MAC5725 – Linguística Computacional – 2020

EP 2: RNNs Bidirecionais, Overfitting, Underfitting

Augusto Cesar de Camargo NETO

NUSP: 11891023

Descrição das configurações e hiper-parâmetros **2**
- Pré-processamento 2
- Redes utilizadas 3
- Hiper-parâmetros estudados 3
- Grid Search 4

Gráficos Gerados **7**
- LSTM 7
- Biderecional 9

Tabela de Acurácias **10**
- LSTM 10
- Bidirecional 10

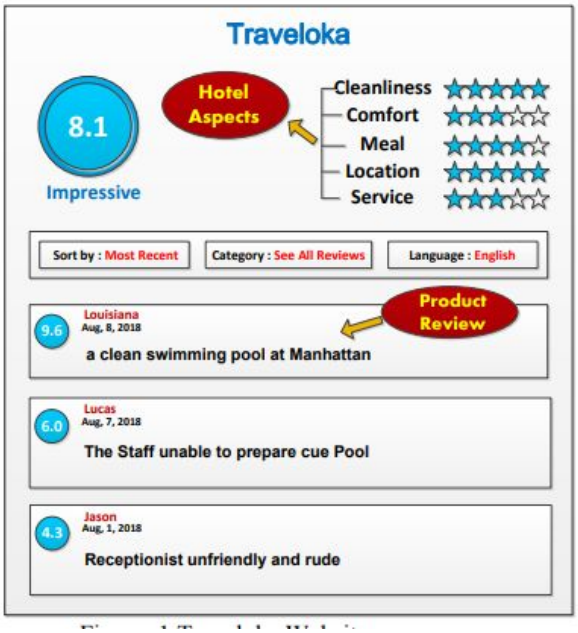
Discussão de Resultados **11**

Referências **12**

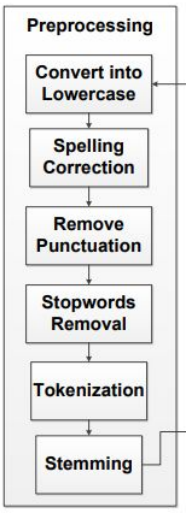
Descrição das configurações e hiper-parâmetros

Pré-processamento

Em (AYU KHUSNUL KHOTIMAH; SARNO, 2019) podemos ver uma classificação semelhante à que utilizamos aqui (1-5):



Neste trabalho utilizamos o mesmo pré-processamento de Ayu Khusnul Khotimah e Sarno (2019) menos a correção e estemização do texto:



Redes utilizadas

Foram utilizadas duas redes:

- keras.layers.LSTM
- keras.layers.Bidirectional(forward_layer, backward_layer=backward_layer)

Hiper-parâmetros estudados

Os hiper-parâmetros utilizados no grid search foram:

- Units
- Dropouts
- Batch size
- Epochs

As configurações da utilizadas foram:

```
def myNet(
    SEQUENCE_MAXLEN,
    emb,
    nome,
    tipo,
    units,
    dropout,
    batch_size,
    epochs,
    x_train,
    y_train,
    x_val,
    y_val,
    x_test,
    y_test,
    ):

    if os.path.exists('weights.hdf5'):
        os.remove('weights.hdf5')

    model = keras.Sequential()
    model.add(layers.Input(shape=(SEQUENCE_MAXLEN, )))
    model.add(emb)
```

```
    if tipo == 'LSTM':
        model.add(keras.layers.LSTM(units, dropout=dropout))
        opt = 'adam'
    else:
        forward_layer = keras.layers.LSTM(units, activation='relu',
                dropout=dropout)
        backward_layer = keras.layers.LSTM(units, activation='relu',
                go_backwards=True, dropout=dropout)
        model.add(keras.layers.Bidirectional(forward_layer,
                  backward_layer=backward_layer))
        model.add(keras.layers.Dropout(dropout))
        opt = 'adam'
    model.add(keras.layers.Dense(5, activation='softmax'))
    model.compile(optimizer=opt, loss=sparse_categorical_crossentropy,
                  metrics=['accuracy'])
    checkpointer = tf.keras.callbacks.ModelCheckpoint(filepath='weights'
             + '_ ' + nome + '_Units_' + str(units) + '_Dropouts_'
            + str(dropout) + '_Batchs_' + str(batch_size) + '.hdf5',
            verbose=1, save_best_only=True)
    es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min'
            , verbose=1, patience=5)
    csv_logger = tf.keras.callbacks.CSVLogger('History_Log' + '_'
            + nome + '_Units_' + str(units) + '_Dropouts_'
            + str(dropout) + '_Batchs_' + str(batch_size) + '.csv',
            append=True, separator=',')
    history = model.fit(
        x=x_train,
        y=y_train,
        batch_size=batch_size,
        epochs=epochs,
        validation_data=(x_val, y_val),
        callbacks=[checkpointer, es, csv_logger],
        )
```

Grid Search

Inicialmente foi realizado uma série de experimentos iniciais (vide arquivo experimentos iniciais.pdf) variando alguns hiper-parâmetros onde não se chegou a resultados de alta performance (acurácia de mais de 85%). Na dúvida se era possível chegar a resultados

excelentes de acurácia foi realizado um grid search com as seguintes redes e hiper-parâmetros:

experimento	Rede	units	dropouts	batches	acc	acc'
1	LSTM	32	0	16	60.31%	60.31%
2	LSTM	32	0	32	60.07%	60.07%
3	LSTM	32	0	64	59.88%	59.88%
4	LSTM	32	0.25	16	60.54%	60.54%
5	LSTM	32	0.25	32	60.69%	60.69%
6	LSTM	32	0.25	64	59.93%	59.93%
7	LSTM	32	0.5	16	60.14%	60.14%
8	LSTM	32	0.5	32	59.00%	59.00%
9	LSTM	32	0.5	64	59.13%	59.13%
10	LSTM	64	0	16	60.20%	60.20%
11	LSTM	64	0	32	59.91%	59.91%
12	LSTM	64	0	64	59.59%	59.59%
13	LSTM	64	0.25	16	60.97%	60.97%
14	LSTM	64	0.25	32	60.85%	60.85%
15	LSTM	64	0.25	64	60.56%	60.56%
16	LSTM	64	0.5	16	60.67%	60.67%
17	LSTM	64	0.5	32	59.05%	59.05%
18	LSTM	64	0.5	64	60.66%	60.66%
19	LSTM	128	0	16	60.02%	60.02%
20	LSTM	128	0	32	60.16%	60.16%
21	LSTM	128	0	64	59.84%	59.84%
22	LSTM	128	0.25	16	60.76%	60.76%
23	LSTM	128	0.25	32	60.85%	60.85%
24	LSTM	128	0.25	64	60.16%	60.16%
25	LSTM	128	0.5	16	61.19%	61.19%
26	LSTM	128	0.5	32	61.03%	61.03%
27	LSTM	128	0.5	64	59.98%	59.98%
28	LSTM	256	0	16	59.69%	59.69%
29	LSTM	256	0	32	59.62%	59.62%
30	LSTM	256	0	64	59.67%	59.67%
31	LSTM	256	0.25	16	60.38%	60.38%

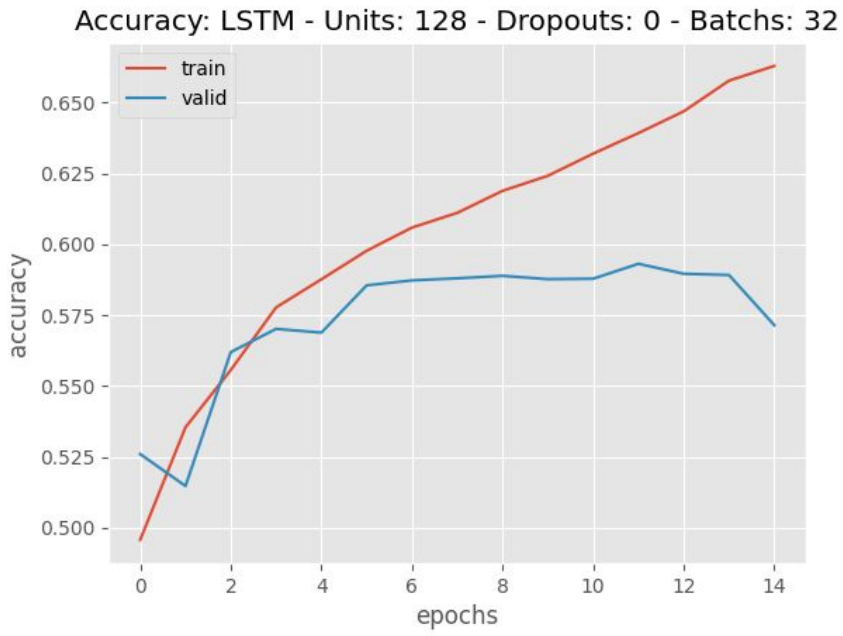
32	LSTM	256	0.25	32	60.70%	60.70%
33	LSTM	256	0.25	64	60.22%	60.22%
34	LSTM	256	0.5	16	60.61%	60.61%
35	LSTM	256	0.5	32	60.53%	60.53%
36	LSTM	256	0.5	64	60.37%	60.37%
37	Bideracional	32	0	16	60.39%	60.39%
38	Bideracional	32	0	32	60.49%	60.49%
39	Bideracional	32	0	64	60.81%	60.81%
40	Bideracional	32	0.25	16	60.46%	60.46%
41	Bideracional	32	0.25	32	59.95%	59.95%
42	Bideracional	32	0.25	64	60.50%	60.50%
43	Bideracional	32	0.5	16	58.46%	58.46%
44	Bideracional	32	0.5	32	59.32%	59.32%
45	Bideracional	32	0.5	64	59.46%	59.46%
46	Bideracional	64	0	16	59.45%	59.45%
47	Bideracional	64	0	32	60.21%	60.21%
48	Bideracional	64	0	64	60.59%	60.59%
49	Bideracional	64	0.25	16	61.09%	61.09%
50	Bideracional	64	0.25	32	60.89%	60.89%
51	Bideracional	64	0.25	64	60.71%	60.71%
52	Bideracional	64	0.5	16	59.98%	59.98%
53	Bideracional	64	0.5	32	59.67%	59.67%
54	Bideracional	64	0.5	64	59.53%	59.53%
55	Bideracional	128	0	16	59.83%	59.83%
56	Bideracional	128	0	32	60.42%	60.42%
57	Bideracional	128	0	64	46.90%	
58	Bideracional	128	0.25	16	60.81%	60.81%
59	Bideracional	128	0.25	32	61.52%	61.52%
60	Bideracional	128	0.25	64	61.38%	61.38%
61	Bideracional	128	0.5	16	60.88%	60.88%
62	Bideracional	128	0.5	32	56.96%	
63	Bideracional	128	0.5	64	58.78%	58.78%
64	Bideracional	256	0	16	59.31%	59.31%
65	Bideracional	256	0	32	60.18%	60.18%
66	Bideracional	256	0	64	58.68%	58.68%

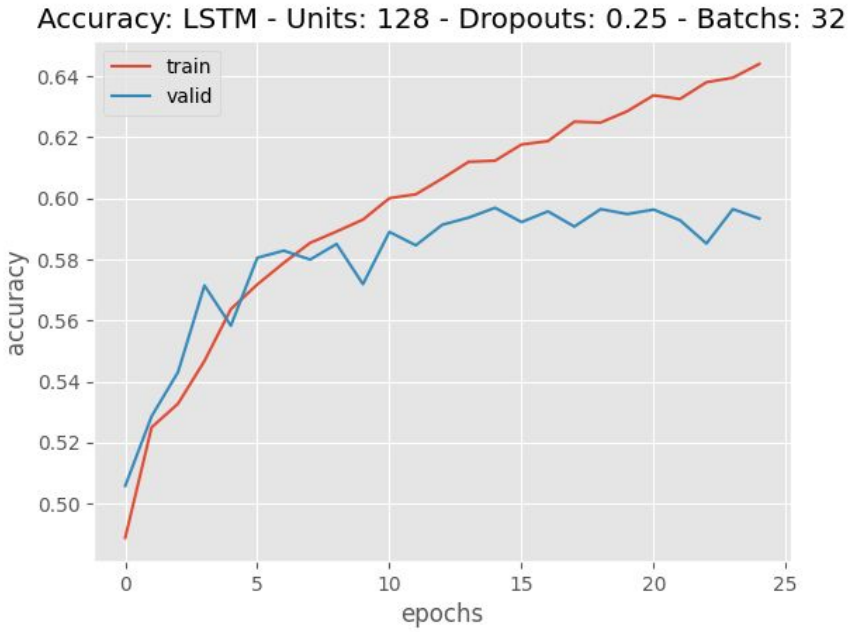
67	Bideracional	256	0.25	16	59.77%	59.77%
68	Bideracional	256	0.25	32	60.95%	60.95%
69	Bideracional	256	0.25	64	52.65%	
70	Bideracional	256	0.5	16	60.74%	60.74%
71	Bideracional	256	0.5	32	55.81%	
72	Bideracional	256	0.5	64	60.84%	60.84%
				Média	59.81%	60.21%
				Desvio Padrão	0.0198	0.0066
				%CV	3.31%	1.09%

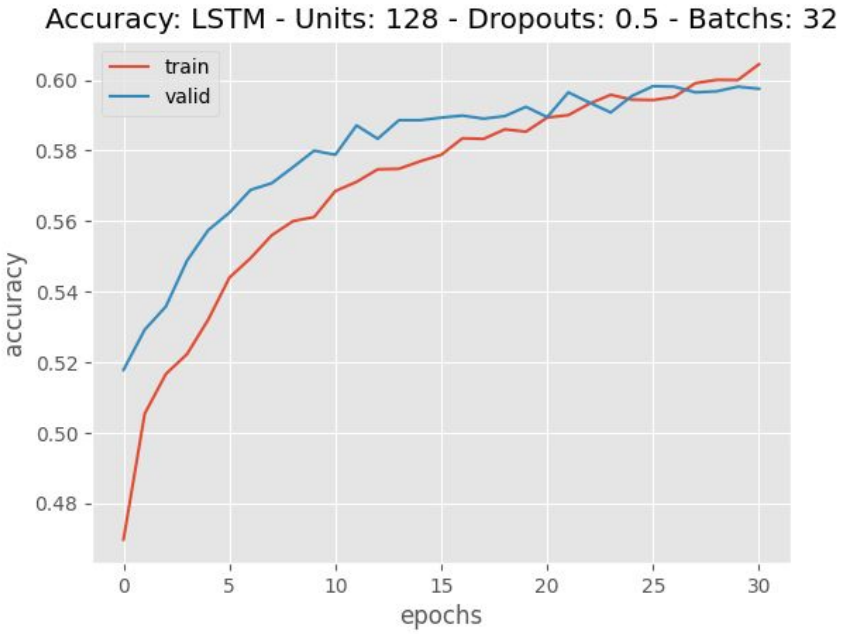
Gráficos Gerados

Os melhores resultados obtidos no grid search seguem abaixo.

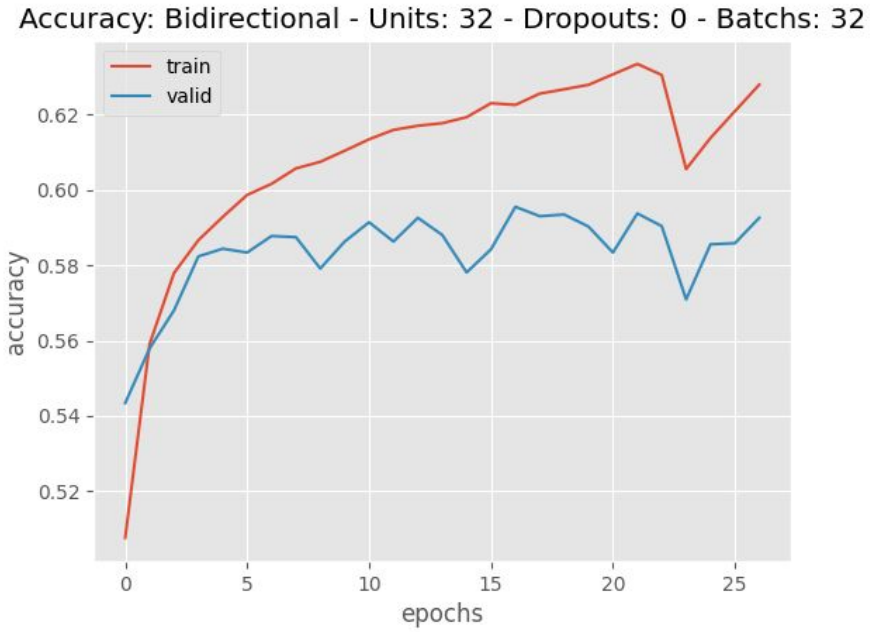
LSTM

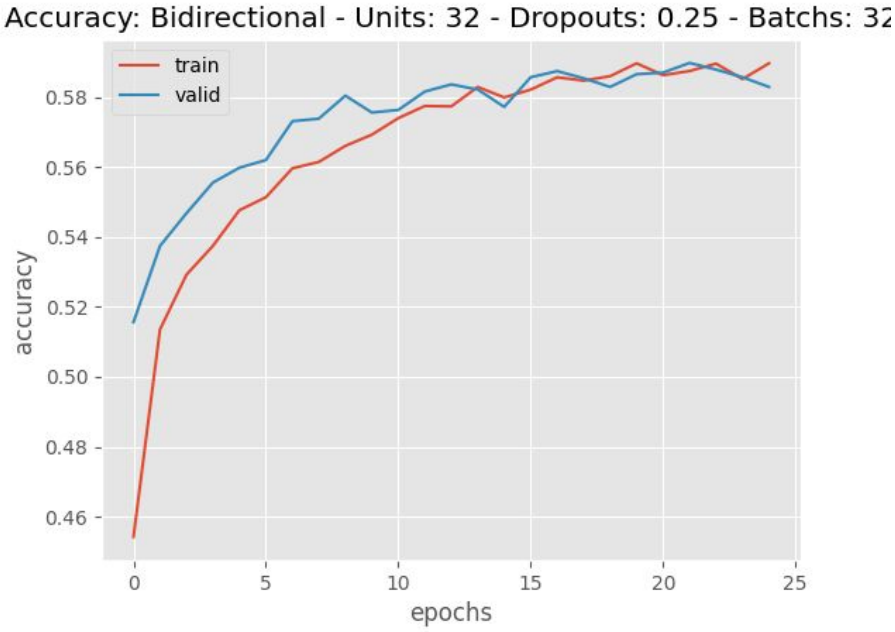



Accuracy: LSTM - Units: 128 - Dropouts: 0.25 - Batchs: 32
train
valid
accuracy
epochs
0.64
0.62
0.60
0.58
0.56
0.54
0.52
0.50
0
5
10
15
20
25


Accuracy: LSTM - Units: 128 - Dropouts: 0.5 - Batchs: 32
train
valid
accuracy
epochs
0.60
0.58
0.56
0.54
0.52
0.50
0.48
0
5
10
15
20
25
30

Biderecional





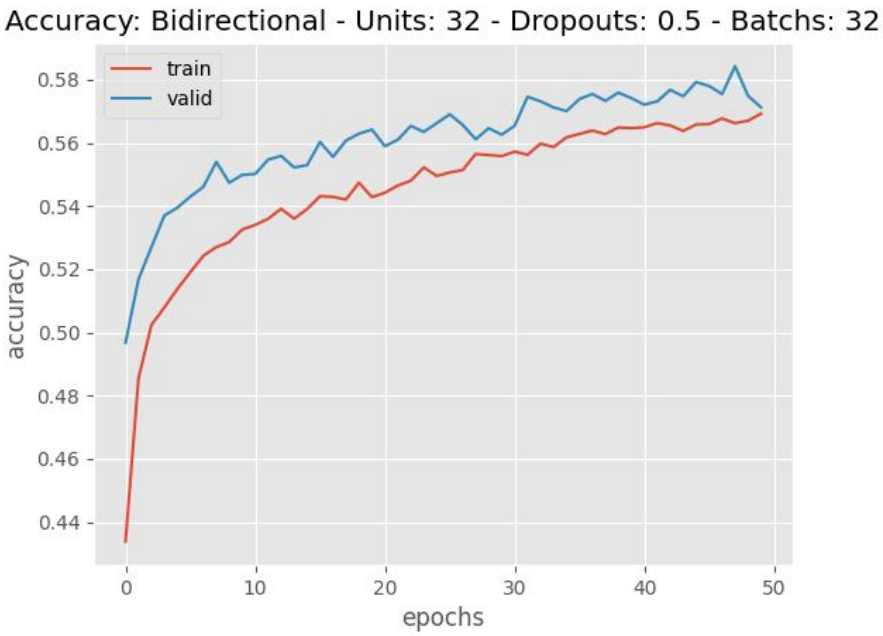


Tabela de Acurácias

Segue as acurácias obtidas no corpo de teste referentes aos gráficos citados anteriormente.

LSTM

experimento	Rede	units	dropouts	batches	acc
20	LSTM	128	0	32	60.16%
23	LSTM	128	0.25	32	60.85%
26	LSTM	128	0.5	32	61.03%

Bidirecional

experimento	Rede	units	dropouts	batches	acc
56	Bideracional	128	0	32	60.42%
59	Bideracional	128	0.25	32	61.52%
62	Bideracional	128	0.5	32	56.96%

Discussão de Resultados

Nas redes apresentadas concluimos sobre underfitting overfitting:

- Os gráficos apresentados não retratam underfitting.
- Nas redes LSTM indicados nos gráficos, somente o dropout de 0.5 resolveu o overfitting. Mas mesmo assim, quando pegamos os melhores modelos das redes com overfitting e aplicamos no corpo de teste elas ainda apresentam bons resultados de acurácia.
- Nas redes Bidirecionais os dois dropouts (0.25 e 5) evitam o overfitting. Como nas redes LSTM, os melhores modelos das redes com overfitting apresentam bons resultados de acurácia no corpo de teste.

.

As explorações iniciais (vide arquivo experimentos iniciais.pdf), indicaram que as redes não teriam uma alta performance (acurácia acima de 85%) .

Após o grid search como indicado anteriormente, foi confirmada a suspeita que as redes não teriam uma acurácia acima que 61% no corpo de teste.

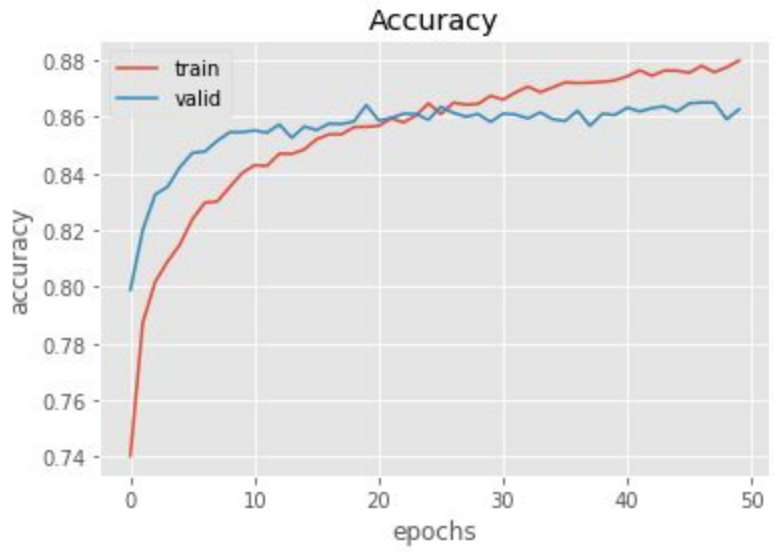
Novos experimentos, agora com um pré-processamento bem simples (letras minúsculas e tokenização), produziram a mesma performance de 61%.

O tamanho da frase também foi variado entre 40, 50 e 60 palavras e sem qualquer incremento ou decremento na performance das redes.

A variação dos hiper-parâmetros não conseguiu produzir uma acurácia acima 61%, como pode-se ver no grid search. As buscas tiveram impacto mínimo na acurácia, após 72 experimentos tivemos uma acurácia média de 60,2% e um %CV de 1,09% (quando tiramos 4 outliers de 72 no total).

Existia uma dúvida se a rede custava performar por conta do tamanho do corpus ou pela complexidade de aprender 5 labels (1-5 estrelas) com este tamanho de corpus. Para tirar esta dúvida, e ter certeza se o dataset era suficiente para o aprendizado de um label binário convertemos os labels {1,2,3} = 0 e (4,5) = 1 e o resultado foi:

Com o resultado acima entendemos que as nuances entre 5 labels não pode ser capturada pelas várias redes testadas, agora quando aumentamos a discrepância entre os labels (binário) a rede aprendeu muito mais, chegando a ter 86% de acurácia, um aumento de 41% frente ao resultado inicial, vide o gráfico abaixo:



Concluímos assim que para este corpus e estes labels, as redes RSTM e Bidirecional não apresentam uma alta performance.

Acredito que para um estudo de overfitting e underfitting este EP cumpriui seu papel, caso desejemos ir mais a fundo na classificação de sentimentos com labels de 1-5 vale um estudo do artigo de Ayu Khusnul Khotimah e Sarno (2019).

Referências

AYU KHUSNUL KHOTIMAH, Dewi; SARNO, Riyanarto. Sentiment Analysis of Hotel Aspect Using Probabilistic Latent Semantic Analysis, Word Embedding and LSTM. **International Journal of Intelligent Engineering and Systems**, Fukuoka, Japão, ano 2019, v. 12, ed. 4, p. 275-290, 24 jun. 2019.