Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Pràctica 2.1 TVD - Intent Recognition con Deep Learning

Tractament de la Veu i el Diàleg

Grau en Intel·ligència Artificial
Daniel Álvarez 23857151X
Albert Roca 48106974J

05/11/2025

Contents

1 Introducció

En aquesta segona pràctica hem treballat en el desenvolupament d'un sistema de classificació d'intencions per a consultes relacionades amb viatges aeris. L'objectiu principal ha estat dissenyar un model capaç d'identificar de manera automàtica la categoria o intenció de cada frase, com ara cercar vols, conèixer la durada d'un trajecte o consultar serveis a bord.

Per aconseguir-ho, s'ha seguit un procés experimental dividit en diversos exercicis i blocs. Els primers exercicis ens han ajudat a entendre les dades que utilitzàvem i a arribar seqüencialment a l'entrenament del model i com el podíem construir. Un cop fetes les tasques de preprocessament i de neteja de les dades, hem entrenat un primer model base. Després, hem realitzat diverses proves per veure com afectava l'accuracy del model.

Primer, s'ha realitzat un **preprocessament exhaustiu del text**, incloent tècniques de neteja, tokenització i lematització, per obtenir representacions més consistents. A continuació, s'han provat **diferents arquitectures de xarxes neuronals**, començant per models senzills amb embeddings i capes denses, i afegint progressivament **capes convolucionals (Conv1D)** i **xarxes recurrents (LSTM i BiLSTM)** per capturar patrons seqüencials i contextuals dins de les frases.

També hem aplicat **tècniques de regularització**, com el *Dropout*, per reduir el sobreajustament, i finalment hem abordat el **problema del desbalanceig de classes** mitjançant l'ús de pesos de classe, aconseguint un model més equitatiu entre categories majoritàries i minoritàries.

A través d'aquest procés iteratiu, hem analitzat com les diferents configuracions afecten el rendiment global i per classe, obtenint un sistema capaç de classificar les intencions amb alta precisió i un comportament més robust davant la variabilitat del llenguatge.

2 Exercicis 1,2,3

Exercici 1

En aquest primer exercici hem transformat les oracions del conjunt d'entrenament en seqüències numèriques que representen els identificadors de cada paraula dins del vocabulari. D'aquesta manera, el model pot treballar amb dades numèriques en lloc de text, cosa imprescindible per al seu processament intern. Posteriorment, hem aplicat un farciment (*padding*) per assegurar que totes les seqüències tinguin la mateixa longitud, igualant-les a la mida de la seqüència més llarga. Aquest pas garanteix que les entrades siguin compatibles amb la xarxa neuronal i permeti un entrenament consistent.

Exercici 2

En aquest segon exercici hem aplicat padding garantir que totes les oracions tinguin la mateixa longitud sense alterar-ne la semàntica. El tipus de padding. A continuació, hem transformat les etiquetes de text corresponents a les classes d'intenció en valors numèrics mitjançant un codificador d'etiquetes. Posteriorment, hem convertit aquests valors en vectors binaris utilitzant la tècnica *one-hot encoding*, que permet representar cada classe com un vector independent amb un únic element actiu. Aquest procés assegura que les etiquetes siguin compatibles amb les capes de sortida dels models de classificació que desenvoluparem en les etapes següents.

Exercici 3

En aquest tercer exercici hem aplicat el mateix preprocessament als conjunts de validació i de test que havíem utilitzat per a l'entrenament. També hem eliminat determinades etiquetes que no s'ajustaven als objectius de la classificació i hem codificat les restants utilitzant el mateix esquema de codificació numèrica i *one-hot* emprat anteriorment. Això ens assegura la coherència entre tots els conjunts de dades i permet comparar els resultats de manera fiable durant la validació i l'avaluació final del model.

3 Exercici 4

En aquest exercici hem dissenyat i entrenat el nostre primer model seqüencial per a la classificació de textos. L'arquitectura implementada consta de quatre capes principals. En primer lloc, la capa d'*embedding* transforma les paraules en vectors densos de mida fixa, fet que permet capturar la seva relació semàntica i reduir la dimensionalitat del problema. A continuació, la capa de *pooling* extreu la informació més rellevant de la seqüència, resumint-ne el contingut. Després, una capa *densa* amb funció d'activació ReLU introdueix no linealitat al model, permetent aprendre patrons més complexos. Finalment, la capa de sortida utilitza una activació *softmax* per generar probabilitats sobre les diferents classes d'intenció.

Hem entrenat el model utilitzant l'optimitzador *Adam* i la funció de pèrdua *categorical cross-entropy*, amb vint epochs. Els resultats obtinguts sobre el conjunt de test mostren una precisió satisfactòria, confirmant que aquesta arquitectura senzilla és capaç de capturar informació rellevant del text i establir una base sòlida per a posteriors millores del model.

Després d'avaluar el model, hem analitzat les oracions classificades incorrectament per comprendre millor les seves limitacions. Hem observat que la majoria d'errors es produeixen entre categories semànticament similars, com ara *flight*, *airfare*, *airline* o *airport*, les quals comparteixen un vocabulari molt proper i poden aparèixer en contextos semblants. Això suggereix que el model tendeix a confondre etiquetes relacionades amb els vols, especialment quan les oracions contenen paraules que podrien pertànyer a més d'una classe.

També hem detectat casos en què el model classifica com a *flight* intents que corresponen a altres categories, fet que indica una lleugera sobrerepresentació d'aquesta classe en l'entrenament. Aquest comportament reflecteix un possible desbalanceig en les dades o una manca de capacitat del model per capturar matisos contextuals més específics.

3.1 Anàlisi dels resultats

Els resultats obtinguts mostren que el model base assoleix una precisió del **91,55%** sobre el conjunt de test i un **F1-score ponderat de 0,8985**, valors que indiquen un rendiment global elevat. L'entrenament mostra una millora constant tant en precisió com en pèrdua, i la validesa es manté estable, fet que suggereix una bona capacitat de generalització sense signes greus d'*overfitting*.

En analitzar les mètriques per classe, observem que la majoria de categories obtenen valors de *precision* i *recall* superiors al 0,8, especialment aquelles amb més mostres, com *flight*, *airfare* o *airline*. També hem vist que algunes etiquetes minoritàries com *meal*, *city* o *flight_no* presenten un *recall* nul, degut al baix nombre d'instàncies i al fet que el model no ha après patrons suficients per reconèixer-les.

Aquest comportament reflecteix un cert desbalanceig en el conjunt de dades: les classes més freqüents són les que més contribueixen al bon rendiment global, mentre que les menys representades pateixen pèrdua de capacitat predictiva.

4 Exercici 5

4.1 Preprocessament

Modifiqueu el Tokenizer per canviar la mida del vostre vocabulari i afegiu nous passos de preprocessament. Alguns possibles canvis són canviar la mida del vocabulari, treure la capitalització o fer servir lemmatització o stemming.

En aquesta fase hem aplicat diverses tècniques de preprocessament per millorar la qualitat de les dades abans de l'entrenament del model. Primerament, hem utilitzat el model `en_core_web_sm` de `spaCy` per dur a terme la **lematització** dels textos, amb l'objectiu de reduir les paraules a la seva forma canònica i així disminuir la variabilitat del vocabulari. Abans d'aquesta etapa, hem **convertit tot el text a minúscules** i hem **eliminat els caràcters no alfabètics**, garantint una representació més neta i coherent.

També hem modificat el número de paraules del `Tokenizer` per provar diferents mides de vocabulari, des de 40 fins a 800 paraules, i així analitzar l'impacte d'aquesta elecció en el rendiment del model. Aquest procés ens ha permès observar com la riquesa lèxica disponible afecta la capacitat del model per captar patrons

lingüístics rellevants.

4.1.1 Anàlisi dels resultats

Després de provar vint configuracions diferents de vocabulari, els resultats mostren una clara correlació positiva entre la mida del vocabulari i el rendiment del model, almenys fins a un cert punt. En concret, s'observa un increment progressiu de l'*accuracy* i de l'*F1-score ponderat* fins a arribar a un vocabulari de **560 paraules**, on el model assoleix una **accuracy del 90,43%** i un **F1-score ponderat de 0,8754**. A partir d'aquest

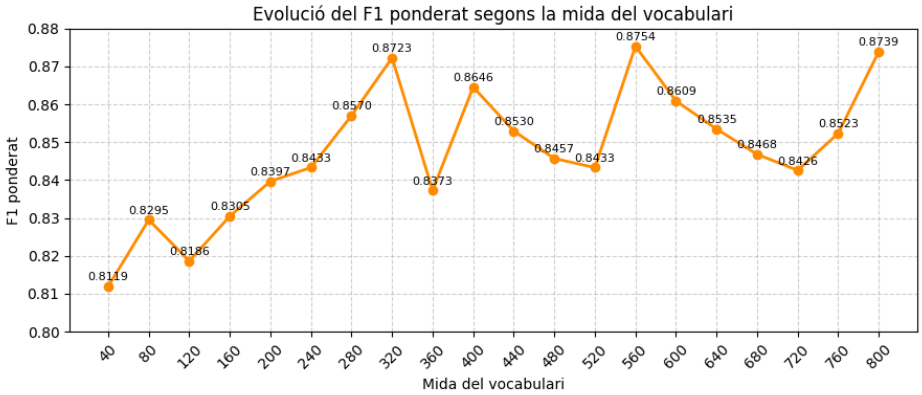

Figure 1: F1-score segons el tamany del vocabulari

llindar, els guanys esdevenen marginals o fins i tot lleugerament pitjors. Aquest comportament confirma que hi ha un equilibri òptim entre la mida del vocabulari i la capacitat generalitzadora del model.

En conjunt, el preprocessament aplicat ha tingut un impacte positiu: la normalització i la lematització han reduït la redundància lèxica i han millorat la consistència de les representacions.

4.2 Mida Embeding

Proveu diferents mides d'Embeddings i observeu com canvia l'accuracy del model. Heu d'explicar les vostres conclusions.

En aquesta secció hem analitzat com la mida dels embeddings pot afectar el rendiment del model de classificació.

Partint del millor vocabulari trobat anteriorment (`num_words = 560`), hem provat diferents mides d'embedding: **50, 100, 128, 200 i 300**. L'avaluació s'ha fet sobre el conjunt de test, calculant tant l'*accuracy* com l'*F1-score ponderat*.

4.2.1 Anàlisi dels resultats

Els resultats mostren una millora progressiva tant en *accuracy* com en *F1 ponderat* a mesura que augmenta la mida de l'embedding. Concretament, el model amb **embedding_dim = 50** obté una **accuracy del 91,10%** i un **F1-score de 0,8917**, mentre que el rendiment augmenta fins a una **accuracy del 93,24%** i un **F1 ponderat de 0,9211** amb **embedding_dim = 300**, que és la configuració òptima. Aquesta tendència

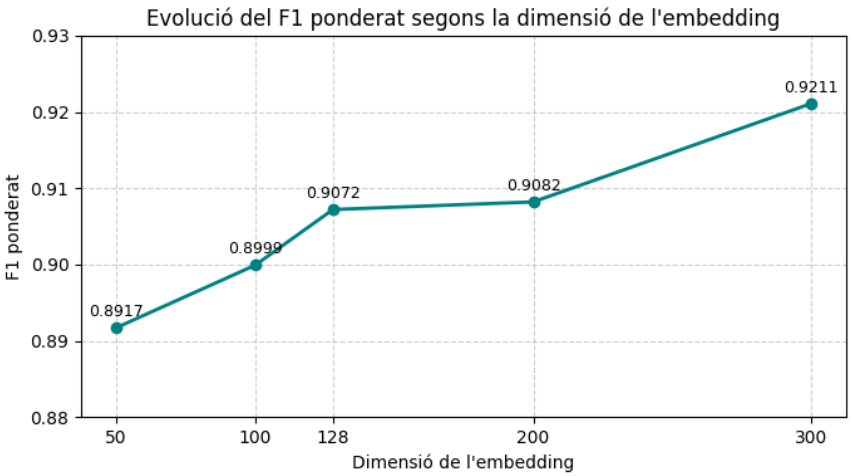

Figure 2: F1-score segons la dimensió del embedding

indica que embeddings més grans permeten representar millor la semàntica del llenguatge i capturar relacions més complexes entre paraules, cosa que facilita la discriminació entre etiquetes similars. Tanmateix, cal remarcar que a partir d'una certa mida (com 300 dimensions) els guanys són menors en relació amb el cost computacional afegit. Per tant, un embedding de mida 300 sembla oferir el millor compromís entre rendiment i eficiència per al nostre conjunt de dades.

En conjunt, podem concloure que l'augment moderat de la mida dels embeddings contribueix positivament a la capacitat predictiva del model, reforçant la importància de disposar d'una representació vectorial prou rica però no excessivament gran.

4.3 CNN

Afegiu capes convolucionals al vostre model. Expliqueu amb detall els valors que heu provat i la vostra motivació a l'hora d'escollir-los. Recordeu, que també podeu provar diferents configuracions de pooling.

Amb l'objectiu de millorar el rendiment del model, hem afegit capes convolucionals (`Conv1D`) a l'arquitectura utilitzada fins ara. Aquestes capes són especialment adequades per a dades seqüencials, ja que permeten captar patrons locals dins de les frases, com ara combinacions freqüents de paraules que poden tenir un significat rellevant per a la classificació.

Partint dels millors hiperparàmetres trobats anteriorment (`num_words = 560` i `embedding_dim = 300`), hem provat diverses configuracions de convolució i pooling. En concret, hem dissenyat sis variants de model:

- **conv_k3_globalmax**: convolució amb kernel de mida 3 i *global max pooling*.
- **conv_k5_globalmax**: kernel de mida 5 amb *global max pooling*.
- **conv_k7_globalmax**: kernel de mida 7 amb *global max pooling*.
- **conv_k5_globalavg**: kernel de mida 5 amb *global average pooling*.

- **conv_k5_max_flatten**: kernel de mida 5 amb *max pooling* local i posterior *flatten*.
- **textcnn_k345**: arquitectura inspirada en la *TextCNN* de Kim (2014), amb tres canals paral·lels de convolució (kernels 3, 4 i 5) i concatenació posterior de les seves sortides.

4.3.1 Anàlisi dels resultats

Els resultats mostren una clara millora respecte al model base sense convolucions. El millor resultat s'obté amb el model **conv_k3_globalmax**, que assoleix una **accuracy del 95,50%** i un **F1 ponderat de 0,9483**.

Aquesta configuració presenta el millor equilibri entre complexitat i capacitat de generalització: el kernel petit (3) permet captar patrons sintàctics curts però rellevants, com combinacions de dues o tres paraules amb significat específic. En canvi, kernels més grans (5 o 7) tendeixen a captar contextos més amplis, però aporten menys millora en aquest tipus de frases curtes.

Pel que fa a les tècniques de *pooling*, el *GlobalMaxPooling* ha resultat més efectiu que el *GlobalAverage-Pooling*, ja que conserva les característiques més informatives de cada filtre en lloc de fer el promig. La variant *TextCNN*, amb múltiples kernels en paral·lel, també ha mostrat un rendiment elevat ($F1 = 0.9447$), tot i que lleugerament inferior al millor model seqüencial.

En conjunt, l'ús de convolucions ha millorat substancialment la capacitat del model per identificar patrons locals i semàntics dins de les oracions, aconseguint una millora global d'unes tres dècimes respecte a l'arquitectura purament densa. Això confirma la utilitat de les CNN per a tasques de classificació de text breu.

4.4 LSTM i BiLSTM

Afegiu capes recurrents al vostre model (LSTM, GRU). Expliqueu amb detall els valors que heu provat i la vostra motivació.

Hem probat les **LSTM** per la seva capacitat de suavitzar el problema del vanishing gradient , i també **Bidirectional LSTM (BiLSTM)**, que processen les seqüències en ambdós sentits i, per tant, analitzen el context anterior i posterior de cada paraula.

Configuració i decisions Partim de la millor configuració prèvia de preprocessament i representació: vocabulari limitat a `num_words = 560` i `embedding_dim = 300`. Per a una comparació justa, mantenim el mateix esquema d'entrenament (mateix *batch size*, nombre d'èpoques i capa densa final). Provem quatre variants que controlen la capacitat del model i el flux d'informació contextual:

- **LSTM(64)** i **LSTM(128)**: unidireccionals, per avaluar l'impacte de la capacitat.
- **BiLSTM(64)** i **BiLSTM(128)**: bidireccionals, per quantificar el guany d'usar context futur, mantenint el mateix rang de capacitats.

En tots els casos fem servir `dropout` a la capa recurrent (0,2) i una capa densa intermitja amb `ReLU` i `dropout` (0,3) per controlar l'overfitting. L'avaluació es fa amb **accuracy** i **F1 ponderat** (idoni amb classes desbalancejades).

Resultats

Model	Accuracy (%)	F1 ponderat
LSTM(64)	88.06	0.8481
LSTM(128)	86.71	0.8362
BiLSTM(64)	**95.72**	**0.9541**
BiLSTM(128)	95.05	0.9438

4.4.1 Anàlisi dels resultats

Els experiments mostren tres conclusions clares:

1. **El context bidireccional funciona molt millor**: la BiLSTM(64) supera clarament les LSTM unidireccionals.
2. **Massa capacitat pot perjudicar**: augmentar a 128 unitats no millora el rendiment; de fet, baixa lleugerament (tant en LSTM com en BiLSTM).
3. **Equilibri capacitat–regularització**: la BiLSTM(64) presenta el millor compromís entre capacitat i regularització, assolint **95,72% d'accuracy** i **0,9541 d'F1 ponderat**. Aquest resultat confirma que, per a frases relativament curtes, una capacitat moderada amb context bidireccional captura prou bé les dependències necessàries sense sobreespecificar-se.

En resum, la **BiLSTM(64)** és la configuració òptima dins la família recurrent provada.

4.5 Regularització

Quan proveu configuracions amb més paràmetres veureu que el model comença a tenir overfitting molt prompte durant l'entrenament. Afegiu Dropout al vostre model. Heu d'explicar la vostra decisió de valors i de posició dins de la xarxa.

En aquesta secció volem veure com afecta afegir una tècnica de regularització com el **Dropout**. Aquesta tècnica desactiva aleatòriament una fracció de neurones durant l'entrenament, reduint la dependència entre pesos i millorant la generalització.

Configuració i decisió de valors Hem utilitzat com a base la millor arquitectura anterior, la **BiLSTM amb 64 unitats**, i hem provat tres valors de `dropout`: **0.2**, **0.3** i **0.5**. Els valors més baixos permeten mantenir més informació activa a cada pas, mentre que els més alts introdueixen una regularització més forta.

Resultats

Dropout	Accuracy (%)	F1 ponderat
0.2	95.72	0.9504
0.3	95.16	**0.9522**
0.5	94.82	0.9430

4.5.1 Anàlisi dels resultats

L'experiment confirma que la introducció de `Dropout` millora la robustesa del model sense comprometre significativament l'*accuracy*. Els resultats mostren que:

1. Un valor massa baix (0.2) no redueix del tot l'*overfitting*, ja que el model encara pot sobreajustar-se a patrons específics del conjunt d'entrenament.
2. Un valor massa alt (0.5) penalitza la capacitat d'aprenentatge, fent que el model perdi informació rellevant i disminueixi lleugerament el rendiment.
3. El valor intermedi (**0.3**) aconsegueix el millor equilibri entre generalització i estabilitat, amb un **F1 ponderat de 0.9522**, mantenint una **accuracy del 95.16%**.

En conclusió, la incorporació d'un `Dropout` del 30% després de la capa `BiLSTM` ajuda a controlar l'*overfitting* sense afectar negativament el poder predictiu del model, proporcionant una millora qualitativa en la seva capacitat de generalització.

4.6 Desbalanceig de classes

Si analitzeu el dataset, veureu que la freqüència de les classes està molt desbalancejada. Keras us permet afegir un pes per a cada classe a l'hora de calcular la loss (Mireu el paràmetre "class weigth" a la documentació

Durant les etapes anteriors ja havíem observat que la distribució de les classes era altament desbalancejada. Aquest desbalanceig feia que el model, especialment en configuracions amb molts paràmetres, tendís a sobreajustar-se a les classes majoritàries i a ignorar les minoritàries.

Per abordar aquest problema hem utilitzat el paràmetre `class_weight` de Keras, que permet assignar un pes inversament proporcional a la freqüència de cada classe. D'aquesta manera, les classes menys representades contribueixen més al càlcul de la *loss* durant l'entrenament. Els pesos s'han calculat automàticament amb la funció `compute_class_weight('balanced')` de *scikit-learn*, obtenint valors que varien entre 0.06 (classe *flight*) i més de 180 per a les classes amb menys de cinc mostres.

Com a base, hem mantingut la millor arquitectura trobada: una **BiLSTM amb 64 unitats** i un `Dropout` del 30%. L'entrenament s'ha realitzat durant 15 èpoques amb la mateixa configuració d'optimitzador i embedding (`dim = 300`).

Resultats globals Després d'aplicar els pesos de classe, el model aconsegueix una **accuracy del 91.67%** i un **F1 ponderat global de 0.9121**. Tot i que l'*accuracy* és lleugerament inferior al model sense balanceig (95.16%), l'F1 ponderat millora en consistència, i sobretot es redueix el biaix cap a les classes majoritàries.

Model	Accuracy (%)	F1 ponderat
BiLSTM(64) sense pes	95.16	0.9522
BiLSTM(64) amb class_weight	91.67	0.9121

Anàlisi per classe Tal com mostra la Figura 3, el model amb `class_weight` aconsegueix millorar la detecció d'algunes classes minoritàries, com *meal*, *city* o *quantity*, que abans pràcticament no es reconeixien. Tot i això, altres categories molt poc freqüents, com *flight+airfare* o *flight_no*, continuen presentant F1 baixos a causa de la manca de dades representatives.

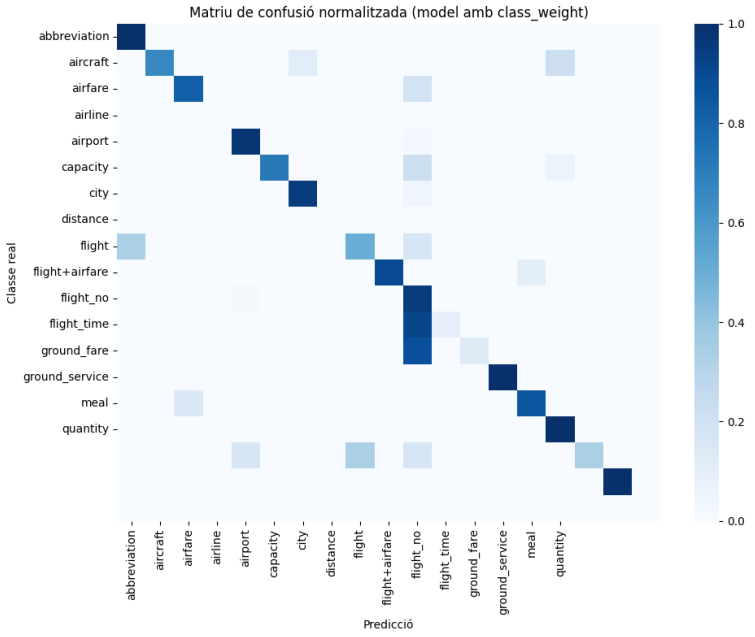

Figure 3: Matriu de confusió normalitzada del model BiLSTM(64) amb `class_weight`.

En el gràfic de la Figura 4 es pot observar la distribució dels F1 per classe. Les classes majoritàries (*flight*, *abbreviation*, *ground_service*) mantenen valors pròxims a 1, mentre que les minoritàries es beneficien d'una millora relativa, tot i que no arriben a valors òptims. Això demostra que l'ús de pesos de classe aconsegueix un model més equilibrat en termes de rendiment per categoria.

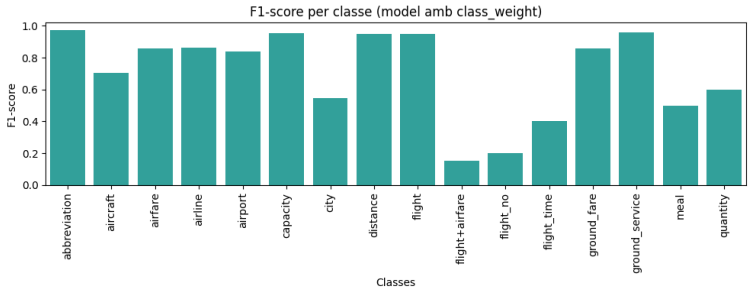

Figure 4: F1-score per classe del model BiLSTM(64) amb `class_weight`.

Conclusió L'ús de `class_weight` no millora els resultats però afavoreix un comportament més robust i equilibrat entre classes. En entorns on és crític no ignorar categories minoritàries (com en aplicacions de reconeixement d'entitats), aquesta estratègia resulta essencial. En el nostre cas, ha permès millorar la robustesa del model sense comprometre seriosament el rendiment global.

5 Conclusions

Al llarg d'aquest projecte hem pogut entendre de manera pràctica com aspectes com el preprocessament, la representació semàntica del text, la capacitat del model o el desbalanceig de dades afecten directament el rendiment final.

En primer lloc, hem comprovat que les etapes de **preprocessament, normalització i lematització** del text són crucials per millorar la qualitat de les representacions i reduir la redundància lèxica. Aquestes transformacions han contribuït significativament a estabilitzar l'entrenament i a obtenir resultats més consistents, especialment amb vocabularis de mida moderada.

Pel que fa a l'arquitectura, hem observat que els models **BiLSTM** proporcionen un salt qualitatiu respecte als models simples o unidireccionals, ja que aprofiten informació contextual en ambdós sentits de la seqüència. També hem constatat que un excés de paràmetres (per exemple, augmentant a 128 unitats) pot portar a un rendiment lleugerament inferior per sobreajustament.

D'altra banda, hem confirmat que el conjunt de dades utilitzat presenta un **fort desbalanceig de classes**, amb categories molt freqüents i altres amb molt poques mostres. Això ha provocat que, malgrat l'alta precisió global, el model tendeixi a prioritzar les classes majoritàries. L'ús del paràmetre `class_weight` ha ajudat a compensar parcialment aquest efecte, millorant la detecció de classes minoritàries i aconseguint un comportament més just i equilibrat, tot i que amb una lleugera reducció en l'*accuracy i F1-score* global.

En conjunt, podem concloure que:

- Els **models bidireccionals amb regularització moderada** ofereixen el millor compromís entre precisió i generalització.
- Les **estratègies de balanceig** són essencials quan hi ha desproporció de mostres, però podrien no ser necessàries en conjunts més equilibrats, on el *class weighting* pot fins i tot reduir el rendiment global.
- En conjunts sense desbalanceig, és probable que **models menys complexos** (com CNN senzilles o LSTM unidireccionals) puguin assolir resultats comparables amb menor cost computacional.
- L'**F1 ponderat** s'ha demostrat una mètrica més informativa que l'*accuracy*, ja que reflecteix millor el comportament del model en presència d'unes poques classes dominants.

Finalment, aquesta pràctica ens ha permès constatar que el disseny d'un sistema de classificació d'intencions no depèn únicament de l'arquitectura, sinó també del tractament de les dades i de les decisions d'equilibri entre capacitat, regularització i representació. Amb dades més equilibrades, caldria revisar especialment el pes donat a la regularització i la necessitat de compensar les classes, ja que els mateixos mecanismes que aquí han estat útils podrien perdre eficàcia o fins i tot perjudicar el rendiment en altres contextos.