# Proiect Inteligență Artificială

Text-Classification

*Burdescu Alexandru, Facultatea de Matematică și Informatică*

## Problema propusă

Problema propusă poate fi găsită alături de toate detaliile la linkul competiției:
https://www.kaggle.com/c/ml-2020-unibuc-3/leaderboard

## Metode de abordare

Pentru această problema, am ales 2 metode de abordare. Una ce este reprezentată de o abordare cu un clasificator classic (ComplementNB) dar și una ce este reprezentată de o rețea neuronală.

## Prelucrarea datelor

Pentru rețeaua neuronală am preferat Tokenizerul oferit de Keras din preprocessing.text. Acesta este antrenat pe datele de training. Acesta creează un dicționar în care ficarui cuvânt îi atrbuie un număr în funcție de cât de des apare în textele noastre. De exemplu, cel frecvent cuvânt va avea atribuit numărul 1, al doilea cel mai frecvent numărul 2 și așa mai departe.
Apoi textul este înlocuit cu numerele aferente cuvintelor. Pentru cuvintele care nu există în vocabular am folosit tokenul OOV ( out of vocabulary ) ce are atribuit numărul 0.

De asemenea, funcția get_first_stop_wrods, da valoarea 0 primelor X ( parametru reglabil ) celor mai commune cuvinte din dicționarul nostru, neconsiderandu-le importante pentru clasificarea noastră. Apoi urmează să filtrăm textele noastre de 0-uri
( lucruri considerate inutile pentru problema noastră)

Alegem sizeul inputului pentru rețeaua noastră prin funcția get_info_about_data. Această returnează media + 2*deviația standard că numărul cuvintelor ce ar trebui considerat din fiecare text ( pentru a nu alocat memorie inutil, având în vedere că dimensiunea inputului trebuie să fie una fixă ) dar și câte cuvinte diferite sunt, procentul de acoperire al dimensiunii alese. Textele ce sunt mai mari vor fi trunchiate, iar cele ce sunt mai mici vor primii padding cu 0-uri la început.

În cazul clasificatorului, am optat pentru TfidfVectorizer cu un analyzer custom ce împarte textul în cuvinte, după spații si ngrams_range= (1,3).

# Proiect Inteligență Artificială

Text-Classification

*Burdescu Alexandru, Facultatea de Matematică și Informatică*

De ce nu am folosit Tfidf și în cazul rețelei având în vedere că este o abordare mult mai ușoară ? Trecând peste problemele de compatibilitate, ce am reușit să le rezolv ( tfidf întoarce matrice de tip sparse ce nu erau acceptate că input ) , am rămas cu un input de tip 2D. Am încercat să bag inputul într-un embedding layer pentru a obține datele de tip 3D, doar că nu am avut suficientă memorie pentru a continuă mai departe cu convolutii/layere recurente etc.

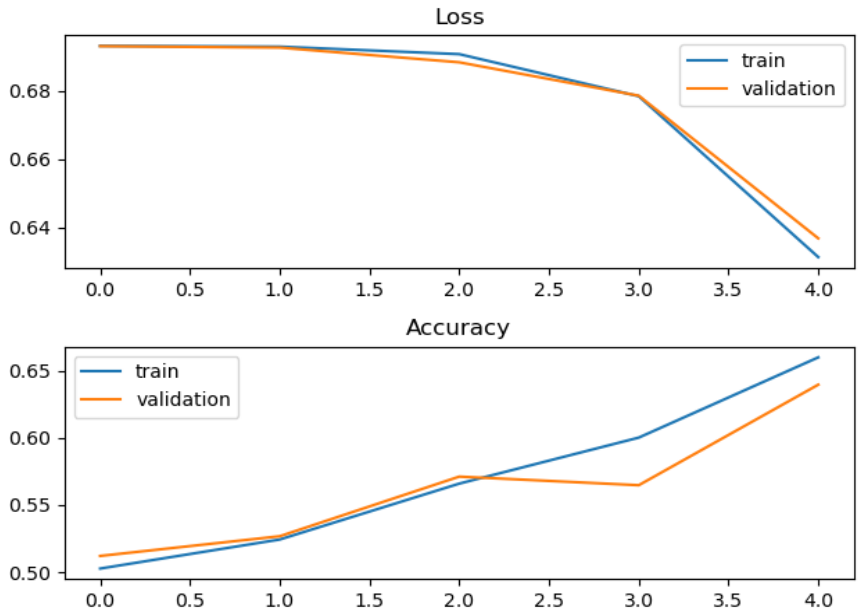
## Metode abordate

1. Rețea Neuronală

a.) Rețeaua:

```
model = Sequential([
    Embedding(input_dim=lungime_vocabular, output_dim=8, input_length=lungime_input),
    Bidirectional(LSTM(64, return_sequences=True, recurrent_dropout=0.25)),
    Bidirectional(LSTM(16, return_sequences=False, recurrent_dropout=0.25)),
    Dense(128),
    Activation('relu'),
    Dense(32),
    Activation('relu'),
    Dense(1),
    Activation('sigmoid')
])
optimizer = Adam(lr=0.0001)
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

b.) Un plot pentru 5 epoci de training:



# Proiect Inteligență Artificială

Text-Classification

*Burdescu Alexandru, Facultatea de Matematică și Informatică*

c.) F1 Score și Matricea de confuzie:

F1 Score = 0.6893865628042843

Confusion Matrix

| | Predicted 0 | Predicted 1 |
|---|---|---|
| Actual 0 | 637 | 293 |
| Actual 1 | 664 | 1062 |

d.) Alte încercări:

Am încercat după Embedding layer, un layer convolutional Conv1D(filters =64, kernel_size = 5, activation = 'relu', stride=1) urmat de un MaxPooling( pool_size=4). De altfel am încercat diferiți parametrii pentru learning rate cât și pentru fiecare layer în parte. Am încercat diferite celule recurente ( GRU / LSTM ) cât și layerul să îl las normal dar și Bidirecțional. Am încercat cu foarte mulți parametrii pentru output_dim-ul din Embedding, și aici s-au putut vedea fluctuații semnificative datorită transpunerii în mai multe/mai puține dimensiuni.

e.) Probleme:

Probleme întâmpinate: Problemele întâmpinate au fost de overfitting. Accuracyul rețelei se plafoneaza undeva pe la 67-68 în timp ce lossul începe să urce drastic. Cum am încercat să rezolv aceste probleme ? Am micșorat learning rateul , am încercat adăugarea layerelor de Dropout, am încercat BatchNormalization, însă niciuna din solutile de mai sus nu m-au ajutat foarte mult, în afară de amânarea fenomenului și câștigul a 2-3 procente.

f.) Alegeri sigure:

Pentru layerul de output am ales activarea sigmoid pentru că această da valori în intervalul (0,1). Orice prediction peste 0.5 l-am luat că fiind 1, orice prediction sub 0.5 l-am luat că fiind 0.

Funcția de loss aleasă este binary_crossentropy, funcție folosită când avem de diferențiat 2 categorii de obiecte.

Optimizatorul ales a fost Adam, deși am încercat și RMSprop, Adam a părut a avea cele mai bune rezultate.

Batch_sizeul pentru fitting l-am setat la 128, tot în sepranta că făcând back-propagation mai rar o să reduc overfittingul

# Proiect Inteligență Artificială

Text-Classification

*Burdescu Alexandru, Facultatea de Matematică și Informatică*

### 2. ComplementNB

a.) Motivație:

De ce ComplementNB? Am încercat diferiți calsificatori: svm-uri, NB-uri, decision trees. ComplementNB s-a dovedit a fi cel mai optim dintre acestea.

b.) Parametri:

Față de o rețea, aici nu este extrem de greu să găsești parametrii optimi. Am folosit Grid Search și pur și simplu l-am ales pe cel mai bun dintre toate. Parametrul optim pentru acesta este alpha= 0.12

c.) F1 Score și Matricea de confuzie:

F1 Score = 0.7296137339055795

Confusion Matrix

| | Predicted 0 | Predicted 1 |
|---|---|---|
| Actual 0 | 880 | 335 |
| Actual 1 | 421 | 1020 |

## Concluzii

Un lucru cert după acest proiect este că m-am acomodat cu Keras și cu rețelele în general, dar și cu diferiții calsificatori oferiți de sklearn. Am învățat că lucrul cu o rețea neuronală nu este ușor și în același timp, am învățat că niciodată, dar niciodată să nu subestimez un clasificator.