Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Sistemes basats en Frame 1, Intent recognition

Tractament de la Veu i el Diàleg

Autors:

Zhihao Chen; Zhiqian Zhou

13 d'octubre de 2024

Índex

1 Introducció

En aquesta pràctica desenvoluparem un sistema de *classificació d'intencions*, una tècnica important en el camp del *Processament del Llenguatge Natural* (NLP). L'objectiu és detectar automàticament la intenció d'un usuari a partir d'una consulta textual, una funcionalitat essencial per a aplicacions com els assistents virtuals o els xatbots. En el document no s'inclouen totes les gràfiques, però es poden trobar dins del Notebook a l'apartat corresponent.

1.1 Conjunt de dades

Treballarem amb un conjunt de dades extret d'interaccions reals en anglès, en el qual cada frase està associada a una de les 22 possibles intencions. Aquest conjunt de dades reflecteix diverses accions que un usuari pot voler realitzar, com ara consultar horaris, fer reserves o cancel·lar serveis. Les dades estan formades per dues parts: les frases d'entrada (consulta de l'usuari) i les etiquetes corresponents a les intencions, que seran les nostres variables objectiu durant l'entrenament del model.

Abans de poder utilitzar aquest conjunt de dades en un model, realitzarem un procés de neteja i transformació per adequar-lo als requisits d'un sistema de classificació basat en xarxes neuronals.

1.2 Desenvolupament de la pràctica

Aquesta pràctica es desenvolupa mitjançant una sèrie d'exercicis pràctics que guiaran l'alumne a través de les diferents fases del projecte. A mesura que es resolguin els exercicis, s'aniran construint diferents components del sistema de classificació d'intencions. Els exercicis inclouran tasques com:

- La inspecció i exploració del conjunt de dades per comprendre millor la seva estructura.
- El preprocessament de les dades, on es realitzarà la tokenització, normalització del text i preparació de les etiquetes.
- El disseny i entrenament d'un model de classificació basat en xarxes neuronals.

Cada exercici està dissenyat per introduir progressivament els conceptes clau necessaris per a la implementació completa del projecte. En completar cada secció, l'alumne desenvoluparà una comprensió profunda del procés de classificació d'intencions, tant a nivell teòric com pràctic.

Finalment, els resultats es resumiran en un informe que inclourà les decisions preses durant el desenvolupament del model, així com una avaluació del seu rendiment i possibles millores.

2 Exercici 1

Abans de començar el primer exercici, es va crear un diccionari utilitzant la partició d'entrenament, limitant el nombre de paraules a 500. Això significa que, en transformar una seqüència de paraules a índexs, l'índex de qualsevol paraula no superarà 499. Aquesta limitació ajuda a eliminar paraules poc freqüents, millorant així el rendiment del model, ja que aquestes paraules aporten poca informació rellevant.

Aquest preprocessament és crucial per al primer exercici, on cal convertir les oracions de la partició d'entrenament en seqüències d'IDs mitjançant el diccionari creat.

Un dels inconvenients d'aquesta transformació és la variabilitat en la longitud de les seqüències, ja que cada paraula té un ID corresponent i cada oració pot tenir un nombre diferent de paraules. Això genera seqüències de mides desiguals, fet que dificulta l'entrenament del model, ja que normalment s'esperen vectors de la mateixa mida. Per solucionar-ho, s'aplica la tècnica de *padding*, que afegeix índexs de valor nul a les seqüències més curtes fins que totes tinguin la mateixa longitud, igualant-les a la mida de la seqüència més llarga.

3 Exercici 2

Com l'ordre de les paraules sí que importa als models que utilitzarem en aquesta pràctica, és aconsellable que el padding estiga al final i no al principi. Això és fàcil de realitzar, afegint un simple argument (padding = "post") en la crida de la funció "pad_sequences".

Un altre preprocessament que demana a l'exercici és codificar les etiquetes de les dades perquè els models d'aprenentatge automàtic les puguin tractar. La solució proposada per l'enunciat és codificar-les en "one-hot vectors" . Una manera senzilla de fer-lo és assignar a cada classe un número, ja que la llibreria de "scikit-learn" proporciona un simple model ("Label_Encoder") que permet realitzar-lo. Cal tenir en compte que cal entrenar el model amb la partició d'entrenament per no provocar confusió posteriorment. Després, un cop tinguts els números assignats, només caldria crear un vector nul per a cada classe i assignar un 1 en la posició del número corresponent.

4 Exercici 3

Un aspecte fonamental del preprocessament és garantir que el mateix tipus de transformacions que es fan sobre la partició de `train` es repliquin en les particions de `validation` i `test`. Això es fa per assegurar la coherència entre les dades utilitzades per entrenar i les que es fan servir per validar i avaluar el model. Sense aquesta consistència, el model podria comportar-se de manera imprevisible quan es troba amb dades preprocessades de manera diferent en les fases de validació o prova.

És important destacar que no cal tornar a entrenar un nou model de preprocessament per a les particions de `validation` i `test`. Això és perquè el preprocessament, com ara la tokenització i la codificació de les etiquetes, es basa en un conjunt de regles o parametritzacions apreses durant la fase d'entrenament (per exemple, el vocabulari generat durant la tokenització). Per tant, és suficient aplicar directament el model de preprocessament entrenat amb la partició de `train` a les altres particions.

A més, cal tenir en compte que les etiquetes presents en les particions de `validation` i `test` han d'estar incloses també a la partició de `train`. Això es deu al fet que, si en les particions de validació o prova apareixen etiquetes que el model no ha vist mai durant l'entrenament, aquest no serà capaç de classificar-les correctament. En conseqüència, qualsevol instància de `validation` o `test` que tingui una etiqueta inexistent a la partició d'entrenament ha de ser eliminada.

5 Exercici 4

Després d'un simple preprocessament i d'una anàlisi general de la base de dades, ja podem començar a definir models d'aprenentatge profund per dur a terme la tasca de classificació d'intencions.

L'estructura de la xarxa neuronal proposada conté 4 capes:

- **Capa de `Embedding`:** Converteix les paraules en vectors numèrics que capturen informació semàntica. Això permet al model reconèixer relacions entre paraules similars, millorant la representació del text.
- **Capa de `Pooling`:** Redueix la dimensionalitat seleccionant la informació més rellevant (per exemple, utilitzant *Max Pooling*), simplificant les dades per evitar el sobreajustament i millorar l'eficiència del model.
- **Capa de `Dense Fully Connected`:** Afegeix complexitat a les característiques extretes per tal de generar representacions més abstractes i preparades per a la classificació.
- **Capa de `Dense` amb funció d'activació `Softmax`:** Transforma les sortides en probabilitats per a cada classe d'intenció, i la classe amb la probabilitat més alta és la predicció final del model.

Aquesta arquitectura permet que el model aprengui representacions de text eficients i útils per a la tasca de classificació, alhora que redueix la complexitat del problema, garantint una predicció robusta i precisa.

Els paràmetres com la mida dels vectors d'`embeddings`, la mida del vocabulari, el nombre de neurones, l'optimitzador, i la taxa d'aprenentatge, entre d'altres, s'han definit seguint els valors predeterminats indicats a l'enunciat. Més endavant, aquests paràmetres es modificaran per intentar millorar el rendiment del model.

Tot i així, amb l'arquitectura actual, el model ja és capaç d'aconseguir una exactitud (`accuracy`) del 82% en la partició de prova després de només 2 èpoques d'entrenament, la qual cosa representa un rendiment força bo.

6 Exercici 5

Els resultats dels experiments amb models de deep learning poden presentar molta variabilitat. Això pot ser degut a la inicialització aleatòria dels pesos, als optimitzadors no determinístics, al dataset desbalancejat que el model troba patrons menys consistents, entre d'altres. Per tal de reduir l'efecte d'aquesta variabilitat i obtenir una mesura més fiable del rendiment del model, s'ha realitzat cada experiment cinc vegades per obtenir resultats.

Tot i repetir cada experiment cinc vegades, els resultats encara presenten variabilitat. Si tinguéssim més capacitat de càlcul, podríem obtenir resultats més fiables fent més iteracions per a cada experiment. No obstant això, a causa de les limitacions de recursos, hem decidit fer només cinc iteracions per tal d'obtenir una avaluació raonablement representativa, tot i que encara observem una certa variabilitat en els resultats finals.

6.1 Estudi de desbalanceig de classes

Amb l'estudi de les classes del dataset s'ha observat un desbalanceig significatiu, amb una concentració elevada de mostres en la classe "flight". Aquest desbalanceig pot afectar negativament el rendiment del model, ja que

tendeix a afavorir les classes majoritàries, mentre que les classes minoritàries reben menys atenció. Per tant, no es pot confiar de l'alt valor d'accuracy rebut.

Per abordar aquest desbalanceig i avaluar el model de manera justa, utilitzarem la mètrica de F1-score amb macro average. Aquesta mètrica calcula l'F1-score per a cada classe individualment i després en fa la mitjana, sense considerar la mida de cada classe. Així, tracta totes les classes per igual, independentment del seu pes en el conjunt de dades.

6.2 Ajustar nombre d'epochs

Com a model inicial tenim que el nombre d'epochs és de 2, nombre de dimensions d'embedding és de 100 i la mida del vocabulari és de 500. Però es preveu que els propers models seran més complexos, per tant, 2 epochs seran insuficients per aprendre. Aleshores, analitzem la tendència de les mètriques d'entrenament i validació en funció del nombre d'epochs i obtenim que, inicialment, amb 5 epochs es podria entrenar models. En casos que el model sigui més complex ja el tornarem a adaptar.

6.3 Balancejat de classes

El dataset té un fort desbalanceig de classes, cosa que afecta el rendiment del model. Per abordar aquest problema, hem aplicat class weights de Keras, que assigna més pes a les classes minoritàries durant l'entrenament. Això ha millorat el F1 macro average del model de 0,3 (sense class weights) a 0,6 aproximadament. Tot i que l'accuracy és similar en ambdós casos (al voltant de 0,9), preferim la mètrica F1 macro average, ja que l'accuracy, com ja s'ha dit, és enganyosa pel nostre problema i F1 macro average és el que reflecteix millor el rendiment equilibrat entre totes les classes. Per tant, els experiments següents es faran utilitzant class weights.

6.4 Preprocessament

Per preprocessar les dades, s'ha creat una classe que inclouen els següents passos:

- Passar tot a minúscules per eliminar ambingüetat
- Eliminar stopwords i signes de puntuació
- Lematització o stemming (depenent de la tècnica seleccionada)
- Creació del vocabulari utilitzant el tokenizer (podem definir la mida del vocabulari)
- Creació de seqüències i padding
- Codificació de les etiquetes

Després del preprocessament, les frases no contenen stopwords ni signes de puntuació, que majoritàriament no aporten informació rellevant per a la tasca de classificació. A més, les paraules han estat lematitzades o passades a stemming, cosa que ajuda a reduir les variacions de les paraules. Aquest procés elimina soroll i permet que el model es concentri en les paraules més importants per a la classificació.

6.5 Provar amb diferents Preprocessings

Primer de tot, experimentarem amb els efectes del preprocessament de dades provant dues tècniques: lematització i stemming, per veure quin d'aquests proporciona millors resultats en la tasca de classificació. També provarem diferents mides de vocabulari, per observar com afecta la selecció de paraules rellevants al rendiment del model.

6.5.1 Lemmatizer vs Stemming

Després de fer el preprocessament de dades, el model mostra una lleugera millora en el F1 macro average (cal tenir en compte sempre la variabilitat dels experiments). Això confirma la importància del preprocessament per millorar el rendiment del model. No obstant això, l'elecció entre lematització i stemming no sembla influir significativament en els resultats, indicant que són equivalents en aquest cas per a la tasca de classificació. Per continuar amb els experiments, farem servir stemming.

6.5.2 Provar mida vocabulari

Hem provat diverses configuracions de mida de vocabulari (300, 400, 500, 600 i 700) i els resultats són bastant semblants. Això podria indicar que les paraules més informatives ja estan presents en un vocabulari de mida 300 o 400, o que el model actual encara és massa limitat per beneficiar-se dels vocabularis més amplis.

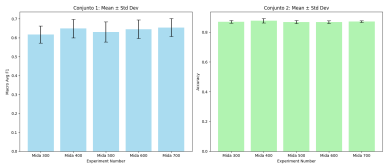

Figura 1: Rendiment del model segons la mida del vocabulari. (Barres d'error amb rang entre 0 i 1 per tenir una comparació més visual, blau = macro avg f1, verd = accuracy)

Tanmateix, en experiments futurs es preveu incrementar la complexitat del sistema, cosa que podria permetre al model captar millor patrons o relacions semàntiques entre les paraules. Per tant, escollim una mida de vocabulari de 600 per deixar marge per a que els models més complexos puguin aprendre matisos addicionals i aprofitar una major varietat lèxica.

6.6 Provar mides de Embeddings

Fixant la mida del vocabulari a 600, hem provat diverses configuracions de dimensions d'embeddings (50, 100, 150, 200, 250, 300 i 350).

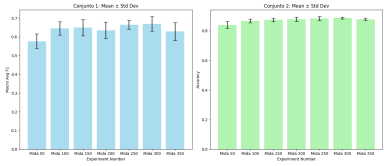

Figura 2: Rendiment del model segons la dimensió del vocabulari

Amb una dimensió de 50, obtenim un rendiment menor, ja que aquesta dimensió no sembla tenir prou

capacitat per representar de manera efectiva les relacions entre les paraules. En canvi, a partir de 100 dimensions, els valors de F1 macro average són molt semblants. Això indica que amb 100 dimensions és suficient per capturar les relacions semàntiques més rellevants entre les paraules, especialment tenint en compte que els textos del nostre conjunt de dades no són excessivament complexos i no necessiten una representació tan profunda.

6.7 Provar xarxes convolucionals

Les CNN són molt útils per a tasques de classificació de text, ja que capten relacions locals entre paraules i poden detectar patrons repetitius dins dels textos. En comparació amb els models recurrents com les RNN o les LSTM, les CNN treballen amb menys paràmetres, cosa que les fa més ràpides i fàcils d'entrenar.

A partir dels paràmetres obtinguts en experiments anteriors, provem diferents configuracions de CNN.

6.7.1 Provar model bàsic: Embedding + Convolució + Pooling (max vs avg)

L'arquitectura composta per embedding + convolució + pooling és una bona opció com a base per començar a treballar amb models de text. Tenint això, hem provat dues opcions per al pooling: max pooling i average pooling. El max pooling ha mostrat millors resultats, amb un millor valor de macro avg f1 comparat amb el average pooling. Aquest comportament és esperat, ja que el max pooling selecciona el valor màxim de cada regió de la capa convolucional, per tant, identifica característiques més destacades.

Tot i això, els resultats obtinguts amb les capes CNN han empitjorat. Observant les dades de loss durant l'entrenament veiem que pot ser degut a que el nombre d'epochs utilitzats durant l'entrenament no ha estat suficient per permetre que el model aprengui.

En resum, en els experiments següents provarem utilitzant max pooling i augmentant el nombre d'epochs per obtenir millor rendiment.

6.7.2 Provar nombre de filtres de CNN

En el model anterior, la loss era alta, i podria indicar que hi ha underfitting. Per tant, per millorar el rendiment del model, provarem d'augmentar la seva complexitat afegint més filtres a les capes convolucionals. La configuració de kernel i pooling queden iguals, és a dir, kernel 3 i utilitzant max pooling. Per que el model pugui aprendre suficientment, augmentem el nombre d'epochs a 10.

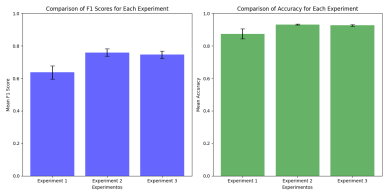

Figura 3: Rendiment del model segons la dimensió del vocabulari (cal tenir en compte la variabilitat dels resultats)

El nombre de filtres provats ha estat de 32, 128 i 256. Hem observat que el rendiment millora a mesura que incrementem el nombre de filtres. Quan utilitzem 256 filtres, el model sembla aprendre molt bé, fins i tot mostrant signes d'overfitting. Això suggereix que afegir més complexitat al model no necessàriament millorarà

el rendiment, i podria fins i tot empitjorar-lo. Per confirmar-ho, podem provar d'augmentar encara més la complexitat del model.

6.7.3 Número de Capes Convolucionals i provar diferents kernel size

Tal com ja esperàvem, afegir més capes convolucionals de 256 filtres no ha millorat el rendiment del model. De fet, observant les gràfiques de la loss, veiem que fins i tot es torna inestable. Això indica que el model ja és prou complex per al tipus de dades que estem treballant, i afegir més capes o filtres podria causar overfitting o dificultar l'aprenentatge de patrons generals.

A més, veiem que augmentar el kernel size disminueix el rendiment. Això és possiblement degut a que els textos de dataset són curts, per tant, les relacions més importants per a la classificació solen ser a nivell local. Aleshores, augmentar la mida del kernel pot fer que el model perdi aquestes relacions petites.

6.8 Provar xarxes recurrents

De vegades, la combinació de CNN+RNN pot ser útil depenent de les característiques de les dades. CNN són bons per detectar patrons locals mentre que les RNN poden capturar dependències a llarg termini. Provem totes aquestes combinacions agafant la CNN obtinguda de l'experiment anterior (256 filtres, kernel de 3 i amb un max pooling).

6.8.1 CNN + LSTM

Per aquest cas hem provat 3 experiments, combinant el CNN obtingut amb LSTM de 32, 64 i 128 unitats.

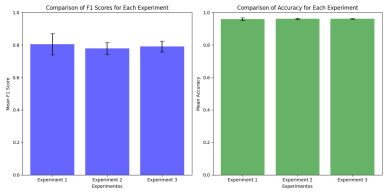

Figura 4: Rendiment del model segons la quantitat d'unitats en LSTM

Els resultats indiquen que amb una configuració senzilla de 32 unitats ja n'hi ha prou. A més, observant les gràfiques de loss i f1 d'entrenament i validació, veiem que el model presenta d'overfitting, per tant, no cal fer-la més complexa. Per comprovar-ho, hem fet experiments afegint més capes i el model comença a ser inestable.

6.8.2 CNN + GRU

Amb CNN+GRU obtenim quasi els mateixos resultats que l'apartat anterior.

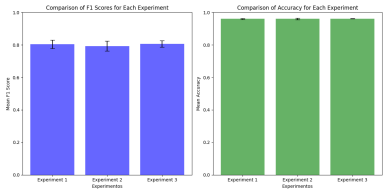

Figura 5: Rendiment del model segons la quantitat d'unitats en GRU

Hem combinat el CNN obtingut amb GRU de 32, 64 i 128 unitats. Arribem a les mateixes conclusions: amb una configuració senzilla de 32 unitats ja n'hi ha prou i el model presenta d'overfitting. Com abans, hem fet experiments afegint més capes i el model comença a ser inestable.

6.8.3 LSTM

En les proves anteriors habíem obtingut un molt bon rendiment, però presentaven d'overfitting. És per això que ara caldria minimitzar la complexitat del model. Per tant, provem de treure la CNN i provar només afegint la capa LSTM.

En aquest cas, tornem a fer tres experiments, amb LSTM de 32, 64 i 128 unitats (sense CNN).

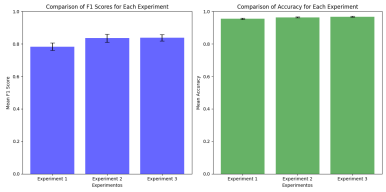

Figura 6: Rendiment del model sense CNN segons la quantitat d'unitats en LSTM

Veiem que s'ha obtingut un millor valor de macro average f1 score (amb 128 unitats podem arribar un valor més alt que 0.8), però, tornant a mirar les gràfiques de loss i f1 d'entrenament i validació, veiem que encara hi ha overfitting. El model ja és massa complex, per tant, no farem experimentacions afegint més capes.

6.8.4 GRU

La idea pel GRU és la mateixa que pel LSRM. En aquest cas, realitzem tres experiments amb GRU, utilitzant 32, 64 i 128 unitats (sense aplicar cap capa CNN).

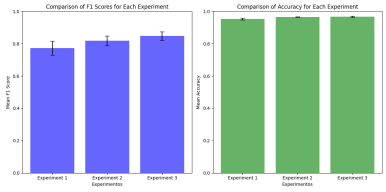

Figura 7: Rendiment del model sense CNN segons la quantitat d'unitats en GRU

Observem que el model presenta un cert sobreajustament (overfitting). Tot i això, el GRU mostra resultats molt similars als de LSTM en termes de precisió, però amb menys paràmetres, el que fa que sigui més ràpid i, per tant, una opció ideal quan els dades de seqüència no són gaire llargues.

Tal com ja s'ha dit en l'apartat anterior, com que el model ja és massa complex, no farem experimentacions afegint més capes.

6.9 Regularització

Com que el nostre dataset no té textos molt elaborats amb relacions semàntiques molt profundes, el model actual pot ser massa complexa. La regularización con Dropout és una tècnica per evitar el sobreajustament en xarxes neuronals. Consisteix a apagar aleatòriament una part de les neurones durant l'entrenament per força el model a ser més general i a no sobreajustar-se a les dades d'entrenament.

Donat el nostre model, podem aplicar la capa Dropout després de les capes Embedding, GRU, o Dense.

6.9.1 Dropout Embedding

Aplicant Dropout després de l'Embedding, força el model a no dependre exclusivament de determinades representacions d'entrada. Per tant, apliquem Dropout després de la capa d'embedding amb un valor de 0.3 com a base.

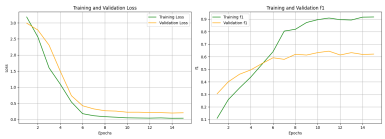

Figura 8: Gràfiques de loss i f1 score de les particions d'entrenament i validació de l'entrenament

Observant les gràfiques de loss i f1 score de l'entrenament veiem que no ha millorat gaire la situació d'overfitting. Però tampoc redueix gaire el rendiment (0.8 aproximadament de macro average f1 score), per tant, el mantenim.

6.9.2 Dropout Embedding + GRU

Els models com GRU tenen una gran capacitat de modelar seqüències llargues i complexes, però també poden sobreajustar-se fàcilment, així que el Dropout ajuda a evitar aquest problema.

Després d'aplicar Dropout de valor 0.3 a la sortida de GRU veiem que ha minimitzat una mica l'overfitting, però ha causat una caiguda de valor del macro average f1 score fins a 0.73 aproximadament. Per tant, no afegirem aquest Dropout.

6.9.3 Dropout Embedding + Dense

Aplicant Dropout a les capes densament connectades, forcem al model a no dependre massa d'algunes neurones. Això evita que el model aprengui patrons massa específics dels dades d'entrenament.

Després d'aplicar Dropout de valor 0.3 a la sortida de Dense veiem que el model ja no té overfitting. El punt negatiu és que, potser s'han apagat masses neurones i el rendiment ha baixat intensament fins a 0.66 aproximadament de macro average f1 score.

6.9.4 Dropout(0.2) Embedding + Dense

Per evitar que s'apaguin tantes neurones i que el model pugui tenir un rendiment millor, aplicarem de nou Dropout després de les capes d'Embedding i Dense, però ara amb un valor de 0.2.

Veiem que torna a haver-hi una mica d'overfitting, però s'ha aconseguit augmentar el rendiment fins a 0.75 aproximadament de macro average f1 score.

7 Inferencia en test

Recapitulem. El model actual utilitza la tècnica stemming, té una dimensió d'embedding de 100, té una mida de vocabulari de 600, segueix la mètrica macro average f1 score, fa servir class weights, no té capes de convolució ni LSTM, té una capa de GRU amb 128 unitats i dos Dropout de 0.2 (un després de la capa d'embedding i l'altre després de la Dense).

Un cop tenim el model definitiu, el següent pas és fer la inferència en les dades de test per comprovar el rendiment.

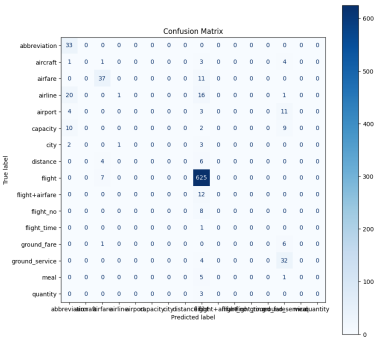

Figura 9: Matriu de confusió inicial

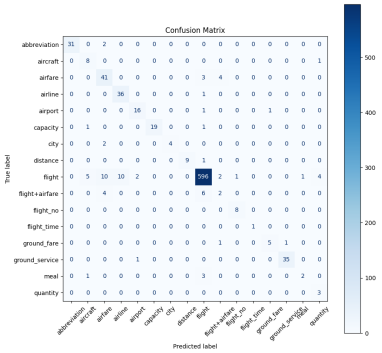

Figura 10: Matriu de confusió final

En comparació amb la matriu de confusió del model inicial, veiem que aquest prediu també les classes minoritàries. Calculant el macro average f1 score obtenim un valor de 0.702. Hem pogut augmentar molt el rendiment tenint en compte que l'inicial era 0.19.

8 Conclusió

Aquest treball ha consistit en aprendre tècniques de deep learning aplicades a la millora d'un model per a la classificació d'intencions. Hem provat diferents estratègies d'optimització que han tingut un impacte en el rendiment i la capacitat de generalització del model. El preprocessament ha ajudat a simplificar les representacions textuals, i en ajustar la mida dels embeddings, hem comprovat que una mida intermèdia era suficient per captar patrons rellevants sense complexitat addicional. Equilibrar les classes amb class weights ha augmentat la precisió en les classes minoritàries, fent el model més robust. L'ús de capes convolucionals ha permès detectar patrons locals, mentre que les capes recurrents (LSTM i GRU) han captat dependències a llarg termini. A més, afegir dropout com a regularització ha estat essencial per controlar l'overfitting en els models més complexos.