UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Grau en Intel·ligència Artificial

Tractament de Veu i Diàleg

Pràctica 2.1: Intent Recognition

Maria Gil Casas i Lola Monroy Mir

13 de Novembre, 2024

Índex

1 Introducció

La capacitat de reconèixer les intencions dels usuaris és fonamental per desenvolupar aplicacions més intel·ligents i personalitzades, millorant així l'experiència de l'usuari. Aquest treball se centra en el desenvolupament i l'optimització d'una xarxa neuronal per al reconeixement d'intencions a partir de dades de text. La intenció d'un usuari es refereix a l'objectiu o la finalitat darrere d'una consulta o comanda específica. Per exemple, quan un usuari diu "Fes-me una cançó de rock", l'intenció és escoltar música, específicament del gènere rock. La correcta identificació d'aquestes intencions permet als sistemes automàtics respondre de manera més precisa i rellevant.

Per aconseguir aquest objectiu, s'ha utilitzat un conjunt de dades que conté diverses entrades de consultes d'usuaris, etiquetades amb les seves corresponents intencions. Aquest conjunt de dades ha estat preprocesat mitjançant tècniques com la tokenització, el padding i la codificació de les etiquetes, assegurant una representació adequada per al model de xarxa neuronal. L'arquitectura del model inclou capes d'embedding per transformar les paraules en vectors densos, seguides de capes recurrents (GRU) per capturar les dependències temporals en les seqüències de text, així com capes de regularització com el Dropout per prevenir el sobreajustament.

A més, s'han implementat diverses estratègies d'optimització, com l'ús de regularització L2, l'ajust de la mida dels embeddings i l'aplicació de l'early stopping durant l'entrenament, amb l'objectiu d'augmentar la precisió i la capacitat de generalització del model. També s'ha explorat l'ús d'embeddings preentrenats per millorar les representacions semàntiques de les paraules, aprofitant coneixements previs incorporats en aquests vectors.

Aquest treball no només demostra l'aplicació pràctica de les tècniques avançades d'aprenentatge profund en el reconeixement d'intencions, sinó que també subratlla la importància del preprocesament de dades i de l'ajust dels hiperparàmetres per aconseguir models més robustos i eficients. Els resultats obtinguts evidencien com les diverses optimitzacions poden influir significativament en el rendiment del model, proporcionant una base sòlida per a futurs desenvolupaments en aquest camp.

1.1 Base de dades

El conjunt de dades proporcionat té 4.977 entrades i 3 columnes. Aquestes columnes són:

1. **Frases de consulta:** Conté frases completes, com ara sol·licituds de vol o informació relacionada amb la reserva, expressades en text natural.
2. **Etiquetes de seqüència:** Conté etiquetes associades a cada paraula o part de la frase en format IOB (Inside-Outside-Beginning). Les etiquetes indiquen la funció o categoria de cada paraula dins del context, com ara noms de ciutats, hores de sortida, i altres elements rellevants de les consultes.
3. **Intenció:** Aquesta última columna categoritza el tipus de consulta, per exemple, "flight", "airfare", etc., indicant l'objectiu principal de la sol·licitud.

2 Exercicis

Els exercicis 1 al 4 es troben al fitxer *1_ original.ipynb*, i la resta es distribueixen en diferents fitxers que es dediquen cadascun en l'experimentació de diferents trets de la xarxa. Anirem comentant on es troba cadascun a mesura que avançem.

2.1 Exercici 1: Tokenització del text

El primer pas és convertir les frases en una seqüència d'índexs que representen paraules individuals. Això es fa usant tokenizer.texts to sequences(), una funció que converteix cada paraula en un número únic segons el vocabulari après durant la fase d'entrenament del tokenitzador.

Per fer aquest exercici hem entrat a la documentació de keras fins a donar amb l'acció que volem: 'text to sequences'. D'aquí, anem a parar a una funció de la mateixa llibreria a la que cridarem.

2.2 Exercici 2: Padding

El padding serveix per dir strings de dades que siguin de la mateixa longitud. això es necessari per poder fer operacions amb les dades. Si no fos així, operar seria mes complicat. La manera mes facil d'igualar la longitud de les paraules es afegir 0s. Quan fem padding (afegim 0s), si enganxem els 0s al darrere en comptes de al davant estem reduint el soroll de les dades. Per enganxarlos pel darrere, es tan simple com afegir l'etiqueta 'padding = 'post" .

2.3 Exercici 3: Validation i Train

En aquest exercici, apliquem els mateixos passos de preprocessament als conjunts de validació i test. Això inclou: convertir les frases en seqüències d'índexs, estandarditzar la longitud de les seqüències amb pad sequences, assegurant que totes tinguin la mateixa longitud.

Aquest preprocessament garanteix que les particions de validació i test siguin consistents amb el conjunt d'entrenament, permetent al model generalitzar millor sobre les dades no vistes.

Es important recordar que quan transformen conjunt de validació i train, hem d'utilitzar el nostre propi diccionari. Sino, el problema seria que tindria paraules que no estiguessin en el val i el test. Hauriem de marcarles com a unknown o el classificador no les reconeixeria i donaria error. Per sort, keras ens proporciona l'etiqueta unknown.

2.4 Exercici 4: Disseny de l'arquitectura del model

En aquesta secció de la pràctica, l'objectiu és dissenyar l'arquitectura d'un model de xarxa neuronal compost per quatre capes diferenciades. El primer component és una capa d'embedding, que transforma les dades de text d'entrada en vectors densos de mida fixa, capturant les relacions semàntiques entre les paraules i reduint la complexitat computacional. A continuació, s'utilitza una capa de pooling que sintetitza la informació

rellevant de la seqüència d'entrada mitjançant la selecció del valor màxim, simplificant així la seva representació a un tensor de menor dimensió. La tercera capa és una capa densa amb una funció d'activació ReLU, la qual permet al model aprendre patrons no lineals en les dades, augmentant la seva capacitat per identificar relacions complexes. Finalment, el model conclou amb una altra capa densa que utilitza la funció Softmax per convertir els valors de sortida en probabilitats normalitzades, assignant una probabilitat a cada classe possible per a les prediccions finals. Aquesta estructura pretén oferir una solució robusta i eficient per al processament i la classificació de dades de text.

Els paràmetres d'aquestes capes han estat seleccionats basant-nos en l'experimentació, on hem provat diferents tamanys per la capa de pooling i la capa densa. No obstant això, els millors resultats els hem obtingut amb l'arquitectura descrita anteriorment. En el proper exercici, aprofundirem en aquestes experimentacions per optimitzar encara més el model.

Actualment, el model aconsegueix una precisió accuracy de 0.82. Les classificacions incorrectes es produeixen majoritàriament amb l'etiqueta 'flight'.

2.5 Exercici 5: Modificació de parámetres

En aquest exercici, modificarem diversos paràmetres del model per analitzar el seu impacte en la *accuracy*. Primer, experimentarem amb el preprocessament del text, ajustant la mida del vocabulari i aplicant tècniques com la lematització o el *stemming*. També proverem diferents mides d'*embeddings* per avaluar com afecten la representació semàntica del text i la precisió del model. A més, inclourem capes convolucionals per detectar patrons locals i capes recurrents, com LSTM i GRU, per capturar dependències a llarg termini en les seqüències de text, detallant els valors utilitzats i les decisions preses en el procés. Per abordar el problema de *overfitting*, afegirem *Dropout* en punts estratègics del model, explicant la seva posició i els valors seleccionats. Finalment, ajustarem els pesos de les classes desbalancejades per millorar l'entrenament mitjançant el paràmetre *class weight* de Keras, assegurant així un tractament equitatiu de les classes en el conjunt de dades. Aquestes modificacions tenen com a objectiu optimitzar el rendiment del model, maximitzant la seva capacitat de predicció sota diferents configuracions.

Aquests canvis estan fets cadascun en un notebook individual, per il·lustrar el més clarament possible les modificacions que realitzem. Els recollirem tots al final en un únic notebook, i cadascun recull els canvis de tots els fitxers previs a ell.

2.5.1 Preprocessament

Stemming Vs Lemmatizing

El fitxer que conté els experiments realitzats en aquesta secció es troba a *2-stemm-lemm.ipynb*. En aquest notebook, hem aplicat el mateix procés utilitzant exactament la mateixa arquitectura de model, modificant únicament la tècnica de tokenització: *stemming* i *lemmatizing*.

Stemming i lemmatizing són dues tècniques de normalització de text utilitzades en el processament del llenguatge natural (NLP) per reduir les paraules a la seva forma base. Aquestes tècniques ajuden a millorar la qualitat de les dades d'entrada, facilitant així el procés d'aprenentatge del model. El *stemming* consisteix en tallar els sufixos de les

paraules per obtenir una arrel comuna, coneguda com a "stem". És més ràpid i senzill d'implementar, pero pot ocasionalment generar arrels que no són paraules reconeixibles lingüísticament. El *lemmatizing* redueix les paraules a la seva lema, que és la forma base o diccionari de la paraula. És més precís que el *stemming*, ja que manté la integritat lingüística de les paraules, però requereix més recursos computacionals i pot ser més lent d'executar. Després de realitzar diverses iteracions amb ambdues tècniques, hem observat que utilitzant el *stemmer* obtenim una precisió (*accuracy*) de 0.855, mentre que amb el *lemmatizer* arribem a una precisió de 0.84. Aquesta diferència indica que el *stemming* proporciona una millor representació de les paraules per al nostre model en comparació amb el *lemmatizing*.

Per tant, per a les fases restants del preprocessament i la pràctica en general, hem decidit utilitzar el *stemmer*. Aquesta elecció s'ha basat en els resultats experimentals que demostren una millora en la precisió del model quan s'aplica el *stemming* sobre el *lemmatizing*

La resta del preprocessament

En el fitxer *3-preprocess.ipynb* ja hem afegit el *stemmer* com el únic tokenitzador, ja que hem vist que el millor rendiment presenta per a aquesta pràctica.

Amb el tamany de vocaburari de l'original (500) obtenim un accuracy de 0.82. En canvi, si augmentem el vocabulari a 1500, la mitjana de l'accuracy disminueix una decima. Pero, si usem un vocab de tamany 1000, tenim **accuracy de mitjana 0.83 canviant el tamany del vocabulari**, per tant determinem que aquest es et tamany òptim.

També s'han ajustat els valors de *batch size* i *epochs* (executant 4 epochs en lloc de 2, per exemple), la qual cosa es nota en els resultats de precisió. Tal i com s'indica al fitxer, es poden veure uns valors d'accuracy millorats. Concretament, *un valor de accuracy de 0.88 augmentant el nombre d'èpocs i minvant el batch size.*

Les etiquetes de test i validació passen per un procés de neteja de caràcters especials i espais addicionals. Això garanteix que les etiquetes tinguin un format adequat abans de ser transformades i codificades. També passem totes les paraules a minuscules per facilitar el seu processament.

2.5.2 Mida dels Embeddings

L'arxiu que correspon a aquesta secció de la pràctica és *4-embedding-dim.ipynb*. L'enunciat ens demana que provem diferents mides d'Embeddings i observem com canvia l'accuracy del model. La mida de l'embedding determina la dimensionalitat de l'espai on es representen les paraules, i pot afectar la capacitat del model per capturar relacions semàntiques i patrons al text.

Comencem provant un embedding de mida 50, obtenint una precisió de validació de 0.90 i una precisió de test de 0.87. Tot i que el rendiment és bo, observem que el model no capta tota la complexitat semàntica que permetria una major generalització en el conjunt de tests.

Amb un embedding de mida 100, trobem millores en ambdues particions, aconseguint una precisió de validació de 0,92 i de test de 0,89. Això ens indica que un augment de

la dimensionalitat facilita una representació més precisa de les relacions entre paraules i ofereix un bon balanç entre capacitat de representació i complexitat computacional.

En incrementar la mida a 200, la precisió de validació continua en augment amb un valor de 0.93, encara que la precisió en el conjunt de test es manté en 0.89, similar a l'obtinguda amb una mida de 100. Aquesta manca de millora significativa test suggereix que el model podria començar a sobreajustar-se a les dades d'entrenament.

Provem llavors amb una mida de 300, cosa que produeix millores addicionals tant en el conjunt de validació, aconseguint una precisió de 0.94, com en el conjunt de test, amb una precisió de 0.90. El model sembla aprofitar millor la major dimensionalitat per representar de manera més completa les relacions semàntiques al text.

Finalment, experimentem amb un embedding de mida 500, que assoleix els millors resultats al conjunt de test, amb una precisió de 0.91, i al conjunt de validació, amb una precisió de 0.94. Tot i que l'increment en la precisió respecte a la mida 300 és marginal, la precisió de validació s'estabilitza i la pèrdua continua disminuint, cosa que indica que el model ha trobat una representació adequada de les dades.

En conclusió, observem que l'increment de la mida de l'embedding millora el rendiment del model, encara que el benefici marginal disminueix a partir de les mides 300-500. Per a aquest experiment, sembla que un embedding de 300 i 500 dimensions és la millor opció quant a equilibri entre generalització i rendiment.

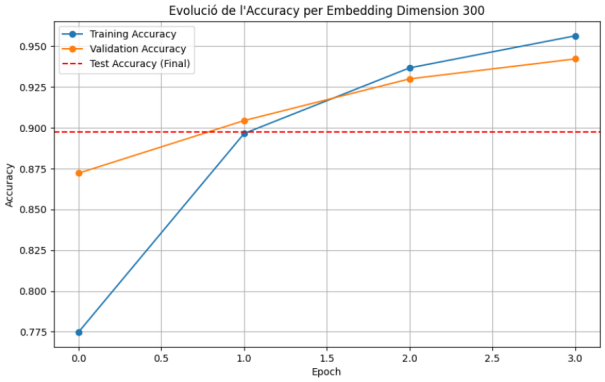

Figure 1: Evolució de l'accuracy per embedding dimensió 300

El gràfic mostra un augment ràpid en la precisió d'entrenament durant les primeres èpoques, seguit d'una millora gradual a mesura que s'ajusta el model, mentre que la precisió de validació segueix una tendència similar, indicant un aprenentatge sense sobreajustament. La línia de precisió final en test, al voltant de 0.90, és a prop de la precisió de validació, la qual cosa suggereix que el model generalitza bé i manté un bon equilibri entre precisió i generalització. Això dóna suport a l'elecció d'un embedding de mida 300 com a configuració adequada per al model. Es poden veure la resta de gràfics al fitxer que correspona aquesta secció.

2.5.3 Xarxes Convolucionals

En el fitxer *5-convolucional.ipynb* s'experimenta amb diferents configuracions de xarxa convolucional per trobar la configuració òptima.

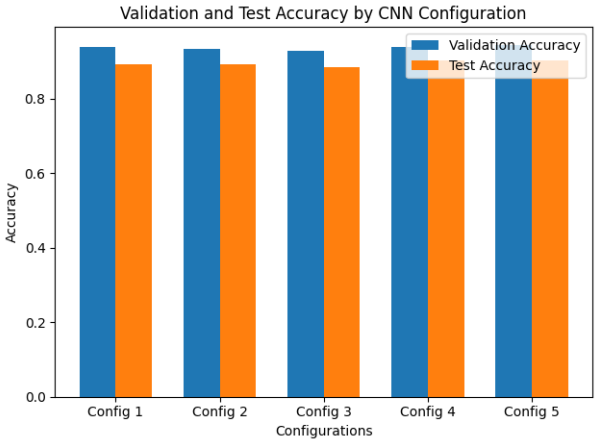

Figure 2: Accuracy amb diferents configuracions

En comparar els resultats de les diferents configuracions de la xarxa convolucional, observem un patró en el rendiment que varia segons els valors de filtres, la mida del nucli i la mida del pooling. Hem dissenyat diferents configuracions per posar a prova el seu rendiment. La Configuració 1, amb 32 filtres, un nucli de mida 3 i pool de 2, assoleix una precisió de validació de 0.94 i una precisió en el conjunt de tests de 0.89. Aquesta configuració inicial mostra un bon rendiment general, encara que algunes classes presenten baixos valors de precisió i recall al report de classificació, la qual cosa indica dificultat en la identificació de certs patrons.

La Configuració 2 utilitza 64 filtres amb kernel de mida 3 i pool de 2, aconseguint una precisió de validació de 0.93 i de test de 0.89. Tot i que aquest augment de filtres no aporta grans millores respecte a la primera configuració, s'observen lleugeres variacions en la precisió i el *recall* de certes classes.

Amb la Configuració 3 (32 filtres, kernel de mida 5, pool de 2), el model obté una precisió similar en validació (0.93) però lleugerament inferior en test (0.88), mostrant que un kernel més gran no millora significativament el rendiment.

La Configuració 4 (64 filtres, kernel de mida 5, pool de 2) aconsegueix una precisió de validació de 0.94 i de test de 0.90, suggerint que l'augment de filtres i mida de kernel ajuda el model a captar patrons més complexos.

Finalment, la Configuració 5 (128 filtres, kernel de mida 3, pool de 3) manté la precisió de validació (0.94) i test (0.90) de la configuració 4, indicant que es pot estar arribant a la capacitat màxima de generalització amb aquests hiperparàmetres.

En resum, les configuracions 4 i 5, amb més filtres i mida de kernel i un pool moderat, mostren el millor rendiment en test. Tot i això, no totes les classes obtenen alts valors de precisió i *recall*, la qual cosa reflecteix dificultats en identificar patrons de classes minoritàries. Triem la configuració 5 per la seva consistència en les iteracions.

2.5.4 Xarxes Recurrents

En el fitxer *6-recorrents.ipynb* hem provat diferents configuracions de xarxes recurrents per avaluar el seu rendiment en la tasca de classificació. Inicialment, vam crear dos models

separats: un amb una capa LSTM i un altre amb una capa GRU, tots dos configurats amb 64 unitats. L'objectiu era observar com es comportaven les arquitectures recurrents clàssiques en el nostre conjunt de dades i determinar quin tipus de capa recurrent oferia millors resultats.

Els resultats van mostrar que el model amb la capa LSTM obtenia una precisió relativament baixa, similar als primers intents abans d'afinar el preprocesament. Això indica que l'arquitectura LSTM no estava capturant de manera òptima els patrons seqüencials necessaris per a la tasca. En canvi, el model amb la capa GRU va oferir un rendiment molt més satisfactori, amb una precisió de validació d'aproximadament 0.91. Aquest resultat superior de GRU es pot atribuir a la seva estructura més simple, que tendeix a generalitzar millor en problemes on les relacions temporals no són excessivament complexes.

Amb aquests resultats preliminars, vam decidir experimentar amb una combinació de capes LSTM i GRU en un mateix model, per veure si aquesta combinació aportava millores. Tot i que aquesta configuració va millorar lleugerament els resultats respecte al model únicament amb LSTM, encara no va superar la precisió obtinguda amb només la capa GRU. Això ens va portar a descartar l'ús de LSTM i optar per GRU com a arquitectura recurrent principal en el model final, ja que proporciona un bon rendiment sense afegir una complexitat innecessària.

Després d'haver seleccionat GRU, vam fer proves amb diferents quantitats d'unitats: 32, 64 i 128. Els millors resultats es van obtenir amb les configuracions de 64 i 128 unitats, amb una precisió similar en ambdós casos. Com que la diferència en precisió entre les dues configuracions era gairebé imperceptible, vam optar per la configuració de 64 unitats per evitar una major complexitat computacional.

2.5.5 Regularització

A mesura que provem configuracions amb un nombre més elevat de paràmetres, observem que el model tendeix a experimentar overfitting de forma precoç durant l'entrenament. Per mitigar aquest problema, hem decidit incorporar capes de Dropout al model. Aquesta tècnica ens permet reduir el sobreajustament en desactivar aleatòriament una proporció de les neurones en cada època, evitant que el model es faci excessivament depenent d'unes connexions específiques. Això ho veiem en el fitxer *7-regularització*.

En primer lloc, amb un valor de Dropout de 0.2, el model aconsegueix una alta precisió tant en validació com en test, aconseguint un 0.96 en validació i un 0.89 en el conjunt de test que és gairebé el que aconseguiem abans. Aquesta configuració permet al model aprendre els patrons rellevants sense una regularització excessiva, aconseguint una bona generalització. La baixa taxa de desactivació de neurones facilita la retenció dinformació útil, permetent al model captar relacions significatives en les dades.

A mesura que augmentem el Dropout a 0.3, el rendiment del model en el conjunt de tests disminueix lleugerament, assolint una precisió de 0.84 en test. Encara que aquest valor de Dropout comença a mitigar el sobreajustament, també provoca una petita pèrdua d'informació, afectant la capacitat del model per generalitzar alguns patrons clau. Amb un Dropout de 0.4 assoleix una precisió de 0.93 en validació i 0.87 en test.

Quan augmentem el Dropout a 0.5, la capacitat del model per captar patrons específics es veu clarament afectada. Tant la precisió en validació com en tests baixen consider-

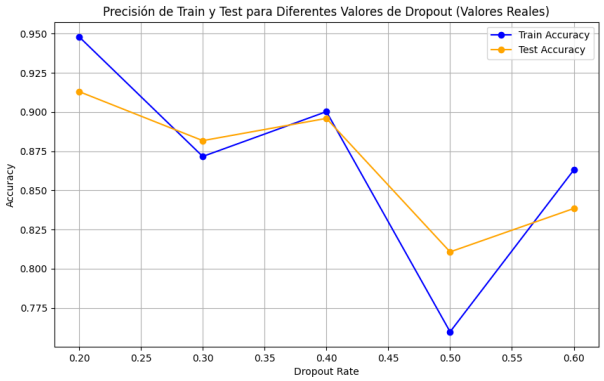

Figure 3: Precisió de Train i Test per a Diferents Valors de Dropout

ablement, indicant que una gran quantitat de neurones desactivades provoca una pèrdua significativa d'informació i porta el model a un subajustament. Finalment, amb un valor de Dropout de 0.6, el rendiment continua sent limitat en ambdós conjunts, evidenciant un subajustament més gran que impedeix que el model aprofiti adequadament els patrons de les dades. En aquest cas, el model té dificultat per capturar relacions complexes a causa de la quantitat dinformació que perd en cada pas dentrenament.

En conclusió, els resultats suggereixen que un Dropout moderat, especialment en valors de 0.2, 0.3 o 0.4 permet al model aprendre de manera efectiva sense incórrer en sobreajustament. Veient la imatge seguent, la configuració de Dropout de 0.2 sembla oferir el millor equilibri entre aprenentatge i generalització, aconseguint la precisió més alta al conjunt de test.

2.5.6 Balencejat de classes

La classe "flight" és molt més nombrosa que altres, com ara "cheapest", "aircraft+flight+flight no" i "airfare+flight time", que apareixen només una vegada. Per abordar el desbalanceig de classes en el conjunt de dades, hem intentat utilitzar el paràmetre *class weight* en el mètode `fit` de Keras. Aquesta estratègia permet que el model doni més importància a les classes menys representades durant l'entrenament.

No cal calcular ni aplicar els pesos de classe als conjunts de validació i prova. Els pesos de classe s'utilitzen exclusivament durant l'entrenament per ajustar la funció de pèrdua, de manera que el model preste més atenció a les classes menys freqüents. Durant l'avaluació als conjunts de validació i prova, generalment no s'apliquen aquests pesos.

En el primer intent, l'ús del *class weight* ha resultat en una precisió (*accuracy*) de 0.02, molt baixa. Aquest resultat es pot veure al fitxer *8-balanceig.upynb*. Això es deu a la dificultat de balancejar adequadament les classes amb dades etiquetades de manera desigual.

Per fer front a aquest problema, hem considerat una altra opció: eliminar dades. Si reduïm la classe més present, "flight", podem disminuir el desbalanceig. No obstant això, això comporta una reducció significativa del conjunt de dades, cosa que no ens podem permetre si volem que el model aprengui el màxim possible de la informació disponible. Hem arribat a la conclusió que la millor opció per al balanceig és eliminar aquelles entrades

que corresponguin a classes inexistents, i deixar la resta com està, ja que la precisió era satisfactòria fins i tot abans d'ajustar els pesos de classe amb Keras.

2.5.7 Refinament final

En aquesta secció, prenem el model rere haver fet tots els canvis individualment, i realitzem experiments amb les eines que hem descrit fins ara. Això queda recollit al fitxer *9-final.ipynb*. Ho fem perque fins ara, hem considerat el ordre i les accions per separat, epro es important considerar el conjunt també, així que per acabar, farem el seguent:

1. Afegir capa bidireccional GRU: Les xarxes neuronals recurrents bidireccionals processen la seqüència en les dues direccions (cap endavant i cap enrere), cosa que permet que el model capturi informació de context tant passat com futur. Això pot millorar el rendiment en tasques de processament de llenguatge natural, com ara el reconeixement d'intencions. Hem reemplaçat la capa GRU per una capa Bidirectional(GRU(64)). Això permet que el model processi la seqüència d'entrada a les dues direccions, potencialment millorant la capacitat del model per comprendre el context complet de les intencions.
2. Augmentem de nou el tamany dels embeddings
3. Incorporem Regularització amb L2 a la capa densa i pujem el nombre d'èpochs a 6: La regularització ajuda a prevenir el sobreajustament en penalitzar grans pesos al model. Usar regularització L2 a les capes denses pot millorar la generalització del model a dades no vistes. Hem afegit kernel regularizer=l2(0.001) a la capa Dense(128) per aplicar regularització L2. A més, hi afegim una segona capa Dropout amb una taxa de 0.3 per augmentar la regularització i ajudar a prevenir el sobreajustament. També incrementem el nombre d'èpoques a 6 per permetre que el model s'entreni més completament amb la regularització nova.
4. Implementar early stopping: L'early stopping atura l'entrenament quan el rendiment en el conjunt de validació ja no millora, cosa que ajuda a prevenir el sobreajustament ia estalviar temps d'entrenament. Hem incorporat el callback EarlyStopping que monitoritza la mètrica val loss. Si el valloss no millora durant 2 èpoques consecutives (patience=2), l'entrenament s'aturarà i es restauraran els millors pesos observats durant l'entrenament (restore-best-weights=True). A més, augmentem el nombre d'èpoques a 20 per permetre que l'entrenament es faci completament, però l'early stopping evitarà un entrenament excessiu.

L'accuracy final gira al voltant del 0.96, es molt satifactoria, i no està fent overfit ja qeu hem afegit mesures de regularització per prevenir-ho.

2.6 Possibles ampliacions

Tot i que el model desenvolupat ha demostrat una capacitat notable en el reconeixement d'intencions, hi ha diverses coses que podriem fer per millorar encara més el rendiment i l'eficiència del sistema.

La primera podria ser incrementar la mida del conjunt de dades pot ajudar a millorar la precisió del model, especialment per a classes minoritàries. Això permetria al model aprendre una varietat més àmplia de patrons i reduir el risc de sobreajustament.

En segon lloc, podriem utilitzar vectors d'embeddings preentrenats pot enriquir les representacions semàntiques de les paraules, millorant la capacitat del model per captar relacions complexes entre elles. Implementar models d'embeddings contextuals (com BERT) pot proporcionar una comprensió més profunda del significat de les paraules en diferents contextos, augmentant així la precisió en el reconeixement d'intencions.

Tal i com hem apres a la part teòrica de l'assignatura, podriem implementar arquitectures basades en transformers, lo qual pot oferir avantatges en termes de paral·lelització i captura de relacions complexes entre tokens, millorant el rendiment general del model.

A més a més, com ja hem comentat, aplicar tècniques d'oversampling per a les classes minoritàries o undersampling per a les classes majoritàries pot ajudar a equilibrar el conjunt de dades i millorar la capacitat del model per reconèixer totes les classes de manera equitativa.

Seguint en la línia del que hem aprés a classe, incorporar mecanismes d'atenció pot permetre al model centrar-se en parts específiques de la seqüència d'entrada que són més rellevants per a la predicció de l'intenció, millorant així la precisió i la interpretabilitat. No només això, sinò que utilitzar múltiples capes d'atenció pot enriquir la capacitat del model per capturar diferents nivells de relacions semàntiques i sintàctiques dins del text.

Aquestes ampliacions no només poden contribuir a millorar la precisió i l'eficiència del model de reconeixement d'intencions, sinó també a ampliar la seva aplicabilitat en diferents contextos i aplicacions. La combinació d'aquestes estratègies pot conduir al desenvolupament de sistemes més robusts, adaptatius i capaços de proporcionar respostes més precises i rellevants als usuaris.

3 Conclusions

En aquest treball hem explorat el desenvolupament i l'optimització d'un model de xarxa neuronal per al reconeixement d'intencions a partir de dades de text. Mitjançant un enfocament sistemàtic, hem abordat diverses etapes clau, incloent el preprocessament de les dades, el disseny de l'arquitectura del model, i l'ajust dels hiperparàmetres per millorar el rendiment del sistema.

Inicialment, hem preprocessat el conjunt de dades aplicant tècniques com la tokenització, el padding i la codificació de les etiquetes, assegurant una representació adequada per al model de xarxa neuronal. Aquestes etapes han estat crucials per garantir que les dades d'entrada estiguin en un format que el model pugui processar eficientment.

L'arquitectura del model implementada consta de quatre capes diferenciades: una capa d'embedding per transformar les paraules en vectors densos, una capa de pooling per sintetitzar la informació rellevant de les seqüències, una capa densa amb activació ReLU per aprendre patrons no lineals, i finalment una capa de sortida amb activació Softmax per a la classificació de les intencions. Aquesta estructura ha demostrat ser robusta i eficaç en la tasca de classificació de dades de text.

A través de l'experimentació, hem ajustat diversos paràmetres del model, com la mida dels embeddings, la regularització amb Dropout i L2, així com la implementació de l'early stopping durant l'entrenament. Aquestes modificacions han contribuït significativament

a la millora de la precisió del model, arribant a una accuracy de 0.82. No obstant això, hem observat que el model presenta dificultats específiques en la classificació de l'etiqueta 'flight', la qual és la més freqüent i presenta més desafiament per al sistema.

Les conclusions principals d'aquest estudi són les següents. Les tècniques de preprocessament aplicades, especialment el stemming, han tingut un impacte positiu en el rendiment del model, millorant la precisió i la generalització. L'ús d'una capa d'embedding seguida de capes recurrents i densas ha demostrat ser una combinació efectiva per a la classificació de intencions en dades de text. La implementació de tècniques de regularització com el Dropout i la regularització L2, així com l'ajust acurat dels hiperparàmetres, han ajudat a prevenir el sobreajustament i a millorar la capacitat de generalització del model. Tot i els avenços, el model encara presenta dificultats en la classificació de la classe 'flight', indicant la necessitat d'explorar mètodes addicionals per manejar classes desbalancejades o incrementar la representació d'aquestes classes en el conjunt de dades.

En conclusió, aquest estudi ha proporcionat una base sòlida per al reconeixement d'intencions mitjançant xarxes neuronals, demostrant la importància del preprocessament de dades, el disseny acurat de l'arquitectura del model i l'ajust meticulós dels hiperparàmetres. Amb les millores suggerides i una continuada investigació, és possible augmentar encara més la precisió i l'eficàcia dels sistemes de reconeixement d'intencions en aplicacions de processament del llenguatge natural.