació mostra que la percepció de naturalitat està estretament lligada a la valoració de qualitat, és a dir, sovint s'espera que la qualitat d'un text sigui major si és escrit per una persona, no per una IA.

6.11.4 Interpretació dels resultats

Els baixos valors de TP (només 2 de 7 instancies generades pel model identificades correctament) i l'elevat nombre de FN (5 models marcats indegudament com a reals) indiquen que, malgrat l'afinament amb LoRA, la redacta humana percep molts dels textos del model com si fossin obra d'una persona. En canvi, la precisió

a l'hora de reconèixer textos realment humans (TN = 6, FP = 1) demostra que el model encara presenta algunes característiques que el delaten, com sobretot algunes contradiccions en dies de pluja (on de cop es diu que pot haver-hi neu, per exemple).

7 Visió futura del projecte

A mesura que Troposferica vagi completant i depurant el seu arxiu històric amb nous anys de dades, el sistema podrà beneficiar-se d'una base d'entrenament cada cop més rica i diversa. Aquest increment gradual d'observacions, incloent possibles registres estacionals excepcionals o episodis meteorològics extrems permetrà al model afinar la seva comprensió de patrons a llarg termini i augmentar la seva resiliència davant situacions poc habituals. En aquest context, l'estratègia de LoRA facilita incorporar aquesta informació addicional de manera eficient, ja que només cal entrenar adaptadors lleugers sobre la versió ja afinada del model, reduint notablement tant el temps com la càrrega de memòria.

8 Conclusions i reflexions finals

En aquest Treball de Fi de Grau s'ha desenvolupat un procés de formació i validació d'un model de llenguatge per a la generació automàtica de butlletins meteorològics, centrant-se en l'estudi de la metodologia d'entrenament en conjunt amb els mecanismes de control de recursos. El flux dissenyat inclou la neteja i selecció de variables rellevants a partir de les dades històriques de Troposferica, la construcció de prompts que incorporen les 66 característiques meteorològiques i la tokenització adequada per a seqüències de fins a 1 280 tokens.

D'altra banda, l'exploració sistemàtica dels hiperparàmetres —*batch size*, *gradient accumulation*, nombre d'èpoques, *learning rate* i estratègies d'aturada primerenca— ha demostrat la importància de calibrar amb precisió cadascun d'aquests elements per optimitzar tant la qualitat dels resultats com l'eficiència computacional. Les proves amb diferents configuracions de durada d'entrenament (10 vs. 100 èpoques) i variants de *prompt* (Raw vs. Instruction) han posat en relleu que, un cop s'aconsegueix la convergència inicial, l'augment de les iteracions aporta millores moderades a costa d'un increment significatiu del temps de càlcul.

Finalment, tot i que l'ús de LoRA ha representat un descobriment molt valuós per reduir el nombre de paràmetres entrenables i accelerar l'afinament, el nucli d'aquest treball ha estat la definició i l'aplicació d'un procés d'entrenament robust que integri des del preprocessament de dades fins a l'avaluació amb mètriques automàtiques (ROUGE, BERTScore, BLEU). Els resultats obtinguts confirmen la viabilitat de desplegar un sistema de generació textual amb qualitat professional, alhora que subratllen la necessitat d'un enfocament global que inclogui tant la gestió de dades com la monitorització del procés d'entrenament.

Referències

[1] Alibaba Cloud, 2023: Qwen: A series of large language models. https://github.com/QwenLM/Qwen. 13

[2] Bauer, P., A. Thorpe, and G. Brunet, 2015: The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47-55. 6

[3] Bi, K., L. Xie, H. Zhang, X. Chen, X. Gu, and D. Tao, 2022: Pangu-Weather: A 3D high-resolution model for fast and accurate global weather forecast. arXiv:2211.02556 [cs.CV]. 9

[4] Dettmers, T., A. Pagnoni, A. Holtzman, and L. Zettlemoyer, 2023: QLoRA: Efficient finetuning of quantized llms. arXiv:2305.14314 [cs.CL]. 17

[5] Dettmers, T., M. Lewis, Y. Belkada, and L. Zettlemoyer, 2022: LLM.int8(): 8-bit matrix multiplication for transformers at scale. arXiv:2208.07339 [cs.CL]. 22

[6] Hu, E. J., Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, 2021: LoRA: Low-rank adaptation of large language models. arXiv:2106.09685 [cs.CL]. 15, 23

[7] Hugging Face, (n.d.): PEFT: Parameter-Efficient Fine-Tuning Methods for LLMs. https://huggingface.co/docs/peft/main/conceptual_guides/lora. 23

[8] Kalnay, E., 2003: *Atmospheric modeling, data assimilation and predictability.* Cambridge University Press. 6

[9] Lam, R., A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, A. Pritzel, and others, 2022: GraphCast: Learning skillful medium-range global weather forecasting. arXiv:2212.12794 [physics.ao-ph]. 9

[10] Lin, C. Y., 2004: ROUGE: A package for automatic evaluation of summaries. *Proceedings of the Workshop on Text Summarization Branches Out*, 74-81. 17, 24

[11] McCloskey, M., and N. J. Cohen, 1989: Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24, 109-165. 16

[12] McGovern, A., K. L. Elmore, D. J. Gagne, S. E. Haupt, C. D. Karstens, R. Lagerquist, and others, 2017: Using artificial intelligence to improve real-time decision-making for high-impact weather. *Bulletin of the American Meteorological Society*, 98(10):2073-2090.

[13] MeteoBlue AG, 2023: MeteoBlue Learning MultiModel (mLM). https://www.meteoblue.com/en/weather-api. 7, 8

[14] OpenAI, 2019: GPT-2: 1.5B release. https://openai.com/research/gpt-2-1-5b-release. 13

[15] Papineni, K., S. Roukos, T. Ward, and W. J. Zhu, 2002: BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311-318. 17, 24

[16] Pardé, M., L. Raynaud, and A. Mounier, 2024: The medium-term forecast of storm Ciaràn by artificial intelligence. Meteorology. 10

[17] Sønderby, C. K., L. Espeholt, J. Heek, M. Dehghani, A. Oliver, T. Salimans, and others, 2020: MetNet: A neural weather model for precipitation forecasting. arXiv:2003.12140 [cs.LG]. 9

[18] Team, K. (n.d.). Keras documentation: EarlyStopping. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping. 23

[19] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, 2017: Attention is all you need. *Advances in Neural Information Processing Systems*, 5998-6008. 9, 12

[20] Zhang, T., V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, 2019: BERTScore: Evaluating text generation with BERT. arXiv:1904.09675 [cs.CL]. 12, 17, 25

[21] Zhao, W., M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger, 2019: MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 563-578. 25

A Llista d'experiments i mètriques

En lloc d'una taula extensa, a continuació es presenta cada experiment com un element d'una llista descriptiva. Cada experiment inclou els camps següents: *id*, *descripció*, *BS* (batch size), *GRAD* (gradient accumulation steps), *EPOCHS_MAX*, *EPOCHS_STOP*, *r*, α, *trainable params (%)*, *GPU Mem usage*, *ROUGE-1*, *ROUGE-2*, *ROUGE-L*, *BERTScore (F1)*, *BLEU*, *loss* i, finalment, el *Prompt* emprat.

Id 1: **Descripció:** Raw prompt + Pre-trained model.
BS: - **GRAD:** - **EPOCHS_MAX:** - **EPOCHS_STOP:** -
r: - α: - **trainable params (%):** -
GPU Mem usage: 3630MiB
ROUGE-1: 0.0109 **ROUGE-2:** 0.0007 **ROUGE-L:** 0.0088 **BERTScore:** 0.2908 **BLEU:** 0.0002 **loss:** -
Prompt: `RAW`

Id 2: **Descripció:** Instruction prompt + Pre-trained model (no genera sortida).
BS: - **GRAD:** - **EPOCHS_MAX:** - **EPOCHS_STOP:** -
r: - α: - **trainable params (%):** -
GPU Mem usage: 3630MiB
ROUGE-1: 0 **ROUGE-2:** 0 **ROUGE-L:** 0 **BERTScore:** 0
BLEU: 0 **loss:** -
Prompt:

```
A continuación tienes datos meteorológicos de una estación,
en formato variableHora=valor.
Escribe un resumen meteorológico en español para el cliente
usando estos datos.
Datos: {prompt_parts}
Resumen:
```

Id 3: **Descripció:** Instruction prompt + Pre-trained model (Prompt variant).
BS: - **GRAD:** - **EPOCHS_MAX:** - **EPOCHS_STOP:** -
r: - α: - **trainable params (%):** -
GPU Mem usage: 3630MiB
ROUGE-1: 0.046 **ROUGE-2:** 0.0063 **ROUGE-L:** 0.0301 **BERTScore:** 0.2136 **BLEU:** 0.01 **loss:** -
Prompt:

```
A partir de los siguientes datos meteorológicos, escribe
un informe en español
como si lo explicaras a alguien que planea salir al aire
libre. Sé claro, natural y realista.
Datos disponibles: {prompt_parts}
Informe: TEXT=>
```

Id 4: **Descripció:** Instruction prompt + Pre-trained model (Prompt variant).
BS: - **GRAD:** - **EPOCHS_MAX:** - **EPOCHS_STOP:** -

r: - α: - **trainable params (%):** -
GPU Mem usage: 3630MiB
ROUGE-1: 0.0266 **ROUGE-2:** 0.0038 **ROUGE-L:** 0.0176 **BERTScore:** 0.1276 **BLEU:** 0.0006 **loss:** -
Prompt:

```
Dispones de los siguientes datos meteorológicos, en formato
variableHora=valor: {prompt_parts}
Quiero que me hagas con esos datos un resumen del día meteorológico,
antes de ese texto, debe hacer un TEXT=>
```

Id 5: **Descripció:** Instruction prompt + Pre-trained model (Prompt variant amb 'TEXT=>').
BS: - **GRAD:** - **EPOCHS_MAX:** - **EPOCHS_STOP:** -
r: - α: - **trainable params (%):** -
GPU Mem usage: 3630MiB
ROUGE-1: 0.1055 **ROUGE-2:** 0.0145 **ROUGE-L:** 0.0693 **BERTScore:** 0.4933 **BLEU:** 0.0023 **loss:** -
Prompt:

```
A continuación tienes datos meteorológicos de una estación,
en formato variableHora=valor.
Escribe un resumen meteorológico en español para el cliente
usando estos datos.
Datos: {prompt_parts}
Resumen: TEXT=>
```

Id 6: **Descripció:** Raw prompt + Fine-tuned model (Troposferica).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 10 **EPOCHS_STOP:** 10
r: 16 α: 32 **trainable params (%):** 0.2815
GPU Mem usage: 23046MiB
ROUGE-1: 0.3229 **ROUGE-2:** 0.0568 **ROUGE-L:** 0.1830 **BERTScore:** 0.7134 **BLEU:** 0.0133 **loss:** 0.54
Prompt: `RAW`

Id 7: **Descripció:** Raw prompt + Fine-tuned model (Troposferica, 8 èpoques).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 8 **EPOCHS_STOP:** 8
r: 16 α: 32 **trainable params (%):** 0.2815
GPU Mem usage: 23046MiB
ROUGE-1: 0.3225 **ROUGE-2:** 0.0564 **ROUGE-L:** 0.1785 **BERTScore:** 0.7089 **BLEU:** 0.0121 **loss:** 0.51
Prompt: `RAW`

Id 8: **Descripció:** Raw prompt + Fine-tuned model (Troposferica, 100 èpoques).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 100 **EPOCHS_STOP:** 100
r: 16 α: 32 **trainable params (%):** 0.2815
GPU Mem usage: 23046MiB

ROUGE-1: 0.3277 **ROUGE-2:** 0.0688 **ROUGE-L:** 0.1993 **BERTScore:** 0.7232 **BLEU:** 0.0191 **loss:** 0.382
Prompt: `RAW`

Id 9: **Descripció:** Raw prompt + Fine-tuned model (Troposferica, $r = 8, \alpha = 16$, 10 èpoques).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 10 **EPOCHS_STOP:** 10
r: 8 α**:** 16 **trainable params (%):** 0.1410
GPU Mem usage: 22986MiB
ROUGE-1: 0.3177 **ROUGE-2:** 0.0532 **ROUGE-L:** 0.1762 **BERTScore:** 0.7061 **BLEU:** 0.0113 **loss:** 0.5656
Prompt: `RAW`

Id 10: **Descripció:** Raw prompt + Fine-tuned model (Troposferica, $r = 4, \alpha = 8$, 10 èpoques).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 10 **EPOCHS_STOP:** 10
r: 4 α**:** 8 **trainable params (%):** 0.0705
GPU Mem usage: 22676MiB
ROUGE-1: 0.3075 **ROUGE-2:** 0.0478 **ROUGE-L:** 0.1677 **BERTScore:** 0.6977 **BLEU:** 0.0102 **loss:** 0.620
Prompt: `RAW`

Id 11: **Descripció:** Instruction prompt + Fine-tuned model (Troposferica, $r = 16, \alpha = 32$, 10 èpoques).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 10 **EPOCHS_STOP:** 10
r: 16 α**:** 32 **trainable params (%):** 0.2815
GPU Mem usage: 23046MiB
ROUGE-1: 0.3041 **ROUGE-2:** 0.0534 **ROUGE-L:** 0.1748 **BERTScore:** 0.7122 **BLEU:** 0.0118 **loss:** 0.5027
Prompt:

```
A continuación tienes datos meteorológicos de una estación,
en formato variableHora=valor.
Escribe un resumen meteorológico en español para el cliente
usando estos datos.
Datos: {prompt_parts}
Resumen:
```

Id 12: **Descripció:** Instruction prompt + Fine-tuned model (Troposferica, $r = 16, \alpha = 32$, 10 èpoques, variant).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 10 **EPOCHS_STOP:** 10
r: 16 α**:** 32 **trainable params (%):** 0.2815
GPU Mem usage: 23046MiB
ROUGE-1: 0.2929 **ROUGE-2:** 0.0532 **ROUGE-L:** 0.1733 **BERTScore:** 0.7134 **BLEU:** 0.0107 **loss:** 0.5005
Prompt:

```
A partir de los siguientes datos meteorológicos, escribe
un informe en español
```

```
como si lo explicaras a alguien que planea salir al aire
libre.  Sé claro, natural y realista.
Datos disponibles:  {prompt_parts}
Informe:  TEXT=>
```

Id 13: **Descripció:** Instruction prompt + Fine-tuned model (Troposferica, $r = 16, \alpha = 32$, 10 èpoques, variant).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 10 **EPOCHS_STOP:** 10
r: 16 α**:** 32 **trainable params (%):** 0.2815
GPU Mem usage: 23046MiB
ROUGE-1: 0.2763 **ROUGE-2:** 0.0480 **ROUGE-L:** 0.1631 **BERTScore:** 0.6945 **BLEU:** 0.0098 **loss:** 0.5008
Prompt:

```
Dispones de los siguientes datos meteorológicos, en formato
variableHora=valor:
{prompt_parts} Quiero que me hagas con esos datos un resumen
del día meteorológico,
antes de ese texto, debe hacer un TEXT=>
```

Id 14: **Descripció:** Instruction prompt + Fine-tuned model (Troposferica, $r = 16, \alpha = 32$, 100 èpoques, variant).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 100 **EPOCHS_STOP:** 100
r: 16 α**:** 32 **trainable params (%):** 0.2815
GPU Mem usage: 23046MiB
ROUGE-1: 0.3050 **ROUGE-2:** 0.0678 **ROUGE-L:** 0.1922 **BERTScore:** 0.7226 **BLEU:** 0.0168 **loss:** 0.3602
Prompt:

```
A partir de los siguientes datos meteorológicos, escribe
un informe en español
como si lo explicaras a alguien que planea salir al aire
libre.  Sé claro, natural y realista.
Datos disponibles:  {prompt_parts}
Informe:  TEXT=>
```

Id extra1:
Descripció: Raw prompt + Fine-tuned model (Troposferica, $r = 32, \alpha = 64$, 10 èpoques).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 10 **EPOCHS_STOP:** 10
r: 32 α**:** 64 **trainable params (%):** 0.5615
GPU Mem usage: 23126MiB
ROUGE-1: 0.3170 **ROUGE-2:** 0.0585 **ROUGE-L:** 0.1863 **BERTScore:** 0.7181 **BLEU:** 0.0140 **loss:** 0.5042
Prompt: `RAW`

Id extra2:
Descripció: Raw prompt + Fine-tuned model (Troposferica, $r = 128, \alpha =$

256, 10 èpoques).
BS: 8 **GRAD:** 16 **EPOCHS_MAX:** 10 **EPOCHS_STOP:** 10
r: 128 α**:** 256 **trainable params (%):** 2.2086
GPU Mem usage: 23770MiB
ROUGE-1: 0.3206 **ROUGE-2:** 0.0656 **ROUGE-L:** 0.1940 **BERTScore:** 0.7222 **BLEU:** 0.0161 **loss:** 0.4482
Prompt: `RAW`

B Pseudocodi complet dels scripts

A continuació es presenta el pseudocodi detallat de cadascun dels tres scripts utilitzats en aquest projecte. S'han inclòs comentaris i passos clars per tal que el lector pugui seguir tota la lògica de funcionament. Aquest annex es pot consultar en acabar la memòria i serveix per documentar completament l'estructura de cada procés.

B.1 Pseudocodi de `train_weather_qwen.py`

```
BEGIN
# ------------------------------------------------
# 1. Definició de constants i creació de directoris
# ------------------------------------------------
MODEL_ID       := "Kukedlc/Qwen2-1.5B-Spanish-1.0"
CSV_PATH       := "./DATOS_FINALES/dataset_train.csv"
OUTPUT_DIR     := "./qwen2-finetuned-meteorologia-lora-v5"
EVAL_ARTIFACTS := OUTPUT_DIR + "/evaluation_artifacts"
EPOCHS         := 3
BATCH_PER_GPU  := 2
GRAD_ACCUM     := 8
LEARNING_RATE  := 2e-5
MAX_SEQ_LEN    := 1280
VALID_RATIO    := 0.1
SEED           := 42

// Crear directoris si no existeixen
crear_directori(OUTPUT_DIR)
crear_directori(EVAL_ARTIFACTS)

// Fixar llavors per reproducibilitat
set_llavor_torch(SEED)
set_llavor_numpy(SEED)

# ------------------------------------------------
# 2. Lectura i preprocessament de dades
```

```
# ------------------------------------------------
df := llegir_csv(CSV_PATH,
separador=";",
na_values="&",
quoting="QUOTE_NONE",
on_bad_lines="warn")

SI nombre_columnes(df) != 67:
LLENÇAR("Error: el CSV ha de contenir 67 columnes")

df := eliminar_fils_duplicades(df)

// Dividir en conjunts d'entrenament i validació
(train_df, val_df) := train_test_split(
df,
test_size=VALID_RATIO,
random_state=SEED
)

input_cols := llistar_columnes(df)[0:66]  // Primeres 66 variables
label_col  := llistar_columnes(df)[66]   // Columna 67 (text objectiu)

FUNC build_prompt(fila):
// Construeix un prompt concatenant les 66 variables i el text de sortida
seq := ""
PER cada c EN input_cols:
seq := seq + f"{c}={fila[c]}, "
// Retirar la última ", " si existeix
SI seq acaba_amb(", "):
seq := seq[0:len(seq)-2]
RETURN seq + " TEXT=>" + fila[label_col] + "<|endoftext|>"

// Aplicar la funció build_prompt a cada fila
train_texts := []
PER cada fila EN train_df:
afegir(train_texts, build_prompt(fila))

val_texts := []
PER cada fila EN val_df:
afegir(val_texts, build_prompt(fila))

// Convertir a datasets de HuggingFace
train_dataset := Dataset.from_dict({"text": train_texts})
val_dataset   := Dataset.from_dict({"text": val_texts})

# ------------------------------------------------
```

```
# 3. Tokenització
# ------------------------------------------------
tokenizer := AutoTokenizer.from_pretrained(MODEL_ID)
tokenizer.pad_token := tokenizer.eos_token

FUNC tokenize_fn(batch):
// Truncament i padding a longitud fixa
RETURN tokenizer(
batch["text"],
truncation=True,
padding="max_length",
max_length=MAX_SEQ_LEN
)

train_tok_ds := train_dataset.map(
tokenize_fn,
batched=True,
remove_columns=["text"]
)

val_tok_ds := val_dataset.map(
tokenize_fn,
batched=True,
remove_columns=["text"]
)

# ------------------------------------------------
# 4. Configuració del data collator
# ------------------------------------------------
data_collator := DataCollatorForLanguageModeling(
tokenizer=tokenizer,
mlm=False
)

# ------------------------------------------------
# 5. Carrega del model preentrenat i preparació per a LoRA
# ------------------------------------------------
bnb_config := BitsAndBytesConfig(
load_in_8bit=True,
llm_int8_threshold=6.0
)

model := AutoModelForCausalLM.from_pretrained(
MODEL_ID,
quantization_config=bnb_config,
device_map="auto",
```

```
torch_dtype=torch.float16
)

// Prepara el model per entrenament de k-bit
model := prepare_model_for_kbit_training(model)

# ------------------------------------------------
# 6. Configuració de LoRA (Low-Rank Adaptation)
# ------------------------------------------------
lora_config := LoraConfig(
r=16,
lora_alpha=32,
target_modules=["q_proj","k_proj","v_proj","o_proj"],
lora_dropout=0.1,
bias="none",
task_type="CAUSAL_LM"
)

// Inserir adaptadors LoRA al model
model := get_peft_model(model, lora_config)

model.enable_gradient_checkpointing()
model.config.use_cache := False

# ------------------------------------------------
# 7. Definició d'arguments d'entrenament (TrainingArguments)
# ------------------------------------------------
training_args := TrainingArguments(
output_dir=OUTPUT_DIR,
num_train_epochs=EPOCHS,
per_device_train_batch_size=BATCH_PER_GPU,
per_device_eval_batch_size=BATCH_PER_GPU,
gradient_accumulation_steps=GRAD_ACCUM,
learning_rate=LEARNING_RATE,
fp16=True,
logging_steps=50,
eval_strategy="steps",
eval_steps=100,
save_strategy="steps",
save_steps=200,
save_total_limit=3,
load_best_model_at_end=True,
metric_for_best_model="eval_loss",
greater_is_better=False,
optim="adamw_torch",
lr_scheduler_type="cosine",
```

```
warmup_ratio=0.1,
weight_decay=0.01,
report_to="none",
seed=SEED
)

# ------------------------------------------------
# 8. Callback per desar mètriques d'avaluació
# ------------------------------------------------
CLASSE SaveEvaluationMetricsCallback(TrainerCallback):
CONSTRUCTOR(eval_dir):
self.eval_dir := eval_dir
self.eval_count := 0

MÈTODE on_evaluate(args, state, control, metrics):
SI state.is_world_process_zero:
self.eval_count := self.eval_count + 1
step := state.global_step
file_path := f"{self.eval_dir}/eval_metrics_step_{step}.json"

serial_metrics := {}
PER cada clau, valor EN metrics.items():
SI valor és numpy o PyTorch tensor:
serial_metrics[clau] := convertir_a_float_o_llista(valor)
ALTRAMENT:
serial_metrics[clau] := valor

escriure_json(file_path, serial_metrics)
imprimir(f"Mètriques guardades a: {file_path}")

save_callback := SaveEvaluationMetricsCallback(EVAL_ARTIFACTS)

callbacks_list := [
EarlyStoppingCallback(early_stopping_patience=3),
save_callback
]

# ------------------------------------------------
# 9. Entrenament amb Trainer
# ------------------------------------------------
trainer := Trainer(
model=model,
args=training_args,
train_dataset=train_tok_ds,
eval_dataset=val_tok_ds,
data_collator=data_collator,
```

```
callbacks=callbacks_list
)

imprimir("Iniciant entrenament...")
trainer.train()

# ------------------------------------------------
# 10. Post-entrenament: gràfica de pèrdua i desament del model
# ------------------------------------------------
log_history := trainer.state.log_history
train_steps := []
train_losses := []
eval_steps := []
eval_losses := []

PER cada log EN log_history:
SI "loss" EN log i "step" EN log:
afegir(train_steps, log["step"])
afegir(train_losses, log["loss"])
SI "eval_loss" EN log i "step" EN log:
afegir(eval_steps, log["step"])
afegir(eval_losses, log["eval_loss"])

SI train_losses NO ÉS BUIDA:
plotar(train_steps, train_losses, label="Training Loss", alpha=0.8)
plotar(eval_steps, eval_losses, label="Validation Loss",
marker="o", linestyle="--")
etiquetar_eixos("Steps", "Loss")
definir_titol("Training i Validation Loss")
llegenda()
quadrícula(True)
plot_path := f"{EVAL_ARTIFACTS}/loss_plot.png"
desar_plot(plot_path)
imprimir(f"Gràfica desada a: {plot_path}")
ALTRAMENT:
imprimir("No s'han trobat dades de pèrdua per fer gràfica")

# Desar model i tokenitzador
imprimir("Guardant model i tokenitzador a les carpetes corresponents...")
trainer.save_model(OUTPUT_DIR)
tokenizer.save_pretrained(OUTPUT_DIR)

imprimir("Fine-tuning completat amb èxit.")
END
```

B.2 Pseudocodi de evaluate_baseline.py

```
BEGIN
# ----------------------------------------------------
# 1. Definició de constants i càrrega de biblioteques
# ----------------------------------------------------
MODEL_ID       := "Kukedlc/Qwen2-1.5B-Spanish-1.0"
VALIDATION_CSV := "./DATOS_FINALES/dataset_validation.csv"
MAX_SEQ_LEN    := 1280

# ----------------------------------------------------
# 2. Funció per carregar model base i tokenitzador
# ----------------------------------------------------
FUNC load_model_and_tokenizer():
tokenizer := AutoTokenizer.from_pretrained(MODEL_ID)
tokenizer.pad_token := tokenizer.eos_token

model := AutoModelForCausalLM.from_pretrained(
MODEL_ID,
device_map="auto",
torch_dtype=torch.float16
)
model.eval()
RETURN model, tokenizer

# ----------------------------------------------------
# 3. Funció per extreure la predicció de la cadena generada
# ----------------------------------------------------
FUNC extract_prediction(text):
SI "TEXT=>" NO ÉS substring DE text:
imprimir("Advertència: no s'ha trobat l'indicador TEXT=>")
RETURN ""
SPLIT_PARTS := dividir(text, "TEXT=>")
RETURN SPLIT_PARTS[1].strip()

# ----------------------------------------------------
# 4. Funció per avaluar el model base sobre validació
# ----------------------------------------------------
FUNC evaluate_baseline(model, tokenizer, df_val):
predictions := []
actuals := []
imprimir("\nExemples de prediccions vs. valors reals:")
imprimir("------------------------------------------------")

PER idx, fila EN enumerate(df_val):
// Preparar el prompt en format ‘‘variableHora=valor’’
```

```
input_cols := columnes(0..65) de df_val
label_col  := columna(66) de df_val

prompt_parts := unir_amb(", ", [f"{c}={fila[c]}" per c en input_cols])
prompt := (
"A continuació tens dades meteorològiques de una estació,\n"
+ "en format variableHora=valor.\n"
+ "Escriu un resum meteorològic en espanyol per al client amb aquests dades.\n"
+ "Datos: " + prompt_parts + "\nResumen:"
)

// Tokenitzar i generar amb el model
inputs := tokenizer(
prompt,
return_tensors="pt",
truncation=True,
max_length=MAX_SEQ_LEN
)
mou_inputs_a_device(inputs, model.device)

AMB torch.no_grad():
outputs := model.generate(
**inputs,
max_length=MAX_SEQ_LEN,
num_return_sequences=1,
temperature=0.7,
do_sample=True,
pad_token_id=tokenizer.eos_token_id
)

generated_text := tokenizer.decode(outputs[0], skip_special_tokens=True)
pred := extract_prediction(generated_text)
act  := str(fila[label_col])

afegir(predictions, pred)
afegir(actuals, act)

SI idx < 3:
imprimir(f"\nExemple {idx+1}:")
imprimir(f"Prompt: {prompt}")
imprimir(f"Text generat: {generated_text}")
imprimir(f"Predicció extreta: {pred}")
imprimir(f"Valor real: {act}")
imprimir("-------------------------------------------------")

RETURN predictions, actuals
```

```
# ------------------------------------------------
# 5. Funció per calcular mètriques automàtiques
# ------------------------------------------------
FUNC calculate_metrics(predictions, actuals):
// a) Precisió lèxica, recall i F1 a partir de conjunts de paraules
precisions := []
recalls := []
f1s := []
PER pred, act EN zip(predictions, actuals):
set_pred := conjunt(pred.split())
set_act  := conjunt(act.split())
SI |set_act| == 0: CONTINUE
// Calcular precisió
SI |set_pred| > 0:
p := |set_pred + set_act| / |set_pred|
ALTRAMENT:
p := 0
afegir(precisions, p)
// Calcular recall
r := |set_pred + set_act| / |set_act|
afegir(recalls, r)
// Calcular F1 lèxic
SI p + r > 0:
f := 2 * p * r / (p + r)
ALTRAMENT:
f := 0
afegir(f1s, f)

// b) BERTScore (precisió, recall, F1)
(P, R, F) := bert_score(
predictions,
actuals,
lang="es",
device="cuda_si_disponible"
)
bert_scores := {
 "precision": P.mean().item(),
 "recall":    R.mean().item(),
 "f1":        F.mean().item()
}

// c) BLEU Score amb suavitzat
bleu_scores := []
smoothie := SmoothingFunction().method1
PER pred, act EN zip(predictions, actuals):
```

```
tokens_pred := pred.split()
tokens_act  := act.split()
TRY:
bleu := sentence_bleu(
[tokens_act],
tokens_pred,
smoothing_function=smoothie
)
EXCEPT:
bleu := 0.0
afegir(bleu_scores, bleu)

// d) ROUGE Metrics (rouge1, rouge2, rougeL)
scorer := rouge_scorer.RougeScorer(
["rouge1", "rouge2", "rougeL"],
use_stemmer=True
)
rec_rouge := {
 "rouge1": {"precision": [], "recall": [], "fmeasure": []},
 "rouge2": {"precision": [], "recall": [], "fmeasure": []},
 "rougeL": {"precision": [], "recall": [], "fmeasure": []}
}
PER pred, act EN zip(predictions, actuals):
scores := scorer.score(act, pred)
PER metric EN ["rouge1", "rouge2", "rougeL"]:
afegir(rec_rouge[metric]["precision"], scores[metric].precision)
afegir(rec_rouge[metric]["recall"],    scores[metric].recall)
afegir(rec_rouge[metric]["fmeasure"],  scores[metric].fmeasure)

// e) MoverScore (TF-IDF + similitud cosí)
mover_scores := []
PER ref, hyp EN zip(actuals, predictions):
tfidf_mat := TfidfVectorizer().fit_transform([ref, hyp])
ref_vec := tfidf_mat[0].toarray()[0]
hyp_vec := tfidf_mat[1].toarray()[0]
sim := 1 - cosine(ref_vec, hyp_vec)
afegir(mover_scores, sim)
mover_mean := mitjana(mover_scores)

RETURN {
 "precision_lexica": mitjana(precisions),
 "recall_lexica":    mitjana(recalls),
 "f1_lexic":         mitjana(f1s),
 "bert_score":       bert_scores,
 "bleu":             mitjana(bleu_scores),
 "rouge1": {
```

```
  "precision": mitjana(rec_rouge["rouge1"]["precision"]),
  "recall":    mitjana(rec_rouge["rouge1"]["recall"]),
  "fmeasure":  mitjana(rec_rouge["rouge1"]["fmeasure"])
 },
 "rouge2": {
  "precision": mitjana(rec_rouge["rouge2"]["precision"]),
  "recall":    mitjana(rec_rouge["rouge2"]["recall"]),
  "fmeasure":  mitjana(rec_rouge["rouge2"]["fmeasure"])
 },
 "rougeL": {
  "precision": mitjana(rec_rouge["rougeL"]["precision"]),
  "recall":    mitjana(rec_rouge["rougeL"]["recall"]),
  "fmeasure":  mitjana(rec_rouge["rougeL"]["fmeasure"])
 },
 "mover_score": mover_mean
}

# ------------------------------------------------
# 6. Funció principal i desament de resultats
# ------------------------------------------------
FUNC main():
df_val := llegir_csv(VALIDATION_CSV,
separador=";",
na_values="&",
quoting="QUOTE_NONE",
encoding="latin1")
SI nombre_columnes(df_val) != 67:
LLENÇAR("Error: el CSV ha de tenir 67 columnes")

(model, tokenizer) := load_model_and_tokenizer()
(predictions, actuals) := evaluate_baseline(model, tokenizer, df_val)

metrics := calculate_metrics(predictions, actuals)

imprimir("\nMètriques de baseline:")
imprimir(metrics)

// Desar prediccions i mètriques
results_df := DataFrame({
 "prediction": predictions,
 "actual": actuals
})
guardar_csv("baseline_validation_results.csv", results_df)

escriure_json("baseline_validation_metrics.json", metrics)
```

```
SI __name__ == "__main__":
main()
END
```

B.3 Pseudocodi de `evaluate_model.py`

```
BEGIN
# ------------------------------------------------
# 1. Definició de constants i càrrega de biblioteques
# ------------------------------------------------
MODEL_ID       := "Kukedlc/Qwen2-1.5B-Spanish-1.0"
ADAPTER_PATH   := "./qwen2-finetuned-meteorologia-lora-v5"
VALIDATION_CSV := "./DATOS_FINALES/dataset_validation.csv"
MAX_SEQ_LEN    := 1280

# ------------------------------------------------
# 2. Funció per carregar model preentrenat i adaptadors LoRA
# ------------------------------------------------
FUNC load_model_and_tokenizer():
tokenizer := AutoTokenizer.from_pretrained(MODEL_ID)
tokenizer.pad_token := tokenizer.eos_token

model_base := AutoModelForCausalLM.from_pretrained(
MODEL_ID,
device_map="auto",
torch_dtype=torch.float16
)
// Carregar adaptadors LoRA ja entrenats
model := PeftModel.from_pretrained(model_base, ADAPTER_PATH)
model.eval()
RETURN model, tokenizer

# ------------------------------------------------
# 3. Funció per extreure la predicció de la cadena generada
# ------------------------------------------------
FUNC extract_prediction(text):
SI "TEXT=>" NO ÉS substring DE text:
imprimir("Advertència: no s'ha trobat l'indicador TEXT=>")
RETURN ""
SPLIT_PARTS := dividir(text, "TEXT=>")
RETURN SPLIT_PARTS[1].strip()

# ------------------------------------------------
# 4. Funció per avaluar el model LoRA sobre validació
# ------------------------------------------------
```

```
FUNC evaluate_model_loRA(model, tokenizer, df_val):
predictions := []
actuals := []
imprimir("\nExemples de prediccions vs. valors reals:")
imprimir("-----------------------------------------------")

PER idx, fila EN enumerate(df_val):
input_cols := columnes(0..65) de df_val
label_col  := columna(66) de df_val

prompt_parts := unir_amb(" , ", [f"{c}={fila[c]}" per c en input_cols])
prompt := prompt_parts + " TEXT=>"

inputs := tokenizer(
prompt,
return_tensors="pt",
truncation=True,
max_length=MAX_SEQ_LEN
)
mou_inputs_a_device(inputs, model.device)

AMB torch.no_grad():
outputs := model.generate(
**inputs,
max_length=MAX_SEQ_LEN,
num_return_sequences=1,
temperature=0.7,
do_sample=True,
pad_token_id=tokenizer.eos_token_id
)

generated_text := tokenizer.decode(outputs[0], skip_special_tokens=True)
pred := extract_prediction(generated_text)
act  := str(fila[label_col])

afegir(predictions, pred)
afegir(actuals, act)

SI idx < 3:
imprimir(f"\nExemple {idx+1}:")
imprimir(f"Prompt: {prompt}")
imprimir(f"Text generat: {generated_text}")
imprimir(f"Predicció extreta: {pred}")
imprimir(f"Valor real: {act}")
imprimir("-----------------------------------------------")
```

```
RETURN predictions, actuals

# ------------------------------------------------
# 5. Funció per calcular mètriques automàtiques
# ------------------------------------------------
FUNC calculate_metrics(predictions, actuals):
// Igual que en evaluate_baseline.py
precisions := []
recalls := []
f1s := []
PER pred, act EN zip(predictions, actuals):
set_pred := conjunt(pred.split())
set_act  := conjunt(act.split())
SI |set_act| == 0: CONTINUE
SI |set_pred| > 0:
p := |set_pred + set_act| / |set_pred|
ALTRAMENT:
p := 0
afegir(precisions, p)
r := |set_pred + set_act| / |set_act|
afegir(recalls, r)
SI p + r > 0:
f := 2 * p * r / (p + r)
ALTRAMENT:
f := 0
afegir(f1s, f)

(P, R, F) := bert_score(
predictions,
actuals,
lang="es",
device="cuda_si_disponible"
)
bert_scores := {
 "precision": P.mean().item(),
 "recall":    R.mean().item(),
 "f1":        F.mean().item()
}

bleu_scores := []
smoothie := SmoothingFunction().method1
PER pred, act EN zip(predictions, actuals):
tokens_pred := pred.split()
tokens_act  := act.split()
TRY:
b := sentence_bleu([tokens_act], tokens_pred, smoothing_function=smoothie)
```

```
EXCEPT:
b := 0.0
afegir(bleu_scores, b)

scorer := rouge_scorer.RougeScorer(
["rouge1", "rouge2", "rougeL"],
use_stemmer=True
)
rec_rouge := {
 "rouge1": {"precision": [], "recall": [], "fmeasure": []},
 "rouge2": {"precision": [], "recall": [], "fmeasure": []},
 "rougeL": {"precision": [], "recall": [], "fmeasure": []}
}
PER pred, act EN zip(predictions, actuals):
scores := scorer.score(act, pred)
PER metric EN ["rouge1", "rouge2", "rougeL"]:
afegir(rec_rouge[metric]["precision"], scores[metric].precision)
afegir(rec_rouge[metric]["recall"],    scores[metric].recall)
afegir(rec_rouge[metric]["fmeasure"],  scores[metric].fmeasure)

mover_scores := []
PER ref, hyp EN zip(actuals, predictions):
tfidf_mat := TfidfVectorizer().fit_transform([ref, hyp])
ref_vec := tfidf_mat[0].toarray()[0]
hyp_vec := tfidf_mat[1].toarray()[0]
sim := 1 - cosine(ref_vec, hyp_vec)
afegir(mover_scores, sim)
mover_mean := mitjana(mover_scores)

RETURN {
 "precision_lexica": mitjana(precisions),
 "recall_lexica":    mitjana(recalls),
 "f1_lexic":         mitjana(f1s),
 "bert_score":       bert_scores,
 "bleu":             mitjana(bleu_scores),
 "rouge1": {
  "precision": mitjana(rec_rouge["rouge1"]["precision"]),
  "recall":    mitjana(rec_rouge["rouge1"]["recall"]),
  "fmeasure":  mitjana(rec_rouge["rouge1"]["fmeasure"])
 },
 "rouge2": {
  "precision": mitjana(rec_rouge["rouge2"]["precision"]),
  "recall":    mitjana(rec_rouge["rouge2"]["recall"]),
  "fmeasure":  mitjana(rec_rouge["rouge2"]["fmeasure"])
 },
 "rougeL": {
```

```
  "precision": mitjana(rec_rouge["rougeL"]["precision"]),
  "recall":    mitjana(rec_rouge["rougeL"]["recall"]),
  "fmeasure":  mitjana(rec_rouge["rougeL"]["fmeasure"])
 },
 "mover_score": mover_mean
}

# ------------------------------------------------
# 6. Funció principal i desament de resultats
# ------------------------------------------------
FUNC main():
df_val := llegir_csv(VALIDATION_CSV,
separador=";",
na_values="&",
quoting="QUOTE_NONE",
encoding="latin1")
SI nombre_columnes(df_val) != 67:
LLENÇAR("Error: el CSV ha de tenir 67 columnes")

(model, tokenizer) := load_model_and_tokenizer()
(predictions, actuals) := evaluate_model_loRA(model, tokenizer, df_val)

metrics := calculate_metrics(predictions, actuals)

imprimir("\nMètriques del model LoRA:")
imprimir(metrics)

results_df := DataFrame({
 "prediction": predictions,
 "actual": actuals
})
guardar_csv("loRA_validation_results.csv", results_df)

escriure_json("loRA_validation_metrics.json", metrics)

SI __name__ == "__main__":
main()
END
```

C Gràfiques de pèrdua dels models

Figura 9: Gràfica de pèrdua del sub-experiment amb Id = 6.

Figura 10: Gràfica de pèrdua del sub-experiment amb Id = 7.

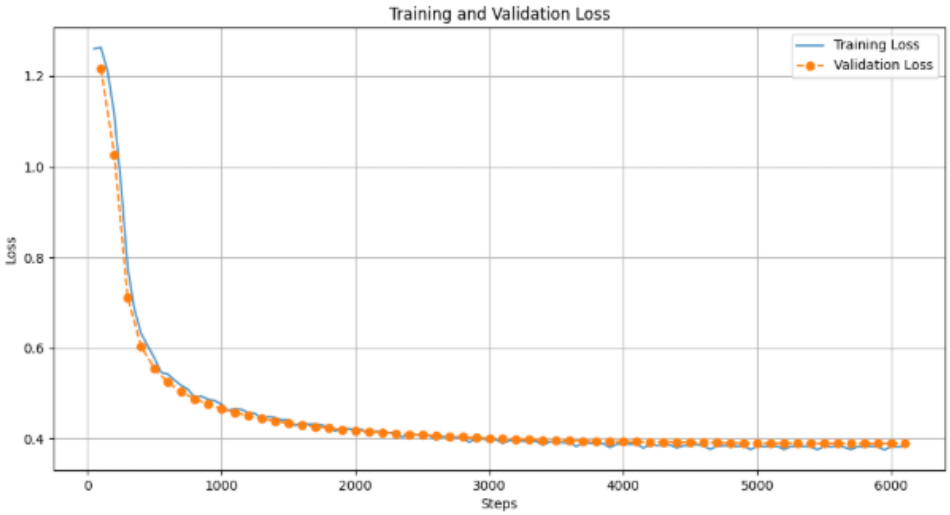

Figura 11: Gràfica de pèrdua del sub-experiment amb Id = 8.

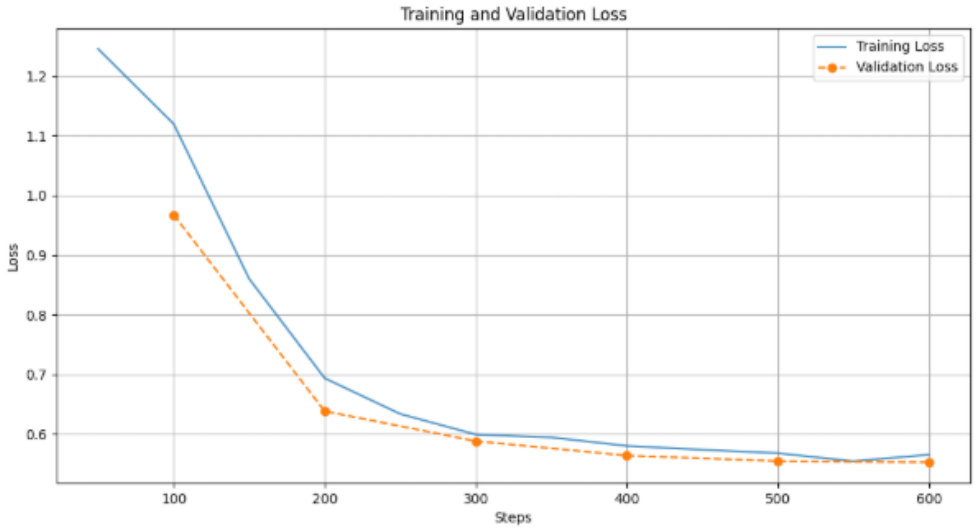

Figura 12: Gràfica de pèrdua del sub-experiment amb Id = 9.

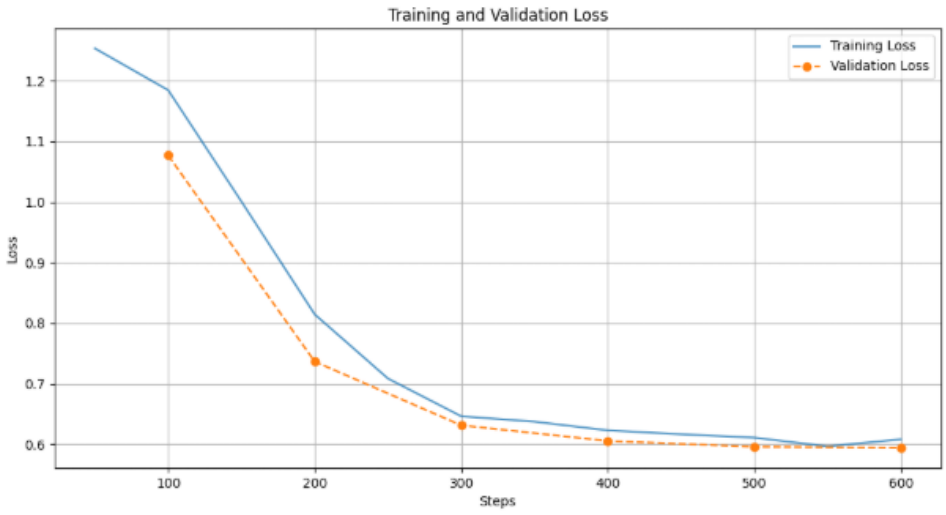

Figura 13: Gràfica de pèrdua del sub-experiment amb Id = 10.

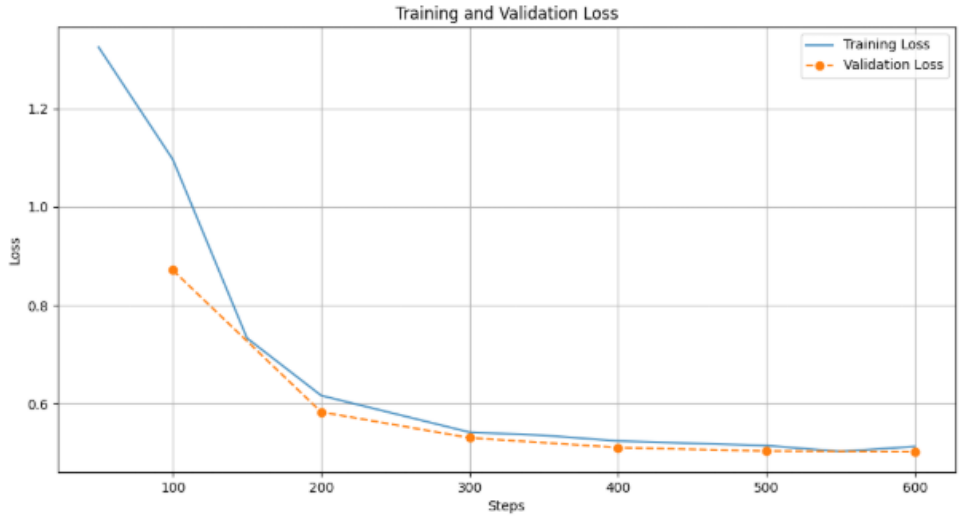

Figura 14: Gràfica de pèrdua del sub-experiment amb Id = 11.

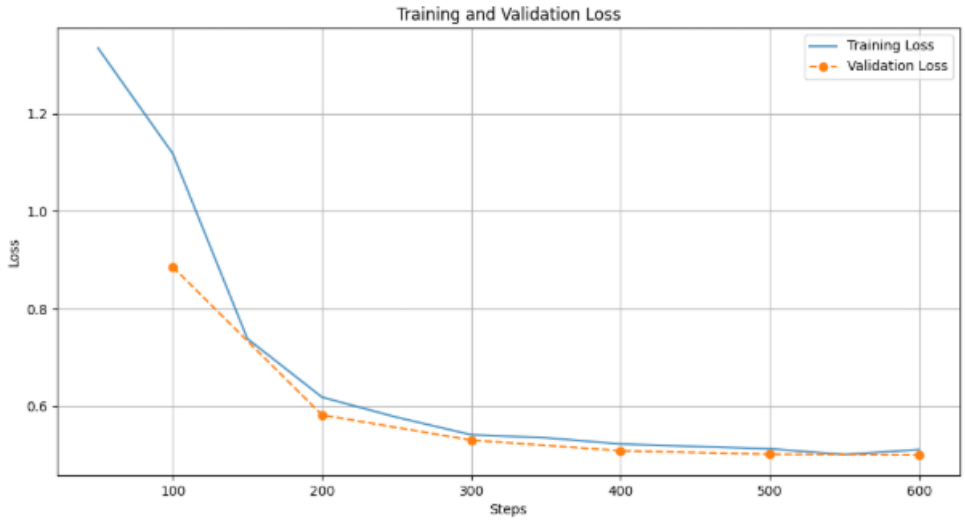

Figura 15: Gràfica de pèrdua del sub-experiment amb Id = 12.

Figura 16: Gràfica de pèrdua del sub-experiment amb Id = 13.

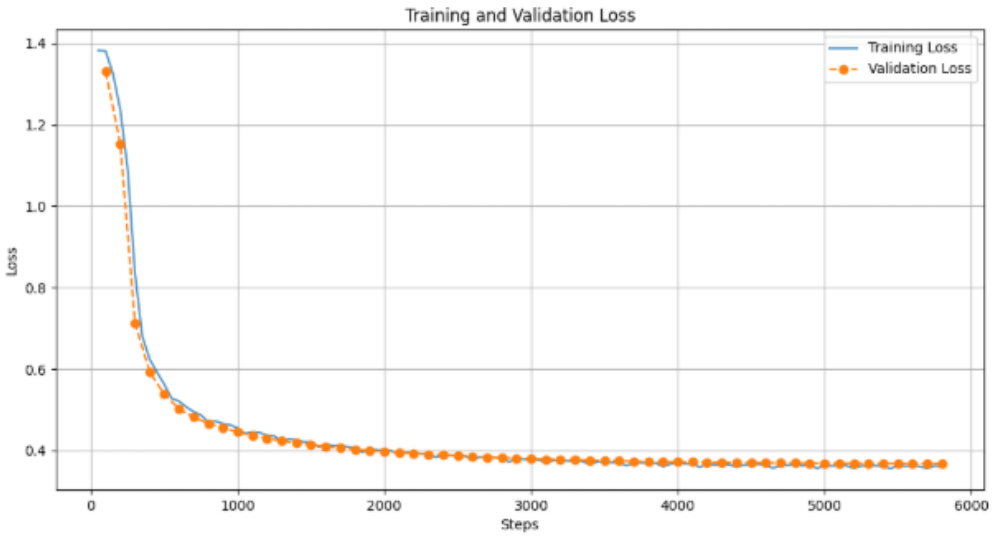

Figura 17: Gràfica de pèrdua del sub-experiment amb Id = 14.

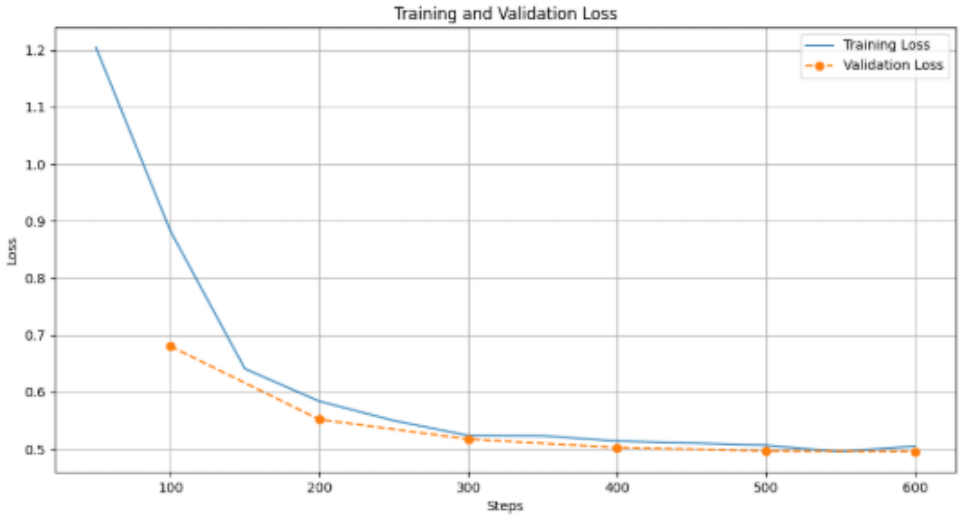

Figura 18: Gràfica de pèrdua del sub-experiment Extra 1.

Figura 19: Gràfica de pèrdua del sub-experiment Extra 2.