Tractament de la Veu i el Diàleg

Pràctica 2 primera part
CLASSIFICACIÓ D'INTENCIONS

Grau en Intel·ligència Artificial

Curs 2024/2025

Marta Juncarol Pi
Jaume Mora Ladària
Abril Risso Matas

Índex

1 Introducció

Aquesta pràctica tracta sobre la classificació d'intencions, una tasca fonamental en sistemes de xatbot, ja que permet identificar les necessitats de l'usuari i oferir una resposta adeqüada a partir del text. La classificació d'intencions és clau per millorar la interacció entre humans i màquines, ja que permet entendre el context i l'objectiu de les consultes dels usuaris.

En aquesta pràctica, ens centrem en el desenvolupament i millora d'un model de classificació d'intencions, capaç d'assignar la intenció correcta a cada entrada de l'usuari.

Per obtenir el model que millor generalitzi, s'han realitzat diversos experiments per avaluar l'impacte de les diferents arquitectures, tècniques de preprocessament, mides d'embeddings i estratègies de regularització. A més, s'ha tingut en compte el desbalanceig de les classes, aplicant tècniques de balanceig per assegurar que el model no afavoreixi cap intenció en particular, sinó que proporcioni una identificació precisa de cada classe.

L'experimentació ens ha permés comparar diferents configuracions i ajustar els paràmetres per optimitzar el rendiment del model, garantint en tot moment que les millores són consistents i reproduïbles en nous conjunts de dades.

2 Experimentació i resultats

En primer lloc, s'han realitzat els exercicis del 1 al 4, seguint les pautes per implementar un model bàsic de reconeixement de la intenció de l'usuari.
A continuació, veurem com es realitzen una sèrie d'experiments amb l'objectiu de millorar el rendiment del model. A partir del model bàsic, s'aniran implementant diverses millores, mantenint aquelles que ofereixin millors resultats, és a dir, cada nou experiment partirà de l'última millora obtinguda. **L'avaluació es farà sobre el conjunt de validació**, reservant el conjunt de test per al model final, es realitzaran 10 èpoques per a cada experiment. Per cada experiment s'ha fixat una seed a 42, d'aquesta manera garantitzem resultats reproduibles.
D'altra banda, donat que les classes estan desbalancejades, s'utilitzarà principalment el F1-score "macro"per a l'avaluació, ja que aquest assigna la mateixa importància a totes les classes.

2.1 Balancejat de classes

Primerament, donat que la freqüència de les classes està molt desbalancejada, apliquem un balanceig, calculant un pes per cada classe i afegint-lo al fit del model. A continuació, comparem la diferència de resultats entre aplicar el balancejat o no fer-ho:

Mètode	**Accuracy**	**F1-score macro**
Sense balanceig	0.936667	0.539805
Amb balanceig	0.881111	0.564676

Taula 1: Resultats amb i sense balanceig

Com es pot veure, el balanceig és efectiu, ja que millora el F1-score de 0.54 a 0.56. Malgrat que l'accuracy disminueixi lleugerament amb el balanceig (de 0.94 a 0.88), donem prioritat al F1-score "macro"perquè aquest proporciona una mesura més robusta en aquest context, reflectint l'eficàcia del model en totes les classes. Per aquest motiu mantindrem el balanceig en els propers apartats.

2.2 Preprocessament

En aquesta segona fase, s'ha creat una estructura per provar diferents combinacions de paràmetres en el preprocesament de dades.
En primer lloc, es defineixen els diversos processos de preprocessament dels quals es provarà l'efectivitat:
- **Lowercase**: converteix tot el text a minúscules per reduir la variabilitat de les paraules.
- **Lemmatize**: redueix les paraules a la seva forma base, ajudant a unificar termes similars.
- **Stem**: elimina els sufixos de les paraules, reduint-les a la seva arrel comuna.
- **Stop Words**: elimina paraules poc informatives com articles o preposicions.
- **Vocab Size**: limita el nombre de paraules únic considerades pel model.

En segon lloc, s'han generat totes les combinacions possibles amb aquests paràmetres (amb les opcions d'incloure o no incloure'ls) i s'ha executat el model per a cada una d'aquestes combinacions. Finalment, s'han comparat els resultats per a cada combinació, escollint la més adequada. En el notebook es pot trobar la taula completa però a continuació es presenten els cinc resultats més exitosos.

Lowercase	Lematize	Stem	StopWords	VocabSize	Accuracy	F1-score
True	**False**	**False**	**False**	**20000**	**0.883333**	**0.663590**
False	False	False	False	5000	0.852222	0.655276
False	False	False	False	10000	0.855556	0.646920
True	False	False	False	5000	0.858889	0.611001
False	False	False	False	10000	0.861111	0.590782

Taula 2: Resultats de diferents configuracions de preprocessament

Els resultats mostren que la configuració que proporciona el millor rendiment és la següent:
- Convertir a minúscules.
- Sense aplicar lematització ni stemming.
- Sense eliminar paraules comunes (stop words).
- Utilitzant un vocabulari de 20,000 paraules.

Amb aquesta configuració, el model assoleix una **val_accuracy** del 88.33% i una **F1-score** de 0.66359. Aquests resultats són els més alts de totes les combinacions, demostrant així que mantenir el text en la seva forma original beneficia la captura de la riquesa i les variacions en les dades.

2.3 Mida dels embeddings

En aquesta fase, s'ha analitzat l'impacte de la mida dels embeddings en la precisió del model. Els embeddings són vectors que representen les paraules en l'espai i, per tant, la seva mida pot afectar la capacitat del model per capturar relacions entre paraules.
A continuació, es presenta la taula amb els resultats obtinguts per a totes les diferents dimensions d'embeddings provades:

Embedding Dim	Accuracy	F1-Score
500	**0.918889**	**0.713261**
150	0.902222	0.691711
100	0.870000	0.624318
200	0.910000	0.612682
50	0.844444	0.608882

Taula 3: Resultats per a diferents mides d'embeddings

Els resultats indiquen que la millor configuració s'obté amb una mida d'embeddings de **500 dimensions**. Aquests resultats mostren una capacitat elevada del model per capturar relacions semàntiques complexes, ja que una mida més gran dels embeddings permet representar les paraules amb major detall. Tot i que les dimensions més petites ofereixen resultats propers, especialment amb 150 i 100 dimensions, es decideix mantenir la configuració de 500 per a maximitzar el rendiment del model.

2.4 Xarxes Neuronals

S'han utilitzat diferents arquitectures de xarxes convolucionals i recurrents, adaptant els paràmetres i la complexitat, per tal de trobar la configuració òptima.

2.4.1 Xarxes Convolucionals

En aquesta secció, s'han provat sis configuracions de capes convolucionals, cadascuna amb diferents capes i valors per a filtres, mida del kernel i capes de pooling, per veure com afecten al model.

A continuació es mostra una breu explicació i justificació de cada configuració. Cal aclarir que totes aquestes configuracions van sempre precedides d'una capa d'embedding i seguides de dues capes denses al final, a part de les diferents capes que proven cada una.

- **Configuració 1**
  Model base sense capes convolucionals, només capa d'embedding i capes denses.
- **Configuració 2**

```
1    model.add(Conv1D(filters=128, kernel_size=5,
         activation='relu'))
```

  Una sola capa convolucional per capturar patrons bàsics, seguida de MaxPooling per reduir la dimensionalitat.

- **Configuració 3**

```
1    model.add(Conv1D(filters=64, kernel_size=3,
         activation='relu'))
2    model.add(MaxPooling1D(pool_size=2))
3    model.add(Conv1D(filters=128, kernel_size=5,
         activation='relu'))
4    model.add(MaxPooling1D(pool_size=2))
```

  Dues capes convolucionals amb mides de kernel diferents per capturar patrons locals i més amplis.

- **Configuració 4**

```
1    model.add(Conv1D(filters=32, kernel_size=7,
         activation='relu'))
2    model.add(MaxPooling1D(pool_size=2))
3    model.add(Conv1D(filters=64, kernel_size=5,
         activation='relu'))
4    model.add(MaxPooling1D(pool_size=2))
5    model.add(Conv1D(filters=128, kernel_size=3,
         activation='relu'))
6    model.add(MaxPooling1D(pool_size=2))
```

  Increment progressiu de filtres i disminució de la mida del kernel per captar patrons multi-escala.

- **Configuració 5**

```
1    model.add(Conv1D(filters=128, kernel_size=5,
         activation='relu'))
2    model.add(GlobalAveragePooling1D())
```

Una capa convolucional seguida de GlobalAveragePooling per capturar patrons clau de la seqüència de forma eficient.

- **Configuració 6**

```
1    model.add(Conv1D(filters=64, kernel_size=3,
         activation='relu'))
2    model.add(MaxPooling1D(pool_size=2))
3    model.add(Conv1D(filters=128, kernel_size=5,
         activation='relu'))
4    model.add(GlobalAveragePooling1D())
```

Combina capes convolucionals i GlobalAveragePooling per capturar patrons locals i resumir la informació general.

Al final de cada configuració es realitza una capa *GlobalMaxPooling* i dues capes *Dense*.
Les configuracions han mostrat els següents resultats:

Configuració	Accuracy	F1 Score
2	**0.971111**	**0.823319**
1	0.920000	0.720262
3	0.934444	0.694650
4	0.822222	0.512057
5	0.666667	0.460982
6	0.712222	0.446635

Taula 4: Resultats per a diferents configuracions convolucionals

Com podem veure a la taula, **la configuració 2, tot i ser una estuctura molt senzilla** que consisteix en una capa convolucional amb 128 filtres i un kernel de mida 5 seguida d'una capa *MaxPooling*, **és la que ens ofereix el millor rendiment**. Això indica que el model pot capturar patrons sense afegir complexitat innecessària.

2.4.2 Xarxes Recurrents

En aquesta fase, s'han provat diverses configuracions amb capes recurrents com per exemple les LSTM i GRU. Recordem que les xarxes recurrents són especialment adequades per a dades seqüencials, com el text, ja que poden mantenir informació contextual a llarg termini. Les configuracions provades són les següents:

- **Configuració 1**

```
    model.add(LSTM(128))
```

Aquesta configuració utilitza una sola capa LSTM amb 128 unitats per capturar relacions seqüencials bàsiques.

- **Configuració 2**

```
    model.add(GRU(128))
```

Utilitza una sola capa GRU amb 128 unitats, una arquitectura més lleugera que LSTM, reduint el cost computacional però mantenint una bona capacitat per capturar seqüències.

- **Configuració 3**

```
    model.add(LSTM(128, return_sequences=True))
    model.add(LSTM(64))
```

Inclou múltiples capes LSTM per capturar tant relacions de llarg abast (128 unitats) com detalls locals (64 unitats), oferint una representació més rica de la seqüència.

- **Configuració 4**

```
    model.add(LSTM(128, return_sequences=True))
    model.add(GRU(64))
```

Combina capes LSTM i GRU, aprofitant la capacitat de l'LSTM per capturar dependències llargues, mentre que la GRU ofereix una computació més eficient per patrons curts.

- **Configuració 5**

```
    model.add(Conv1D(64, kernel_size=3, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(LSTM(128))
```

Combina una capa convolucional seguida d'una LSTM: la convolució captura primer característiques locals, i la LSTM després captura relacions temporals, ideal per seqüències llargues.

- **Configuració 6**

```
    model.add(Bidirectional(LSTM(128, return_sequences=True)))
    model.add(Bidirectional(LSTM(64)))
```

Utilitza dues capes LSTM bidireccionals. La primera capa, amb 128 unitats i 'return_sequences=True', captura les relacions temporals en ambdues direccions mantenint les seqüències per a la següent capa. La segona capa, amb 64 unitats, resumeix la informació seqüencial, permetent al model captar patrons en seqüències complexes i millorar la comprensió contextual.

- **Configuració 7**

```
    model.add(Conv1D(64, kernel_size=3, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Bidirectional(LSTM(128)))
```

  Combina una capa convolucional amb un LSTM bidireccional. La capa convolucional amb 64 filtres i un kernel size de 3 capta les característiques locals de les seqüències, mentre que el MaxPooling redueix la dimensionalitat. La capa Bidirectional LSTM amb 128 unitats capta relacions temporals en ambdues direccions, millorant la comprensió contextual i permetent un processament més profund de les seqüències.

Els resultats d'aquestes configuracions es mostren a la taula següent:

Configuració	Accuracy	F1 Score
6	0.920000	0.662024
7	0.877778	0.601870
2	0.030000	0.003434
3	0.006667	0.000779
4	0.001111	0.000124
1	0.000000	0.000000
5	0.000000	0.000000

Taula 5: Resultats per a diferents configuracions de xarxes recurrents

Com podem veure a la taula, la configuració 6, amb LSTMs bidireccionals, és la que proporciona millors resultats oferint la millor precisió i generalització.

2.4.3 Configuració escollida en les xarxes neuronals

De totes les configuracions provades tant amb capes convolucionals com recurrents, **s'ha escollit la configuració 2 de les capes convolucionals** per diversos motius. En primer lloc, aquesta configuració ha demostrat obtenir el millor resultat en termes de *accuracy* amb un valor de 0.957778 i F1-score de 0.796809, superant en els dos valors a les altres configuracions.
En segon lloc, aquesta configuració proporciona un bon equilibri entre complexitat i eficiència. Amb només una capa convolucional i una capa de max pooling, el model és relativament senzill en comparació amb les configuracions més complexes. Per aquest motiu, la configuració escollida és ideal per capturar patrons bàsics en seqüències de text sense afegir capes addicionals que podrien sobreajustar les dades o no aportar beneficis significatius.

2.5 Regularització

Per evitar problemes de *overfitting*, s'han provat diferents configuracions de regularització, utilitzant la tècnica de **Dropout** en diferents parts de la xarxa. Aquestes configuracions permeten avaluar com l'ús de *Dropout* influeix en la capacitat de generalització del model.
Totes les configuracions han estat provades amb la configuració escollida anteriorment a l'apartat de convolucionals i recurrents. Les configuracions de *Dropout* provades són les següents:

- **Configuració 1**

```
    model.add(GlobalMaxPooling1D())
```

Sense regularització, per observar el rendiment base del model sense efectes de *Dropout*.

- **Configuració 2**

```
    model.add(Dropout(0.2))
    model.add(MaxPooling1D(pool_size=2))
```

Aplicació de *Dropout* lleuger (0.2) després de la capa convolucional per reduir lleugerament el sobreajustament.

- **Configuració 3**

```
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3))
```

Dropout més intens (0.3) després de la capa de *MaxPooling* per proporcionar regularització extra després de reduir la dimensionalitat.

- **Configuració 4**

```
    model.add(Dropout(0.3))
    model.add(MaxPooling1D(pool_size=2))
    model.add(GlobalMaxPooling1D())
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
```

Combinació de *Dropout* després de la capa convolucional (0.3) i en les capes denses (0.5) per a una regularització completa, evitant sobreajustament tant en característiques inicials com en les més avançades.

- **Configuració 5**

```
    model.add(Dropout(0.5))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.5))
    model.add(GlobalMaxPooling1D())
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
```

Dropout agressiu (0.5) aplicat en diverses capes per reduir dràsticament el sobreajustament en models complexos.

Els resultats d'aquestes configuracions es mostren a la taula següent, ordenats per *val_accuracy* de manera descendent:

Configuració	Accuracy	F1 Score
1	**0.964444**	**0.820183**
2	0.950000	0.754904
3	0.918889	0.728809
4	0.866667	0.533787
5	0.555556	0.325004

Taula 6: Resultats per a diferents configuracions de regularització amb *Dropout*

En el cas de la regularització, **s'ha escollit la configuració 1**, és a dir, el model que ja teniem prèviament sense afegir cap canvi. Tot i haver provat altres configuracions amb *Dropout*, s'ha pogut observar que no millora el rendiment del model. El model base ja presenta una bona capacitat de generalització, sense realitzar overfitting, a més, la regularització empitjora el rendiment del model. Per tant, s'ha decidit mantenir la configuració anterior.

3 Model final

El model final consisteix en una xarxa neuronal convolucional que combina una capa d'embeddings, una capa convolucional (Conv1D) amb filtres de 128 i un kernel de mida 5, una capa de pooling (GlobalMaxPooling1D) i dues capes denses (una amb 64 unitats i l'activació ReLU).

	Accuracy	F1-Score
Validació	0.971111	0.823319
Test	0.930180	0.670503

Taula 7: Resultats d'Accuracy i F1-Score per a Validació i Test

Tal com podem veure, aquesta arquitectura demostra ser eficaç per a la tasca de classificació d'intencions en textos, ha aconseguint un accuracy del 93.02% amb 20 èpoques en el conjunt de test, és a dir, en dades no vistes prèviament. Pel que fa al F1-score en canvi, redueix considerablement en comparació amb el conjunt de validació, cosa que podria indicar que s'ha produit un sobreajustament a les dades de validació.

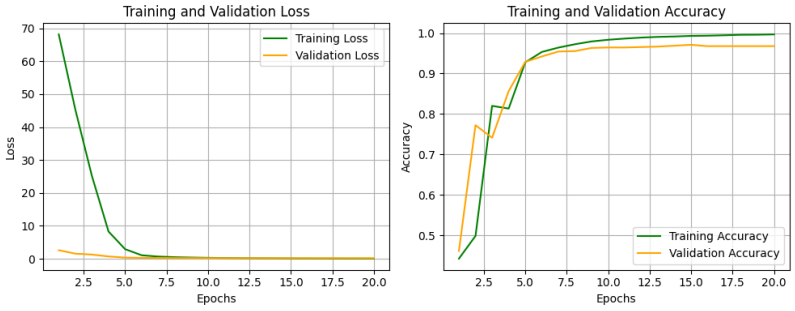

Figura 1: Corbes

Les corbes mostren una clara convergència del model, amb una disminució ràpida de la pèrdua (loss) tant en entrenament com en validació. A més a més, no hi ha un overfitting visible en l'accuracy.

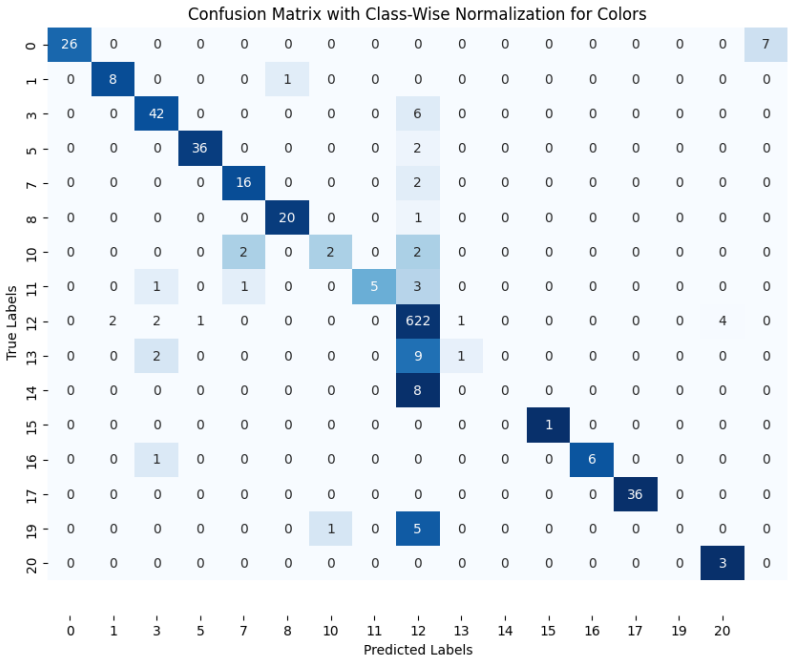

Figura 2: Matriu de Confusió

D'altra banda, la matriu de confusió permet observar el desbalanceig de les classes, i com el model comet errors en les classes amb menys instàncies, predint les etiquetes amb més instàncies (concretament passa en les classes 13, 14 i 19). Aquest és un problema comú en dades desbalancejades i caldria més instàncies en les classes amb poca representació per mitigar-lo.

Podem concloure que malgrat el sobreajustament produit a partir de la mètrica de F1 Score "macro", el model és capaç de reconèixer correctament la intenció de l'usuari en la majoria de casos.

4 Conclusions

En aquesta pràctica s'ha aconseguit dissenyar i experimentar un model de classificació d'intencions per a la classificació de textos, amb l'objectiu de millorar la seva precisió i generalització en identificar les intencions de l'usuari. S'han realitzat diversos experiments per tal de millorar progressivament la precisió del model, avaluant cada millora mitjançant l'F1-score de tipus macro utilitzant el conjunt de validació. Assegurant així una mesura equilibrada i consistent del rendiment global.

Tot i les diverses millores probades, algunes d'elles com l'aplicació de regularització amb *Dropout* o l'augment de complexitat de les xarxes neuronals amb múltiples capes, no van resultar eficients ja que no van aportar cap tipus de millora significativa o, fins i tot, van empitjorar el model. En canvi, s'ha observat que un model més senzill ha proporcionat els millors resultats en termes de precisió i generalització.

En general, podem dir que aquesta tasca de classificació d'intencions és sensible a l'elecció d'arquitectures i tècniques d'entrenament. A més, en aquest cas, la poca representació en algunes classes ha comportat errors en la predicció del model final. No obstant això, el model ha demostrat ser capaç de capturar els patrons clau en les dades, aconseguint un rendiment eficient en la tasca proposada.