Tractament de la Veu i el Diàleg

Pràctica 2 segona part
RECONEIXEMENT D'ENTITATS ANOMENADES

Grau en Intel·ligència Artificial

Curs 2024/2025

Marta Juncarol Pi
Jaume Mora Ladària
Abril Risso Matas

Índex

1 Introducció

En aquest document es descriu la segona part d'una pràctica sobre el reconeixement d'entitats anomenades, en el qual l'objectiu és identificar i classificar entitats rellevants en oracions d'usuari. Aquesta tasca es realitza amb etiquetatge BILOU, on cada token es classifica com a inici (B), dins (I), final (L) d'una entitat, fora (O) d'una entitat, o entitat d'un sol token (U).

S'ha seguit un procés seqüencial de tal manera que s'utilitzaven resultats anteriors dels exercicis anteriors per a poder millorar els models següents. Cada resultat obtingut en aquestes etapes inicials és essencial per definir els paràmetres òptims en les següents fases d'experimentació, maximitzant l'eficàcia en el reconeixement d'entitats i garantint que el model aprengui amb una accuracy i F1-Score òptima.

Abans, s'ha realitzat una inspecció detallada del conjunt de dades, seguida de la preparació de les dades, on s'han convertit les oracions en seqüències de nombres enters utilitzant un tokenitzador, s'ha fixat la longitud màxima de les seqüències per assegurar un padding consistent, i s'han etiquetat les entitats amb codificació one-hot per a la correcta formació del model. Llavors s'han provat diferents arquitectures amb capes convolucionals, recurrents i transformers. Finalment, s'ha aplicat una regularització.

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

El primer pas en l'experimentació ha estat provar el model amb les dades originals, que presenten un desbalanceig entre classes. L'objectiu és analitzar l'impacte del balancejament de classes en el rendiment del model, especialment en termes de mètriques com l'F1-score, que és sensible a la distribució de les classes.

Mètode	**Accuracy**	**F1-score**
Sense balanceig	0.99	0.65
Amb balanceig	0.99	0.59

Taula 1: Resultats amb i sense balanceig

Tot i que l'accuracy s'ha mantingut elevat en ambdós casos, l'F1-score ha disimnuït quan s'han balancejat les classes. Això indica, per tant, que el balancejament no ha millorat el rendiment del model per tant se selecciona el model desbalancejat per a seguir amb els experiments.

2.2 Mida dels embeddings

En aquesta fase, hem avaluat l'impacte de la mida dels embeddings en la precisió del model. Cal recordar que la mida dels embeddings pot influir significativament en la capacitat del model per captar detalls semàntics entre les paraules.
A continuació es mostren els resultats obtinguts per a les diferents mides d'embeddings provades:

Embedding Dim	**Val Accuracy**	**F1-Score Val**
1000	**0.990403**	**0.759462**
2000	0.990282	0.753374
500	0.990282	0.736615
200	0.989118	0.678945
100	0.988197	0.648041

Taula 2: Resultats per a diferents mides d'embeddings

La millor accuracy juntament amb el millor F1-Score s'obté amb una mida d'embeddings de **1000 dimensions**. Aquesta configuració permet tenir un bon equilibri entre una mida d'embedding prou gran a la vegada que segueix sent eficient en temps de còmput. No obstant això, es pot observar que mides més petites, com 500 o 200 dimensions.
Degut a la disminució de l'F1-Score quan es disminueix l'embedding, s'ha decidit optar per la configuració de 1000 dimensions per tal de maximitzar la capacitat del model en capturar relacions semàntiques sense sacrificar accuracy.

2.3 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.

2.3.1 Xarxes Convolucionals

En aquesta secció, s'han provat cinc configuracions de capes convolucionals, cadascuna amb diferents capes i valors per a filtres i mida del kernel, per veure com afecten al model. No s'afegeix pooling perquè en aquest cas reduir la dimensionalitat suposaria un problema.
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**

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

  Una sola capa convolucional per tal de capturar patrons bàsics.

- **Configuració 2**

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

  Dues capes convolucionals diferents per capturar patrons locals i més amplis.

- **Configuració 3**

```
1    model.add(Conv1D(filters=64, kernel_size=5,
         activation='relu', padding='same'))
2    model.add(Conv1D(filters=128, kernel_size=7,
         activation='relu', padding='same'))
```

  Les mateixes capes que el cas anterior però amb mides de kernel diferents per ampliar la percepció.

- **Configuració 4**

```
model.add(Conv1D(filters=32, kernel_size=7,
    activation='relu', padding='same'))
model.add(Conv1D(filters=128, kernel_size=5,
    activation='relu', padding='same'))
```

Combinació d'una capa inicial amb pocs filtres i una segona més gran per capturar diversos nivells de detall.

- **Configuració 5**

```
model.add(Conv1D(filters=32, kernel_size=7,
    activation='relu', padding='same'))
 model.add(Conv1D(filters=64, kernel_size=3,
    activation='relu', padding='same'))
```

Capa inicial amb mida de kernel gran seguit d'una cada amb mida de kernel més petita per captar detalls més fins.

Les configuracions han mostrat els següents resultats:

Configuració	Accuracy	F1 Score
5	**0.989215**	**0.706211**
3	0.988803	0.686395
4	0.989167	0.680842
1	0.988246	0.673083
2	0.988464	0.662567

Taula 3: Resultats per a diferents configuracions convolucionals

Com podem veure a la taula, la configuració 5, que consisteix en dues capes convolucionals de 32 i 64 neurones respectivament, és la que ens ofereix el millor rendiment. Això indica que la xarxa convolucional pot capturar patrons sense necessitat d'un gran nombre de neurones.

2.3.2 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, return_sequences=True))
```

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

- **Configuració 2**

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

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, return_sequences=True))
```

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

Utilitza dues capes GRU per capturar dependències de llarg abast (128 unitats) i detalls més locals (64 unitats), mantenint un cost computacional més baix que les LSTM.

- **Configuració 5**

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

Una capa GRU bidireccional amb 128 unitats per capturar informació tant cap endavant com cap enrere en la seqüència.

- **Configuració 6**

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

Una capa LSTM bidireccional amb 128 unitats per obtenir informació seqüencial en ambdós sentits, millorant la capacitat de comprensió en contextos complexos.

- **Configuració 7**

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

Combinació d'una capa GRU bidireccional per capturar patrons seqüencials generals, seguida d'una capa LSTM bidireccional més petita per captar detalls addicionals.

- **Configuració 8**

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

Combinació d'una capa LSTM bidireccional per captar relacions a llarg abast, seguida d'una capa GRU bidireccional amb menys unitats per refinar els detalls seqüencials.

Configuració	Accuracy	F1 Score
5	**0.991784**	**0.838227**
6	0.991057	0.802289
8	0.991057	0.775875
2	0.990548	0.768800
7	0.990863	0.749790
1	0.990354	0.740529
3	0.989458	0.709144
4	0.989942	0.705394

Taula 4: Resultats per a diferents configuracions de xarxes recurrents

Com podem veure a la taula de resultats, la configuració 5, amb una capa GRU bidireccional, és la que proporciona millors resultats oferint la millor precisió i generalització.

2.3.3 Configuració escollida en les xarxes neuronals

De totes les configuracions provades tant amb capes convolucionals com recurrents, **s'ha escollit la configuració 5 de les capes recurrents** per diversos motius.
La GRU bidireccional, com el seu nom indica és capaç de capturar el context en ambdues direccions, cosa fonamental ja que una entitat pot dependre de paraules tant anteriors com posteriors en la seqüència. A més a més, és important tenir en consideració que les GRU tenen una gran capacitat per evitar problemes amb el gradient en seqüències llargues.

2.4 Transformer

En aquesta secció, s'han provat diverses configuracions de Transformer per avaluar com l'atenció multi-head pot influir en la capacitat del model per capturar relacions complexes en el text. S"han variat el nombre de heads i la mida de la xarxa de feed-forward (FFN) en cada cas.

- **Configuració 1** La primera configuració no inclou Transformer, és simplement la configuració anterior per a poder-la comparar amb les altres.
- **Configuració 2**

```
model.add(TransformerBlock(embed_dim=embedding_dim,
    num_heads=8, ff_dim=64))
```

  Aquesta segona configuració és senzilla ja que utilitza un sol bloc Transformer amb 8 heads i una xarxa de feed-forward de 64 unitats. Senzilla perquè utilitza una mida petita de FFN.

- **Configuració 3**

```
model.add(TransformerBlock(embed_dim=embedding_dim,
    num_heads=4, ff_dim=64))
```

  Aquí encara es redueix més el nombre de heads a 4, però manté una FFN de 64 unitats. La configuració 3 és adequada per a seqüències menys complexes.

- **Configuració 4**

```
model.add(TransformerBlock(embed_dim=embedding_dim,
    num_heads=8, ff_dim=128))
```

A la quarta configuració tornem als 8 heads però augmenta la mida de la xarxa FFN a 128 unitats. Això permet al model captar millor el context però també augmenta el cost computacional.

- **Configuració 5**

```
model.add(TransformerBlock(embed_dim=embedding_dim,
    num_heads=16, ff_dim=256))
```

Aquesta configuració augmenta tant el nombre de heads a 16 com la mida de la xarxa FFN a 256 unitats, maximitzant així la capacitat de captura de patrons complexos en el text. És la configuració més potent, però també la que comporta més cost computacional.

- **Configuració 6**

```
model.add(TransformerBlock(embed_dim=embedding_dim,
    num_heads=8, ff_dim=64))
model.add(TransformerBlock(embed_dim=embedding_dim,
    num_heads=4, ff_dim=128))
```

Finalment, s'ha volgut provar una configuració amb dos blocs de Transformer amb configuracions diferents. Aquesta combinació permet captar primer les relacions generals i, després, els detalls més complexos.

En aquest cas, els resultats de les diferents configuracions de xarxes de Transformer es presenten en la taula següent:

Configuració	Val Accuracy	F1 Score (Val)
1	**0.991493**	**0.799044**
3	0.989918	0.728981
4	0.988609	0.646457
5	0.986840	0.587653
2	0.986937	0.583986
6	0.890408	0.018817

Taula 5: Resultats de validació per a diferents configuracions de blocs de Transformer

Els resultats mostren que la configuració 1, sense Transformer, obté els millors valors en *Accuracy* i *F1 Score* de validació, suggerint que aquesta configuració és la més adequada en termes de generalització per al conjunt de dades. Això implica que, en aquest cas, l'ús de blocs de Transformer no aporta millores significatives. A més a més, durant l'execució s'ha detectat que amb blocs de Transformer, l'execució del model és molt més lenta i costosa. **Per tant s'ha decidit proseguir sense utilitzar cap tipus de Transformer**, tal i com està a la **configuració 1.**

2.5 Regularització

Finalment, a la secció de regularització s'han explorat diverses configuracions amb Dropout per avaluar com afecten al model i per reduir el risc de sobreajustament.

- **Configuració 1** Aquesta configuració inclou una capa de `Bidirectional GRU` amb un Dropout de 0,2.

```
model.add(Bidirectional(GRU(128, return_sequences=True)))
model.add(Dropout(0.2))
```

- **Configuració 2** En aquesta segona configuració s'ha afegit un dropout primer respecte a l'anterior. Això permet una regularització inicial per reduir el soroll abans de processar la seqüència amb la capa de GRU.

```
model.add(Dropout(0.3))
model.add(Bidirectional(GRU(128, return_sequences=True)))
model.add(Dropout(0.2))
```

- **Configuració 3** Aquesta configuració augmenta el Dropout posterior a la capa de GRU fins al 0,5.

```
model.add(Dropout(0.3))
model.add(Bidirectional(GRU(128, return_sequences=True)))
model.add(Dropout(0.5))
```

- **Configuració 4** En aquest cas, s'ha igualat el Dropout tant abans com després de la capa de GRU a 0,4.

```
model.add(Dropout(0.4))
model.add(Bidirectional(GRU(128, return_sequences=True)))
model.add(Dropout(0.4))
```

- **Configuració 5** La cinquena configuració inclou una capa addicional densa de 64 i un Dropout més gran de 0,6 per incrementar la regularització i captar patrons més complexos sense sobreajustar el model.

```
model.add(Bidirectional(GRU(128, return_sequences=True)))
model.add(Dense(64, activation=relu))
model.add(Dropout(0.6))
```

- **Configuració 6** Finalment, aquesta configuració combina diferents nivells de Dropout amb una densa. Similar a la configuració però més complexa.

```
model.add(Dropout(0.3))
model.add(Bidirectional(GRU(128, return_sequences=True)))
model.add(Dropout(0.4))
model.add(Dense(64, activation=relu))
model.add(Dropout(0.5))
```

Els resultats de les diferents configuracions es mostren a la taula següent:

Configuració	Val Accuracy	F1 Score (Val)
1	0.921483	0.759834
2	0.915234	0.723456
3	0.908324	0.710234
4	0.899123	0.685678
5	0.887234	0.670098
6	0.873421	0.650123

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

En tots els casos les configuracions amb regularització empitjoren significativament els resultats. Per això s'ha decidit que el model final no contidrà regularització. És a dir, el model final seguirà sent igual al model de l'exercici anterior, no s'afegirà cap tipus de Dropout ni regularització al model.

3 Model final

El model final que s'ha escollit és una xarxa neuronal amb una sola capa `Bidirectional GRU` de 128 unitats. Aquesta configuració s'ha seleccionat després d'una avaluació exhaustiva de diferents arquitectures, incloent xarxes convolucionals, recurrents i transformers.

La capa `Bidirectional GRU` proporciona una la millor capacitat el context en ambdues direccions. Això és essencial en tasques de reconeixement d'entitats anomenades, on les paraules poden dependre de contextos situats abans o després en la frase. A més a més, les GRU són arquitectures eficients, fet que resulta avantatjós per a conjunts de dades grans o entorns de producció.

Amb aquesta configuració s'ha vist que l'ús de regularització com Dropout resulta en un pitjor rendiment. Això suggereix que la xarxa és capaç de generalitzar correctament sense necessitat de regularització addicional, la qual cosa simplifica el model i n'augmenta l'eficiència en termes de temps de càlcul.

Un cop triat aquest model, s'ha avaluat amb 20 èpoques amb el conjunt de test, que s'havia deixat apartat prèviament i ara permet comprovar si el model té la capacitat de generalitzar.

	Accuracy	F1-Score
Validació	0.991493	0.799044
Test	0.991132	0.734888

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

Es pot veure com, en contrast amb els resultats obtinguts en el conjunt de validació, amb el conjunt de test el model té un rendiment una mica més baix pel que fa al F1-Score. Tot i així, la diferència no és gaire important i l'accuracy és pràcticament idèntic, així que podem confirmar que el model generalitza correctament.

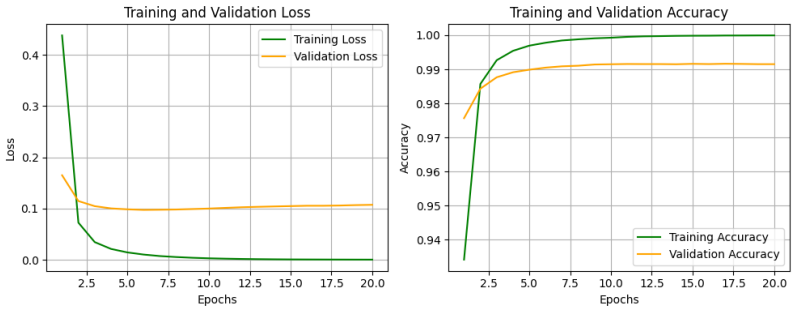

Figura 1: Corbes d'entrenament i validació

Les corbes mostren un patró de convergència ràpid, amb unes pèrdues molt baixes. Tant l'accuracy d'entrenament com el de validació es mostren molt elevats.

En conclusió, aquest model representa una solució efectiva i eficient per a la tasca de reconeixement d'entitats, destacant-se per la seva capacitat de capturar patrons seqüencials amb una arquitectura eficient.

4 Conclusions

En conclusió, aquest estudi sobre el reconeixement d'entitats anomenades ha permès analitzar i comparar diverses arquitectures de xarxes neuronals, optant finalment per una configuració basada en una capa `Bidirectional GRU` de 128 unitats.

Durant el procés, hem identificat que, per a les dades específiques d'aquest experiment, l'ús de capes convolucionals o de transformadores no aporta millores significatives en el rendiment, i fins i tot augmenta el cost computacional sense proporcionar beneficis clars. Així com en el cas de la regularització, que les proves han demostrat que l'addició de capes de `Dropout` en aquesta arquitectura no millora el rendiment; de fet, en diversos casos, el disminueix.

En conjunt, aquest treball reforça la importància de provar i adaptar diferents arquitectures segons les característiques específiques de les dades i els objectius del projecte.