Universitat de Girona

ESCOLA POLITÈCNICA SUPERIOR

PROJECTE GEC

Tècniques Avançades d'Intel·ligència Artificial

JOSEP LLUÍS ARCÓS ROSELL
MARIA BEATRIZ LÓPEZ IBAÑEZ

Iuri Pons Frago, u1956242

Índex

1 Introducció

Aquest projecte de Tècniques Avançades d'Intel·ligència Artificial tracta sobre l'aprenentatge d'un corrector d'errors gramatical (*CEG*; o *GEC* en anglès *Grammatical Error Correction*.

Aquests correctors, usualment, treballen amb llenguatges naturals creats per l'ésser humà. La disciplina de la informàtica que estudia i analitza els llenguatges naturals és el processament del llenguatge natural (*PLN*; o *NLP* del seu nom en anglès, *Natural Language Processing*).

Partint de la premissa que els models *GEC* són costosos d'entrenar, sigui per temps o per capacitat computacional, i nosaltres fem servir una computadora de gamma mitja, el nostre objectiu és entendre, analitzar i aprofundir en aquesta disciplina.

Abans d'entrar en matèria, per entendre millor el conceptes que treballarem a continuació, hauríem de fer una reflexió sobre què comporta crear un GEC. Com sabem tots, el llenguatge d'una màquina és el binari. Però, la tasca que volem resoldre es basa en un llenguatge completament diferent.

Cada NLP funciona diferent, tenen diferents alfabets, s'estructuren diferent i, inclús, l'ordre d'escriptura i lectura són diferents. Tot i tenir aquestes diferències, tots comparteixen una característica molt important, segueixen un conjunt de regles. El problema és que cada llenguatge té moltes regles i cada regla, normalment, té excepcions. A part, el context i l'ambigüitat són molt rellevant i són aquests motius pels quals es fa impossible, o molt tediós, codificar un sol llenguatge humà.

Tot i que hi ha molts programadors que no han descartat el fet de programar totes les regles d'un NLP, aquí només ens centrarem en la potència del *machine learning*. Concretament, farem servir una metodologia que fa molt de temps que s'està investigant i desenvolupant que són les xarxes neuronals.

2 Xarxes Neuronals Artificials

Una xarxa neuronal artificial (*XNA*), també anomenada xarxa neuronal simulada o senzillament xarxa neuronal (denominada habitualment en anglès com *ANN*) és un paradigma d'aprenentatge i processament automàtic inspirat en la forma en què funciona el sistema nerviós dels animals.

Es tracta d'un sistema d'interconnexió de neurones en una xarxa que col·labora per produir un estímul de sortida. Aquest conjunt de neurones artificials interconnectades utilitza un model matemàtic o computacional de processament de dades basat en una aproximació connexionista per a la computació.

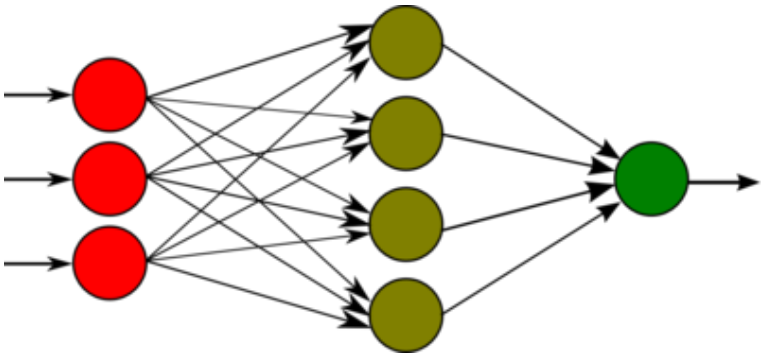

Figura 1: Xarxa neuronal artificial amb 3 neurones d'entrada, 4 neurones en la seva capa oculta i una neurona de sortida.

Sense entrar en detalls del funcionament tan complex que tenen les xarxes neuronals, podem destacar que l'objectiu d'aquestes és donar una sortida respecte als paràmetres d'entrada introduïts mitjançant una selecció de paràmetres i càlculs matemàtics.

També cal saber els avantatges que tenen aquestes xarxes, per saber si realment són una bona eina per resoldre el nostre problema:

- Aprenentatge: Tenen l'habilitat d'aprendre mitjançant una etapa que s'anomena *etapa d'aprenentatge*. Aquesta ho fa sobre un conjunt de dades d'entrada i les compara amb les dades de sortida que se li ha indicat prèviament.
- Auto organització: Una xarxa neuronal crea la seva pròpia representació de la informació al seu interior, sense que el programador se n'hagi de preocupar.
- Tolerància a fallades: Atès que una xarxa neuronal emmagatzema la informació de forma redundant, aquesta pot seguir responent de manera acceptable fins i tot si es fa malbé parcialment.
- Flexibilitat: Una xarxa neuronal pot gestionar canvis no importants en la informació d'entrada (ex. si la informació d'entrada és la imatge d'un objecte, la resposta corresponent no pateix canvis si la imatge canvia una mica la seva brillantor o l'objecte canvia lleugerament).

- Temps real: L'estructura d'una xarxa neuronal és paral·lela, per la qual cosa si això és implementat amb ordinadors o en dispositius electrònics especials, es poden obtenir respostes en temps real.
- Escalabilitat: poden adaptar-se a qualsevol problema d'una determinada àrea.

Veient tots els avantatges que tenen les *XNA*, arribem a la conclusió que efectivament són una molt bona eina per dur a terme la resolució del nostre problema. Però, hi ha un punt molt important que s'ha comentat a la introducció. Com que les xarxes neuronals artificials estan implementades en els computadors, aquestes també treballen amb dades numèriques i nosaltres volem tractar text.

Quan es tracta de treballar amb imatges, àudios o simplement amb dades numèriques tenim molt clar com fer el tractament de dades si cal. Però quan es tracta d'un text i l'enteniment d'aquest no és tan senzill.

3 Representació Numèrica d'un Text

Arran del problema que hem trobat anteriorment, ens hem de posar a pensar com podem representar un text numèricament. Segurament, una de les primeres solucions que sorgeixen a qualsevol persona és assignar un nombre a cada paraula. Aquest ja pot ser a l'atzar o per ordre, però és important que cada paraula tingui un nombre diferent i les paraules amb el mateix lexema comparteixin nombre.

Així doncs el següent text el podríem representar de la següent manera:

El pare està amb el seu fill.
25 15 32 65 25 8 3

Figura 2: Text representat numèricament amb nombres escollits a l'atzar.

Ara ja hem pogut resoldre el problema de la representació numèrica d'un text, però és el correcte. Com millor funcionen els sistemes d'aprenentatges, sigui artificial o humà, és mitjançant la lògica i la correlació entre les dades. Si nosaltres volem fer entendre a una màquina un llenguatge humà, doncs ha de saber que les paraules tener relació i similitud.

Segons la representació numèrica de la figura 2, veiem que la paraula *està* té un valor que duplica el valor de la paraula *pare*. I la diferència entre *pare* i *fill* és més gran que la diferència entre *fill* i *seu*, cosa que no hauria de ser així.

Arribem a la conclusió, que per representar un conjunt de paraules no podem fer servir nombre a l'atzar. Doncs la solució que es va proposar fa uns anys és representar cada paraula vectorialment. Enlloc de tenir un valor per cada paraula, tindrem un vector de mida m.

Per evitar que les paraules quedin relacionades de manera incorrecte, farem que el vector sigui tan llarg com paraules hi hagi en el nostre vocabulari, així doncs $m = midadelvocabulari$. Per exemple, si tinguéssim tres paraules les representaríem com es mostra a la següent figura.

Figura 3: Representació vectorial One-Hot Encoding de 3 paraules diferents.

A la figura 3, veiem que ara cada paraula té la seva pròpia dimensió i la distància entre elles és la mateixa, així doncs hem eliminat el malentés de relació de paraules, però tampoc és veritat que la relació entre cada paraula és la mateixa.

El que hem de fer doncs, és trobar la relació entre les diferents paraules d'un llenguatge, però altre cop, això és una feina molt complicada per un ésser humà. Per tant, farem servir una màquina per dur a terme a aquesta tasca.

Això vol dir que farem servir un model entrenat per relacionar paraules vectorialment, per després fer servir aquestes representacions per entrenar un altre model per entendre un llenguatge humà. Aquest primer model rep el nom del concepte *Embedding*.

4 Xarxes Neuronals Recurrents

Ara ja tenim clar com representar una paraula, però encara no ha acabat el problema. Està clar que depenent del model entrenat per classificar les paraules i del text d'entrada que li donem per entrenar-se, tindrà un criteri d'avaluació en concret. Això vol dir, que podrem representar qualsevol paraula.

La qüestió ara és que per entendre una oració no només val classificar cada paraula segons un corpus de dades gegant. També s'ha d'analitzar el context i la relació de les paraules en aquesta frase.

Per poder tractar una oració tenint en compte cada paraula d'aquesta, s'ha fet servir durant molt de temps les xarxes neuronals recurrents (*XNR*; o bé en anglès *RNN*, acrònim de *Recurrent Neural Network*).

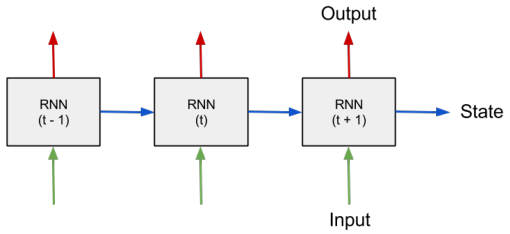

Figura 4: Esquema del comportament d'una RNN.

El comportament d'una RNN, com podem veure a la figura anterior, es basa a tractar les dades d'entrada de manera seqüencial i que cada element que s'analitza tingui en compte els anteriors. Així doncs, li donem un significat a l'estructura de la frase i donem context a oracions com:

La taronja és rodona; *La rodona és taronja*

Com podem observar, en aquestes dues frases es fan servir les mateixes quatre paraules, això vol dir que segons la representació que li havíem donat a cada paraula, haurien de ser iguals. Però, hi ha una evidència molt clara per nosaltres, els humans, i és el canvi de subjecte.

En fer servir una RNN, analitzaria que el verb *és* de la primera frase té relació amb *la taronja*, i el verb emphés de la segona té relació amb *la rodona*. Així doncs, ja hem resolt el problema del context i els models són capaços d'entendre'l. A excepció de les frases extenses...

5 Transformes

Sí, ja havíem arribat a una solució i sí, el tractament de dades seqüencial és la metodologia correcta per resoldre aquest tipus de problemes. Però resulta que com més extensa és l'oració que volem analitzar, menys relació tenen les paraules del principi amb les del final. Per exemple:

L'ocell recollia pals grans i petits per poder acabar de construir el seu niu.

En aquesta oració, el sintagma nominal *el seu niu* fa referència a *l'ocell*. Però la xarxa neuronal ha hagut de tractar tantes paraules pel mig que ha perdut el context. És per això que s'ha d'acabar de matisar el comportament d'una RNN.

El problema que hem de resoldre ara es denomina *falta de memòria* o *falta d'atenció*, i és que de fet, si no estem atents a l'inici de l'oració, no podrem saber la referència de les darreres paraules. Arran d'això, va sortir un nou *paper* que va capgira tota la branca de *machine learning*, Attention Is All You Need.

El que ens presenten en aquest *paper*, a grosso modo, és el Transformer, el primer model de transducció de seqüència basat completament en atenció. Els dos principals punts forts dels transformers són els mecanismes d'atenció, que permet relacionar cadascun dels elements a tractar, i l'ús de la posició relativa i absoluta de cada element de la seqüència.

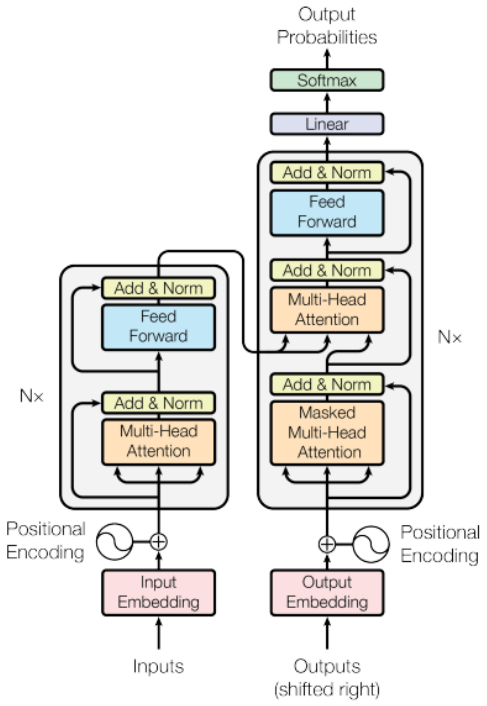

Figura 5: Arquitectura del Transformer.

També s'ha de tenir en compte l'excel·lent avenç de la capacitat d'entrenar aquests models amb temps significativament inferiors a les RNN. Això és degut al fet que ja no depenem del tractament seqüencial sinó que podem tractar els diferents elements paral·lelament.

Avui en dia, pràcticament tots els models que usaven xarxes neuronals recurrents han sigut substituïts per transformers, arribant a així a nou rècords de *machine learning* i aquests han pogut resoldre tasques que fins i tot no havien sigut entrenats per fer-ho.

Aquest és l'exemple de GPT-3, un model transformer entrenat per generar text de tal manera que sigui molt semblant a com ho fem els humans. Aquest rep com a entrada la introducció o els tòpics d'un concepte i és capaç d'interpretar què es demana i realitzar la tasca.

Un exemple seria l'entrada de dues línies d'un diàleg i GPT-3 és capaç de continuar amb el diàleg. Com més específica sigui l'entrada que li donem, més exhaustiva serà la sortida que doni. El que no s'esperava és que aquest model tingués la capacitat de generar codi de programació a partir d'una entrada que defineix la tasca a realitzar.

Altres aplicacions que poden tenir l'ús de la tecnologia dels transformers podrien ser el processament d'imatges i la identificació de patrons per poder-les classificar, el motor de cerca que usen plataformes com *Google*, l'assistència de veu o robots domèstics, el mecanisme d'etiquetar, corregir o traduir textos, entre altres.

6 Realització del Projecte

Arribat en aquest punt, ja sabem que l'ús del transformer és una de les millors metodologies per resoldre la correcció d'errors gramaticals. La llibreria que recomana el *paper* que hem vist anteriorment del transformer és Tensor2Tensor administrada Tensorflow.

El corpus de dades que usarem per entrenar el nostre model serà de la pàgina web Hugging Face. Aquesta les fan servir empreses tan importants com *Amazon*, *Facebook AI*, *Microsoft*, *Google AI* i moltes més.

En concret farem servir el dataset gnad10, el qual recopila deu mil articles dels diaris en alemany. Però, nosaltres farem servir l'entrenament supervisat, és a dir, que a partir de les diferents oracions que hi trobem en el dataset, generarem frases errònies per poder ensenyar a la intel·ligència artificial la resposta correcta.

Per veure la capacitat dels transformers, només ens centrarem en dos tipus d'errors gramaticals, el lèxic i l'ús dels signes de puntuació. Pel primer, agafarem totes les paraules del nostre corpus i a cadascuna li aplicarem una transformació de manera que sigui una paraula errònia. Les transformacions que aplicarem seran els següents:

- Delete: Aquest mètode elimina una lletra diferent a cada iteració, generant un llistat de n paraules, on n és el nombre de lletres. Exemple: *delete('pen')* $\rightarrow$ *['en', 'pn', 'pe']*.
- Swap: Aquesta funció intercanvia cada parella de lletres, generant un llistat de $n-1$ paraules, on n és el nombre de lletres. Exemple: *swap('pen')* $\rightarrow$ *['epn', 'pne']*.
- Replace: Aquest reemplaça cada lletra de la paraula per les diferents lletres del nostre alfabet, generant un llistat de $n * 26$ paraules, on n és el nombre de lletres. Exemple: *replace('pen')* $\rightarrow$ *['aen', 'ben', 'cen', 'den', ..., 'pex', 'pey', 'pez']*.
- Insert: Aquest darrer mètode introdueix les diferents lletres del nostre alfabet a cada posició de la paraula, inclòs davant i darrere, generant un llistat de $(n + 1) * 26$ paraules, on n és el nombre de lletres. Exemple: *replace('pen')* $\rightarrow$ *['apen', 'bpen', 'cpen', 'dpen', ..., 'penx', 'peny', 'penz']*.

Per raons d'escabilitat, els dos darreres mètodes no han sigut usats per l'aprenentatge.

6.1 Resultats Obtinguts

Finalment, s'ha dut a terme un joc de proves per analitzar el comportament i el rendiment del nostre transformer amb diferents paràmetres.

Abans de veure els resultats extrets, s'ha d'aclarir que la configuració interna de la xarxa neuronal és la mateixa per tots els casos, ja que és la recomanada per defecte, i en canviar certs paràmetres afectava de manera negativa al rendiment d'aprenentatge.

En canvi, els paràmetres que sí que varien a les diferents proves són el nombre d'oracions que el model usa per entrenar-se, el nombre de repeticions que realitza (epoch), la taxa d'aprenentatge inicial (learning rate) i el nombre de tokens que s'analitza en un lot (batch size).

Les proves han sigut dividides en tres tipus:

- Correcció del lèxic de paraules aïllades, és a dir, l'entrada és una paraula sola i la sortida la paraula corregida en cas que fos errònia.
- Correcció del lèxic de paraules dins d'oracions. Ara l'entrada serà una oració on totes les paraules estaran ben escrites o hi haurà una mal escrita i la sortida és l'oració completa ben escrita.
- Correcció de signes de puntuació. Aquí tornem a tenir una oració d'entrada, on cap paraula ha sigut modificada, però algun punt i coma ha sigut eliminat. La sortida ha de ser la frase amb els signes de puntuació corresponents.

6.1.1 Correcció Lèxic de Paraules Aïllades

D'aquest apart s'han fet diverses proves canviant els paràmetres, però només s'ha capturat una sola prova, ja que els resultats eren força similars.

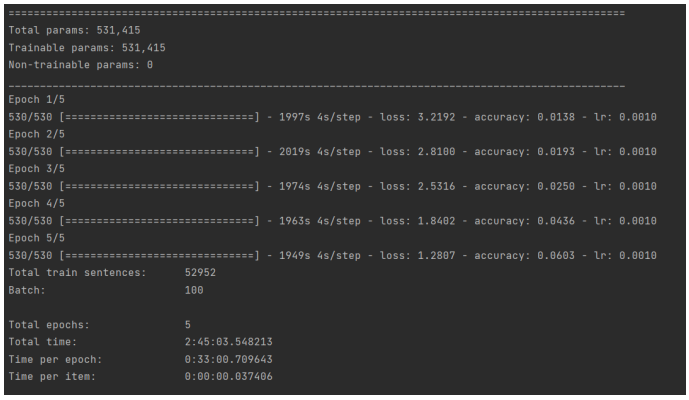

Figura 6: Prova de Correcció Lèxic de Paraules Aïllades.

Com podem veure, el resultat és força incompetent, possiblement causat perquè el text a analitzar no és una seqüència de tokens, sinó un sol token.

6.1.2 Correcció Lèxic de Paraules dins d'Oracions

D'aquest segon apartat també hem fet forces proves amb resultats més interessants. El nombre de epoch és el mateix per tots, ja que aquest no afectava consideradament al resultat. Els valors que donaven canvis significatius són el el nombre de frases, el batch size i el learning rate.

Primera prova, nombre d'oracions a tractar = 100, batch size = 100 i learning rate = 0,001.

```
=================================================================================================
Total params: 531,415
Trainable params: 531,415
Non-trainable params: 0
_________________________________________________________________________________________________
Epoch 1/5
206/206 [==============================] - 774s 4s/step - loss: 3.3568 - accuracy: 0.1194 - lr: 0.0010
Epoch 2/5
206/206 [==============================] - 748s 4s/step - loss: 3.2369 - accuracy: 0.1224 - lr: 0.0010
Epoch 3/5
206/206 [==============================] - 738s 4s/step - loss: 2.5303 - accuracy: 0.2735 - lr: 0.0010
Epoch 4/5
206/206 [==============================] - 738s 4s/step - loss: 1.6773 - accuracy: 0.4560 - lr: 0.0010
Epoch 5/5
206/206 [==============================] - 742s 4s/step - loss: 1.1127 - accuracy: 0.6010 - lr: 0.0010
Total train sentences:      20553
Batch:                      100

Total epochs:               5
Total time:                 1:02:20.573189
Time per epoch:             0:12:28.114638
Time per item:              0:00:00.036399
```

Figura 7: Prova de Correcció Lèxic de Paraules dins d'Oracions.

Segona prova, nombre d'oracions a tractar = 100, batch size = 1 i learning rate = 0,001.

```
=================================================================================================
Total params: 531,415
Trainable params: 531,415
Non-trainable params: 0
_________________________________________________________________________________________________
Epoch 1/5
20553/20553 [==============================] - 1043s 50ms/step - loss: 0.4549 - accuracy: 0.8358 - lr: 0.0010
Epoch 2/5
20553/20553 [==============================] - 1050s 51ms/step - loss: 0.9780 - accuracy: 0.6532 - lr: 0.0010
Epoch 3/5
20553/20553 [==============================] - 1046s 51ms/step - loss: 0.3893 - accuracy: 0.8130 - lr: 0.0010
Epoch 4/5
20553/20553 [==============================] - 1041s 51ms/step - loss: 0.2991 - accuracy: 0.8393 - lr: 0.0010
Epoch 5/5
20553/20553 [==============================] - 1041s 51ms/step - loss: 0.2598 - accuracy: 0.8511 - lr: 0.0010
Total train sentences:      20553
Batch:                      1

Total epochs:               5
Total time:                 1:27:00.642459
Time per epoch:             0:17:24.128492
Time per item:              0:00:00.050802
```

Figura 8: Prova de Correcció Lèxic de Paraules dins d'Oracions.

Tercera prova, nombre d'oracions a tractar = 200, batch size = 1 i learning rate = 0,001.

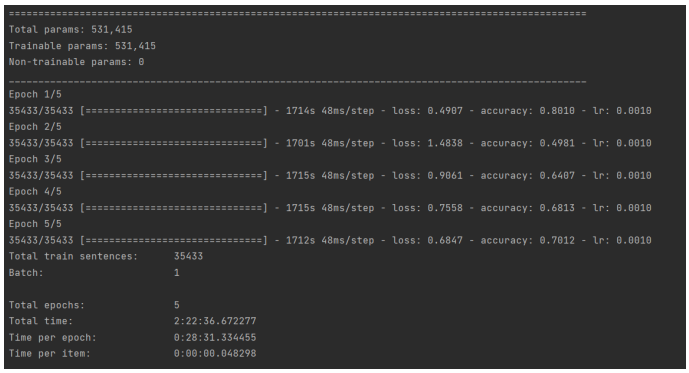

Figura 9: Prova de Correcció Lèxic de Paraules dins d'Oracions.

Quarta prova, nombre d'oracions a tractar = 100, batch size = 1 i learning rate = 0,01.

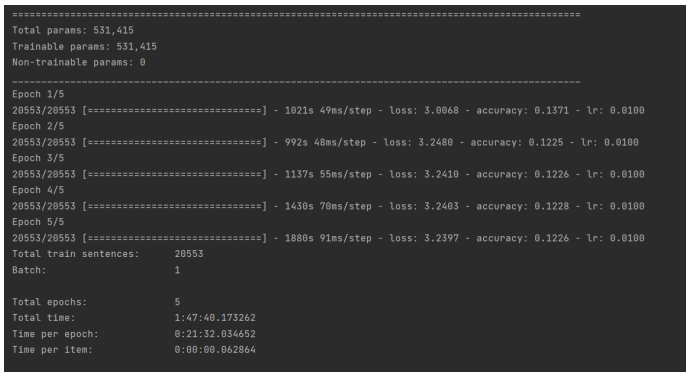

Figura 10: Prova de Correcció Lèxic de Paraules dins d'Oracions.

Les conclusions que podem extreure són que els paràmetres batch size i learning rate afecten notablement al resultat, així com ho fa el nombre d'oracions per entrenar, però sense causar tants danys al resultat.

6.1.3 Correcció de Signes de Puntuació

Finalment, tenim la correcció de signes de puntuació que també ha fallat l'aprenentatge. Tot i ajustar els paràmetres, els resultats eren similars al que es mostra a continuació.

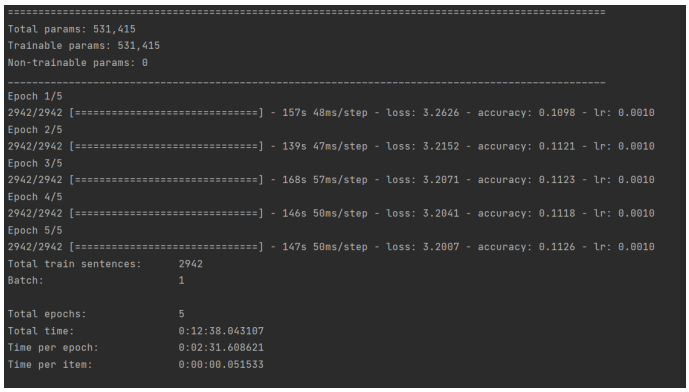

Figura 11: Prova de Correcció de Signes de Puntuació.

En aquest cas es fa servir una mostre de 1.000 oracions, batch size = 1 i learning rate = 0,001. Segurament s'ha comés un error de configuració del transformer o ha sigut entrenat ignorant els punts i les comes.