

INTELIGÊNCIA ARTIFICIAL REDE NEURAL: CLASSIFICAÇÃO DE SPAM

Kalil Saldanha Kaliffe¹,Guilherme Farias da Silva Leite¹,Marcos Eduardo Nascimento Lima¹

¹Instituto de Ciências Exatas e Naturais - Faculdade de Computação
Universidade Federal do Pará (UFPA) - Belém, PA - Brasil

{kalil.kaliffe, guilherme.leite, marcos.lima}@icen.ufpa.br

Abstract. *This article aims to implement and run a classification algorithm in an artificial neural network using the KDD method on top of the database obtained from the UCI website repository as a step in the evaluation of the Artificial Intelligence discipline, taught by Professor Reginaldo Santos. The algorithm was applied to the database and metrics were analyzed, including accuracy, confusion matrix, sensitivity, specificity, positive and negative reliability.*

Resumo. *Esse artigo tem o intuito de implementar e executar um algoritmo de classificação em rede neural artificial se utilizando do método KDD em cima da base de dados obtida no repositório do site UCI como etapa da avaliação na disciplina de Inteligência Artificial, ministrada pelo professor Reginaldo Santos. O algoritmo foi aplicado sobre a base de dados e analisadas métricas que incluem a acurácia, matriz de confusão, sensibilidade, especificidade, confiabilidade positiva e negativa.*

1. Introdução

Redes neurais são implementações artificiais do que buscam expressar com excelência o funcionamento do cérebro humano, ou seja, é uma tecnologia que busca assemelhar neurônios com nós e ligações entre os mesmos. Essa tecnologia é utilizada na área de inteligência artificial como forma de aprendizado de máquina, pois a mesma é de suma importância e bastante eficiente quando se trata de resolução de problemas complexos e não lineares. Assim como no cérebro humano, onde vários neurônios estão interligados e se comunicando entre si por meio de pulsos elétricos, a RNA utiliza de seus nós para descobrir padrões e “aprender” baseado em dados preestabelecidos pelo programador.

O trabalho em questão, objetiva executar o método KDD na base de dados escolhida, ao se aplicar o KDD, ele fornece as condições necessárias para encontrar o conhecimento de dados relevantes na base dados através de algoritmos de mineração de dados, isso será realizado por meio da classificação dos dados que é uma técnica de aprendizado supervisionado. Essa técnica consiste na execução em duas etapas sendo a primeira apresentar uma base de treino, na qual o algoritmo consegue assimilar as informações e aprender, e após isso a segunda etapa sendo a base de teste em que se pode realizar a classificação de amostras analisando os seus atributos.

Dessa forma o algoritmo de mineração utilizado foi o de redes neurais, com o intuito de comparar e analisar os resultados obtidos por meio da implementação de uma rede neural recorrente (LSTM) e (RNN), está última sendo um tipo de rede neural que inclui dados dentro de uma mesma camada. Utilizando uma base de dados textual, o algoritmo terá como objetivo detectar um padrão e mostrar um resultado com excelência.

2. Descrição da base de dados

A base de tipo texto selecionada na implementação foi a YouTube Spam Collection Data Set Figura 1, a qual tem como objetivo classificar e realizar uma pesquisa de spam nos comentários coletados a partir de cinco conjuntos de dados compostos por mais de 1.900 vídeos que estiveram entre os 10 mais vistos no período de 26/03/2017.

A base possui um total de 5 atributos, 1956 amostras e não tem valores faltando se utilizando da API de dados do Youtube v3.

	COMMENT_ID	AUTHOR	DATE	CONTENT	CLASS
0	LZQPQhLyRh80UYxNuaDWhtGQYNO96luCg-AYWqNPjpU	Julius NM	2013-11-07T06:20:48	Huh, anyway check out this you[tube] channel: ...	1
1	LZQPQhLyRh_C2cTld9MvFRJedxydaVW-2sNg5Diuo4A	adam riyati	2013-11-07T12:37:15	Hey guys check out my new channel and our firs...	1
2	LZQPQhLyRh9MSZynf8djk0gEF9BHDpYnK-qCczY8	Evgeny Murashkin	2013-11-08T17:34:21	just for test I have to say murdev.com	1
3	z13jhp0bxqncu512g22wzksxmvvzjaz04	ElNino Melendez	2013-11-09T08:28:43	me shaking my sexy ass on my channel enjoy ^_^	1
4	z13fwbwp1oujthgqj04chlingpvzmtl3r3dw	GsMega	2013-11-10T16:05:38	watch?v=vtaRGgvGIWQ Check this out .	1
...
1951	_2viQ_Qnc6-bMSjqyL1NK57ROicCSJV5SwTnw-RFFA	Katie Mettam	2013-07-13T13:27:39.441000	I love this song because we sing it at Camp al...	0
1952	_2viQ_Qnc6-pY-1yR6K2FhmC5i48-WuNx5CumIHLDAI	Sabina Pearson-Smith	2013-07-13T13:14:30.021000	I love this song for two reasons: 1.it is abou...	0
1953	_2viQ_Qnc6_k_n_Bse9zVhJP8JReZpo8uM2uZfnzDs	jeffrey jules	2013-07-13T12:09:31.188000	wow	0
1954	_2viQ_Qnc6_yBt8UGMWyg3vh0PuTtqcqyQldE7d4FI0	Aishlin Maciel	2013-07-13T11:17:52.308000	Shakira u are so wiredo	0
1955	_2viQ_Qnc685RPw1aSa1tfrtuHXRAQ2rPT9R06KTqA	Latin Bosch	2013-07-12T22:33:27.916000	Shakira is the best dancer	0
1956 rows x 5 columns					

Figura 1. Tabela da base de dados de comentarios

Na Figura 1, cada instância é um comentário que possui um 'COMMENT ID' que é a identificação única do comentário, 'AUTHOR' o nome da conta google que fez o comentário, 'DATE' data de postagem do comentário, 'CONTENT' uma String com o texto do comentário e 'CLASS' um Int que o rótulo do comentário, considerando o valor 1 para Spam e 0 para Ham (não Spam), sendo 951 Spams 1005 Hams no total.

3. Metodologia do trabalho

A base de dados foi primeiro carregada para um Gist em formato csv e convertida para um Dataframe Pandas para poder ser navegada e selecionada.

Antes de tudo, para permitir uma boa entrada para etapa de mineração de dados, é necessário normalizar os dados através de etapas KDD.

Assim como a primeira etapa é selecionar os dados de interesse, a partir da análise da base de dados representada na Figura 1, nota-se que as colunas relevantes para a descoberta e assim a classificação do spam são as colunas CONTENT, o texto do comentado e o seu label CLASS, assim as etapas de seleção e pré-processamento dos dados consistem em obter uma lista de comentários e uma lista de labels(0 ou 1).

A próxima etapa é transformar os dados, uma vez que a RNN utilizada não aceita Strings. Logo, foi utilizado um Tokenizer para converter os comentários em String

em vetores, assim a vetorização vai criar tokens para índices que mapeam os textos em sequências.

Além disso os comentários e labels devem ser divididos em listas de teste e treino, sendo 70% treino e 30% teste fatiando as listas originais.

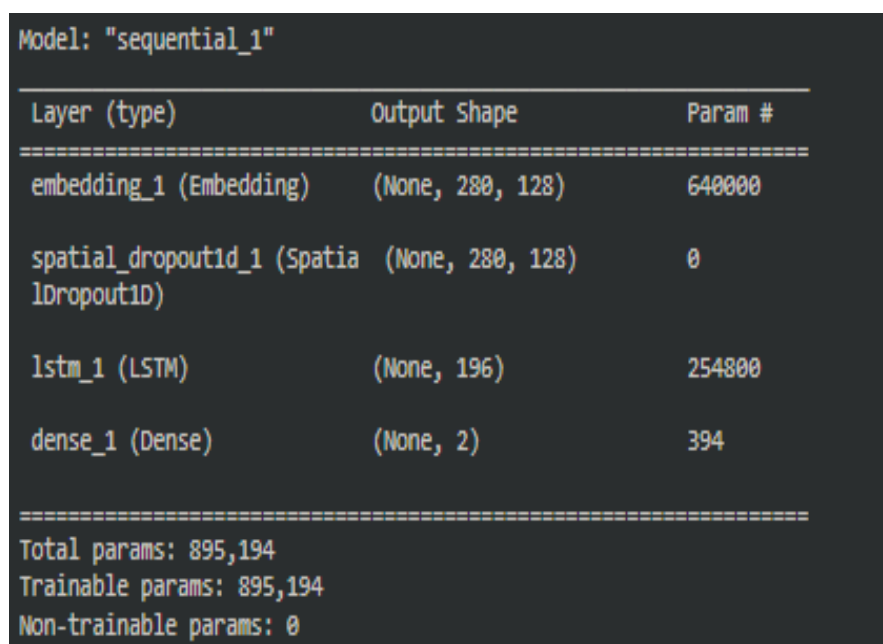
A partir desse ponto as camadas do modelo sequencial da Keras podem ser adicionadas. A primeira camada é uma camada de Embedding, a qual transforma os índices do input em vetores densos e recebe o número máximo de palavras e as dimensões do lista de comentários.

A segunda camada é uma camada de SpatialDropout1D, a qual recebe uma frequência com a qual vai dropar um mapa inteiro de features ao invés de só um elemento com o Dropout normal.

A terceira camada é a camada do LSTM que recebe frequências de Dropout.

A quarta camada é uma camada dense que retorna uma ativação Softmax que comprime os paramentos.

A partir disso, o modelo foi criado e agora deve ser compilado com suas metricas que medem a classificação, as quais vão ser a acurácia, perda, falso negativos, falso positivos, verdadeiro negativo e verdadeiro positivo, o resultado dessa compilação pode ser visto na Figura 2.



```
Model: "sequential_1"
Layer (type)                Output Shape              Param #
=====
embedding_1 (Embedding)      (None, 280, 128)         640000
spatial_dropout1d_1 (SpatialDropout1D) (None, 280, 128)         0
lstm_1 (LSTM)                (None, 196)              254800
dense_1 (Dense)              (None, 2)                394
=====
Total params: 895,194
Trainable params: 895,194
Non-trainable params: 0
```

Figura 2. Modelo sequencial compilado

Então, o modelo exposto na Figura 2, vai ser treinado usando `.fit` que recebe as listas de comentários e labels de treino e teste, o tamanho do batch(o número de amostras até a atualização do modelo) e o número de epochs(o número de loops completos no conjunto de dados).

Assim, o modelo pode ser avaliado usando `.evaluate` que retorna as métricas apresentadas na Figura 3.

	Test loss	Test accuracy	Especificidade	Sensibilidade	Matriz de confusao
0	0.187948390841484	0.9301533102989198	0.93015332197615	0.93015332197615	[[587 294] [293 0]]
1	0.1823361865010452	0.9420783519744872	0.9420783645655876	0.9420783645655876	[[587 294] [293 0]]
2	0.1828040480613708	0.9318568706512452	0.9318568994889268	0.9318568994889268	[[587 294] [293 0]]
3	0.204947829246521	0.9199318289756776	0.919931856899489	0.919931856899489	[[587 294] [293 0]]
4	0.1810419261455536	0.9369676113128862	0.9369676320272572	0.9369676320272572	[[587 294] [293 0]]
5	0.1787808239459991	0.9335604906082152	0.9335604770017036	0.9335604770017036	[[587 294] [293 0]]
6	0.186933159828186	0.9301533102989198	0.93015332197615	0.93015332197615	[[587 294] [293 0]]
7	0.1739915460348129	0.9369676113128862	0.9369676320272572	0.9369676320272572	[[587 294] [293 0]]
8	0.1813649237155914	0.9335604906082152	0.9335604770017036	0.9335604770017036	[[587 294] [293 0]]
9	0.1710256785154342	0.9352640509605408	0.9352640545144804	0.9352640545144804	[[587 294] [293 0]]

Figura 3. Tabela com as métricas obtidas das bases em 10 execuções

4. Resultados

O algoritmo foi executado dez vezes (Figura 3)) em cima da base de dados, e a cada ciclo de execução o resultado foi particionado entre os dados de teste e os dados de treino. No cenário exposto, 30% os dados foram utilizados como unseen data para julgar a eficiência do algoritmo, enquanto os outros 70% para de fato treinar o algoritmo. A cada ciclo de execução, foram utilizadas alterações distintas de dados de treino e de teste da base de dados para maior diversificação dos dados de entrada. Após isso, seis métricas (Figura 3) foram coletadas para cada execução: matriz de confusão, sensibilidade, especificidade, confiabilidade positiva, confiabilidade negativa e acurácia total. Utilizaram-se as bibliotecas random, numpy, os e sys para o cálculo das métricas e para a divisão da base de dados, respectivamente, e para a plotagem da informações foram utilizados o matplotlib.pyplot, pandas, seaborn.

5. Análise comparativa

Para determinar qual o melhor modelo RNN treinado, podemos considerar as métricas da Figura 3), uma vez que, o modelo 4 apesar de não possuir a maior precisão ainda está segundo colocado em precisão, enquanto ainda possui a quarta menor perda e a segunda melhor especificidade, pode ser definido como o melhor modelo superando o modelo 1 que apesar de ter a melhor precisão não tem uma perda proporcionalmente baixa.

A Figura 4, mostra o treino epoch por epoch do modelo sequencial 4.

Além disso, analisando precisões dos modelos na Figura 3, temos que a media das acurácias é de 0.9330493927, é o desvio padrão da amostra é de 0.00584922888154 sendo que o desvio padrão da população é de 0.00554906574639.

6. Conclusão

De modo sucinto, o trabalho se comprometeu em comparar o desempenho do algoritmo de mineração de dados utilizados na estratégia KDD, onde o objetivo era coletar e classificar se um comentário pode ser definido como spam ou ham. Foram descritas as implementações e características utilizadas nos algoritmos, assim coma base de dados.

```

Layer (type)                Output Shape                Param #
-----
embedding_4 (Embedding)     (None, 280, 128)          640000
spatial_dropout1d_4 (SpatialDropout1D)  0
lstm_4 (LSTM)               (None, 196)               254800
dense_4 (Dense)             (None, 2)                 394
-----
Total params: 895,194
Trainable params: 895,194
Non-trainable params: 0
None

a built-in Keras object. Consider renaming <class
Epoch 1/5
43/43 [=====] - 72s 2s/step - loss: 0.5659 - cat_accuracy: 0.6969 - accuracy: 0.0000e+00 - precision: 0.6969 - recall: 0.6969 - true
Epoch 2/5
43/43 [=====] - 68s 2s/step - loss: 0.2478 - cat_accuracy: 0.9262 - accuracy: 0.0245 - precision: 0.9262 - recall: 0.9262 - true
Epoch 3/5
43/43 [=====] - 69s 2s/step - loss: 0.1593 - cat_accuracy: 0.9372 - accuracy: 0.0000e+00 - precision: 0.9372 - recall: 0.9372 - true
Epoch 4/5
43/43 [=====] - 68s 2s/step - loss: 0.1322 - cat_accuracy: 0.9496 - accuracy: 0.0000e+00 - precision: 0.9496 - recall: 0.9496 - true
Epoch 5/5
43/43 [=====] - 67s 2s/step - loss: 0.1027 - cat_accuracy: 0.9598 - accuracy: 0.0000e+00 - precision: 0.9598 - recall: 0.9598 - true
19/19 [=====] - 4s 201ms/step - loss: 0.2049 - cat_accuracy: 0.9199 - accuracy: 0.0000e+00 - precision: 0.9199 - recall: 0.9199
Matriz de confusao:
[[547 294]
 [293  0]]
test_loss(Confiabilidade Negativa): 0.204947829246521
test_accuracy(Confiabilidade Positiva): 0.0
test_categorical_accuracy(Confiabilidade Positiva): 0.9199318289756775
test_precision(Confiabilidade Positiva): 0.9199318289756775

test_true_negatives: 540.0
test_false_positives: 47.0
test_true_positives: 540.0
test_false_negatives: 47.0

Especificidade: 0.919931856899489
Sensibilidade: 0.919931856899489

```

Figura 4. Treino do modelo sequencial 4

observado através de métricas como acurácia e taxa de erro total, obtidas a partir de uma matriz de confusão, e que o algoritmo de Rede Neural Artificial implementado obteve resultado satisfatório.

7. Anexos

Código-Fonte: https://colab.research.google.com/drive/1TWz0akYw12_JBoJBsBluRB3r6k2nkJQm?usp=sharing.

Fonte Base de Dados: <https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection>.

Referências