Analiza de sentiment pe IMDB cu Bidirectional LSTM

1. Descrierea proiectului

Scopul acestui proiect este sa construim un model de clasificare binara care sa prezica daca o recenzie din setul iMDB https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz este pozitiva sau negativa. Am folosit o retea neuronala bidirectionala de tip LSTM pentru a captura dependentele secventiale din text si am evaluat performanta pe un set de test.

Codul sursa: https://github.com/U-1-Decembrie-AB/Procesare-text

2. Tehnologii folosite

- **Limbaj:** Python 3.8
- **TensorFlow / Keras**
  - **Incarcarea dataset-ului** iMDB direct prin imdb.load_data().
  - **Preprocesare** interna pentru tokenizare si maparea cuvintelor la indici.
  - **Definirea arhitecturii**:
    - Embedding — transforma indicii de cuvinte in vectori densi (128 dimensiuni).
    - Bidirectional(LSTM) — capteaza dependente secventiale in ambele directii ale textului.
    - Dropout si Dense cu activare sigmoid — realizeaza regularizare si proiectie catre clasificare binara.
  - **Antrenare**: .fit() cu callback ModelCheckpoint pentru salvarea automata a celor mai bune greutati pe baza pierderii de validare.
  - **Evaluare**: .evaluate() si .predict() direct pe setul de test.
- **NumPy**

- **Structurarea datelor**: transforma listele de indici (din iMDB) in matricee de tip ndarray.

- **Operatii aritmetice** rapide si broadcasting pentru manipularea batch-urilor de date.

- Serveste drept **backend numeric** pentru tensorFlow (o buna parte din calcule se bazeaza pe accelerarea oferita de NumPy).

- **sciPy**
  - Desi nu este invocat direct in cod, sciPy este un set de **functii numerice** si optimizari (folosit “sub capota” de multe ori de tensorFlow/Keras si scikit-learn pentru operatii liniare, factorizari, interpolari etc.).

  - Iin proiecte viitoare poate fi utilizat pentru **preprocesari avansate** (de ex. calcul de statistici, transformari FFt, filtre de semnal etc.).

- **scikit-learn**
  - **Impartirea** setului de antrenare in antrenare + validare cu train_test_split().

  - **Calcul metrici**:

    - classification_report() — iti ofera precision, recall si F1-score pe fiecare clasa.

    - confusion_matrix() — matricea de confuzie pentru analiza distributiei erorilor.

- **Matplotlib**
  - Generarea **grafurilor de evolutie** a pierderii (loss) si acuratetii (accuracy) vs epoca.

  - Configurarea dimensiunii figurii, axelor, titlurilor si legendelor pentru rapoarte clare.

- **seaborn**
  - **Heatmap** pentru matricea de confuzie, cu:

    - paleta de culori („Blues”) care evidentiaza corect si gresit clasificate.

    - adnotari numerice („annot=true”) pentru citire rapida a valorilor.

- simplifica stilizarea si ofera un look mai curat decat metoda „raw” Matplotlib.

3. Instructiuni de rulare

In VS Code:

- **Deschizi folderul proiectului**
  „File → Open Folder…” si selectezi C:\Procesare-text. VS Code trebuie sa vada fisierele main.py

- **Creezi si activezi un mediu virtual în terminalul VS Code**
  - Deschide terminalul integrat

- **Ruleaza**:
  - python -m venv .venv
  - .\.venv\Scripts\Activate.ps1

- **Instalezi TensorFlow in acel mediu**
  - pip install --upgrade pip
  - pip install tensorflow numpy matplotlib scikit-learn seaborn

4. Date de intrare

- **Setul de date:** iMDB Movie Reviews (25.000 recenzii pentru antrenare, 25.000 pentru test), preincarcat din Keras cu num_words=20000.

- **Split-uri:**

  - Antrenare: 20.000
  - Validare: 5.000
  - Test: 25.000

5. Preprocesarea datelor

- Trunchiere la lungimea fixa **MAXLEN = 200**
- Padding la sfarsitul secventei (padding='post')
- Rezulta o matrice de dimensiuni (num_samples, 200)

6. Arhitectura modelului

```
model = sequential([
   Embedding(input_dim=20000, output_dim=128, input_length=200),
   Bidirectional(LSTM(64, return_sequences=False)),
   Dropout(0.5),
   Dense(1, activation='sigmoid')
])
```

- **Embedding:** cu dimensiunea de 128
- **Bidirectional LSTM:** 64 de unitati
- **Dropout:** 50%
- **Head final:** un neuron cu activare sigmoid

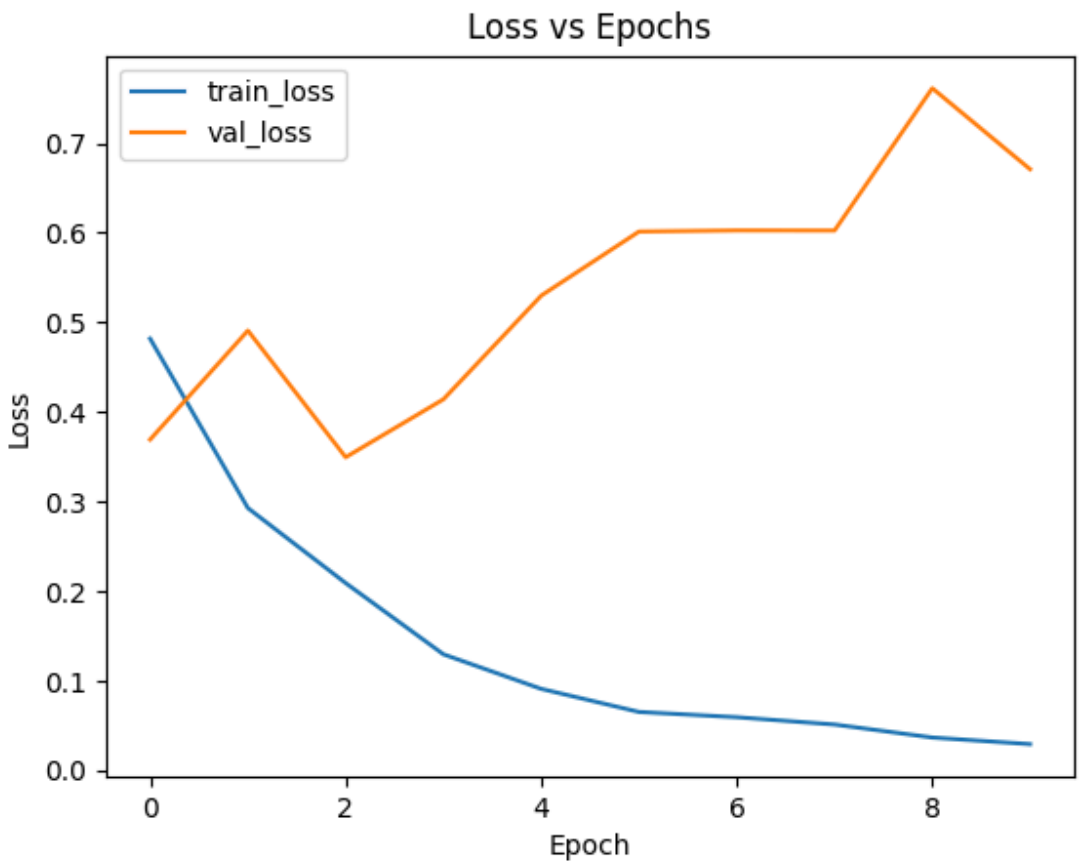
7. Antrenarea modelului

- Functia de pierdere: **binary_crossentropy**
- Optimizer: **adam** (learning_rate=1e-3)
- Epoci: **10**
- Batch size: **64**
- Callback: ModelCheckpoint salveaza greutatile cu cea mai mica valoare a pierderii pe validare

8. Rezultate

8.1 Evolutia pierderii

Figura 1. train_loss si val_loss vs epoca.



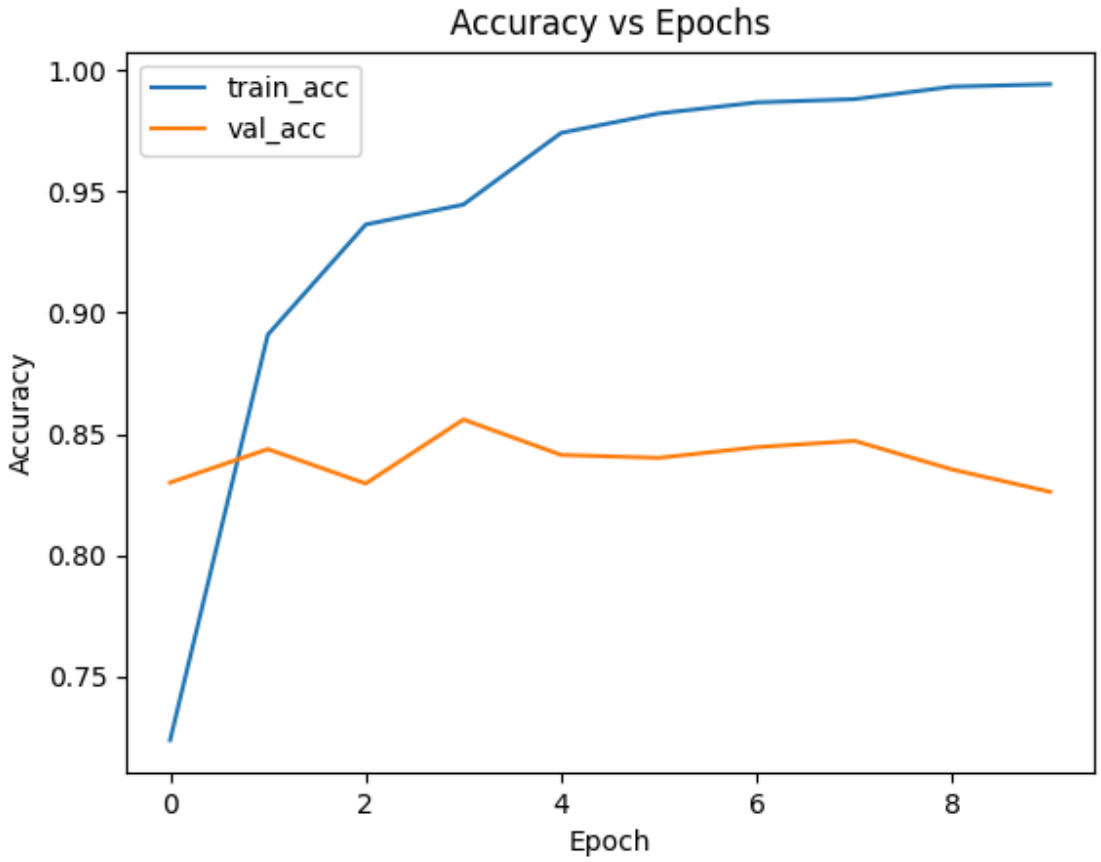
Train loss porneste de la ≈0.48 la epoca 0 si scade foarte rapid, ajungand sub 0.1 deja la epoca 4 si la ≈0.03 la epoca 9.

Val loss scade initial de la ≈0.37 la epoca 0 pana la ≈0.35 la epoca 2, apoi incepe sa creasca constant, ajungand la ≈0.75 in jurul epocii 8.

Interpretare: dupa epoca 2–3 modelul incepe sa invete prea specific pe datele de antrenare (overfitting), motiv pentru care pierderea de validare creste chiar daca pierderea de antrenament continua sa scada.

8.2 Evolutia acuratetii

Figura 2. train_acc si val_acc vs epoca.



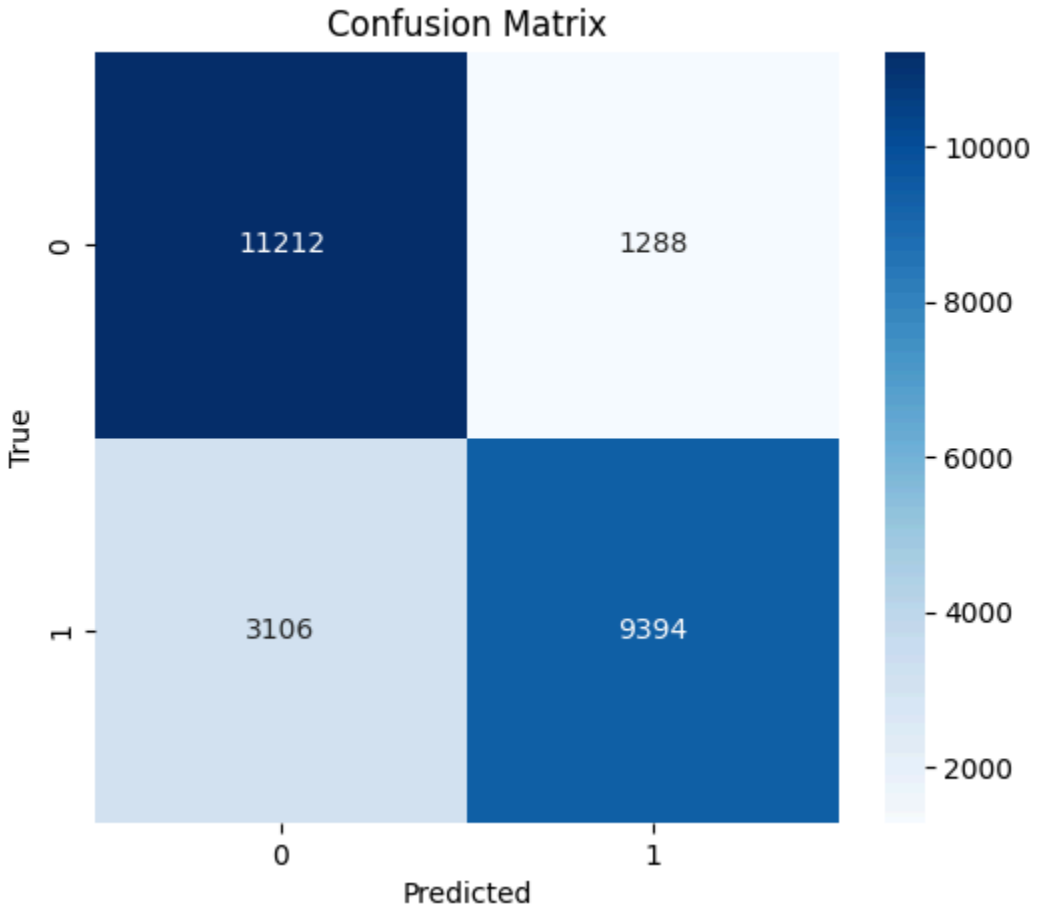
Train accuracy urca de la ≈0.72 la epoca 0 pana la ≈0.995 la epoca 9.

Val accuracy atinge un maximum de ≈0.85 in jurul epocii 4, apoi se stabilizeaza in jurul valorii de 0.84–0.845 si chiar scade usor pana la ≈0.825 la epoca 9.

Interpretare: trendul confirma overfitting-ul–acuratetea pe antrenament continua sa creasca, pe validare stagneaza si scade usor dupa cateva epoci.

8.3 Matricea de confuzie pe test

Figura 3. Matrice de confuzie:



- true Negative (0↦0): 11 212
- False Positive (0↦1): 1 288
- False Negative (1↦0): 3 106
- true Positive (1↦1): 9 394

Interpretare: modelul clasifica corect majoritatea recenziilor negative (tN), insa are un numar mai mare de fals negative (FN = 3 106) decat de fals positive (FP = 1 288), ceea ce indica sub-detectarea sentimentului pozitiv in unele cazuri.

8.4 Metrici de clasificare

Incluzand precision, recall si F1-score obtinute pe setul de test:

```
Classification Report:
              precision    recall  f1-score   support

           0     0.7729    0.9066    0.8344     12500
           1     0.8870    0.7337    0.8031     12500

    accuracy                         0.8201     25000
   macro avg     0.8300    0.8201    0.8188     25000
weighted avg     0.8300    0.8201    0.8188     25000
```

- **Clasa 0 (negativ)**:
    - *Precision* 0.773 → ~23% din exemplele prezise negative erau de fapt pozitive
    - *Recall* 0.907 → 90.7% din recenziile negative au fost detectate
    - *F1* 0.834
- **Clasa 1 (pozitiv)**:
    - *Precision* 0.887 → majoritatea predictiilor de pozitiv erau corecte
    - *Recall* 0.734 → doar 73.4% din recenziile pozitive au fost identificate
    - *F1* 0.803
- **Accuracy globala** ≈ 0.83

Interpretare finala: modelul are o performanta buna pe ambele clase, cu un usor dezechilibru: detecteaza mai bine recenziile negative (recall 0.907) decat pe cele pozitive (recall 0.734), desi atunci cand prezice „pozitiv” o face cu o precizie mai mare (0.887). acest fapt reiese si din distributia erorilor din matricea de confuzie.

9. Posibile imbunatatiri

- **Experimentare hiperparametri:** rate de invatare, batch size, numar epoci
- **initializari ponderi:** Xavier, He
- **Functii de activare alternative:** ReLU, tanh inainte de head-ul sigmoid
- **Tokenizare si preprocesare avansata:** lowercasing, stemmare/lemmatizare, indepartare stop-words
- **Aalternative de model:** GRU, transformer (BERt, LSTM cu atentie)
- **Augmentarea datelor text** (back-translation, synonym replacement)

10. Concluzii

Modelul Bidirectional LSTM a atins o acuratete de ~0.83 pe test, demonstrand potentialul RNN-urilor pentru sarcini de analiza de sentiment. Rezultatele pot fi imbunatatite prin experimente suplimentare si preprocesare avansata.