|  |
| --- |
| Caio Feiertag (1510590) - Lorenzo Saraiva (1110649) - Johann CORCUFF-REBECCHI (1720167) |
| **Inteligência Artificial** |
| Projeto 3 |

10/12/2017

INF1771



# Index

[**I. Introduction**](#_1blam5wmt09y) **3**

[**II. Definition of the problem**](#_408nzrne449b) **3**

[**III. Solutions implemented**](#_2zz3grz3uid1) **3**

[A. Neural network](#_rla0vw65mkpq) 3

[How does it work ?](#_8011yk6r5g99) 3

[Implementation](#_jhjvs89ih2po) 3

[Attributes selected](#_sim96h3tstv6) 4

[Experiences realised](#_t28fj6iufkfp) 4

[Results](#_kqi0mige58bj) 5

[B. Decision tree](#_cr89t0ags0cf) 5

[How does it work ?](#_qajp9v20sgj6) 5

[Implementation](#_jruxzirbe0vr) 6

[Attributes selected](#_3ixqpa67uzns) 6

[Experiences realised](#_v7ikylb8zrok) 6

[Results](#_fck6tbkzq6w) 7

[**IV. Conclusion**](#_h40rv5j0nued) **8**

# 

# 

# I. Introduction

Machine learning is a subject that was born with computer science since the 50s. With faster and faster computers, the development of these technologies and its incredible usage today, makes Machine Learning a major subject of today’s engineering.

# II. Definition of the problem

The goal of this project is to be able to build a model that can predict if a movie review is positive or negative. For this purpose we have 2 datasets of positive and negative reviews. We will use the first one to train the dataset and the second one one to test it.



# III. Solutions implemented

## A. Neural network

### How does it work ?

Para poder ler os dados mais rápido, usamos o arquivo file\_reader.py para colocar todas as reviews negativas e positivas agrupadas em poucos arquivos. Basta executar a file\_reader.py na mesma pasta dos arquivos a serem lidos, e ele criará um txt com o conteúdos dos arquivos separados por uma quebra de linha. Esse arquivos estão incluídos no projeto para facilitar a execução. Basta executar todo o código do arquivo bagofwords.py que ele fará o treinamento e teste da rede neural.

### Implementation

Para implementar a rede neural foram usados os frameworks Keras e Sklearn. A rede tem 2 camadas escondidas além da camada de input, com diferentes ratios de dropout. O código foi baseado parcialmente em um tutorial da Udemy sobre redes neurais. Para interpretar os dados foi usado o modelo bag of words, com 500 palavras únicas

### Attributes selected

Palavras de opinião.

### Experiences realised

A primeira rede tinha apenas 1 hidden layer, com 2500 neurônios. Essa quantidade de dados se mostrou um problema - uma tabela (25000,5000) ocupava boa parte da memória do computador - e o treinamento levava muito tempo. Colocamos um batch size = 25 e epochs = 100 e o modelo ficou treinando durante 25 horas aproximadamente. Os resultados não foram muito bons, aproximadamente 51%.

Foi testado com 5000, 7500 e 10000 palavras únicas, e batch sizes entre 10-100, epochs entre 10-100, 1 e 2 layers. Independente desses valores, a accuracy sempre subia muito rápido, logo no primeiro ou segundo loop alcançando 0.9~, e no teste do treino algo ao redor de 85%. Porém, ao fazer as previsões dos dados de teste, a previsão sempre caía em algo entre 51-53%, pouco acima de uma previsão aleatória, o que indicava que havia algo errado, porém sempre acima de 50%, o que indicava que o treinamento estava funcionando em uma proporção pequena. Isso indicava um caso de overfitting - onde o modelo se adapta "bem demais" aos dados de treino, e na hora de prever dados reais acaba errando, pois acaba ficando muito específico para o caso de treino.

Tentamos diferentes abordagens para resolver isso. Primeiro, colocar mais layers, e mudar a quantidade de neurônios em cada um, o que foi inefetivo. Depois, colocamos dropouts - que aleatoriamente descarta uma porcentagem de neurônios - uma abordagem comum contra overfitting, mas também não afetou os resultados. Tentamos trocar a função de optimização da Adam para a Stochastic gradient descent. Aplicamos também early stop, que faz alguns loops do treino serem interrompidos no meio, outra técnica usada para prevenir overfitting que faz com que o treino seja interrompido mais cedo de acordo com um certo valor monitorado, mas que também não funcionou.

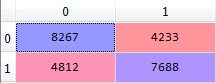
Finalmente, percebemos que o problema poderia ser ter o modelo de dados complexo demais, e diminuimos o número de palavras únicas para 1000. Isso fez uma grande diferença. A accuracy se manteve ao redor de 0.84 nos testes, e na hora de testar com o conjunto de testes obteve a pontuação de 62%, com batch size de 100 e 100 epochs. Buscando simplificar ainda mais o modelo, retiramos todas as palavras que não eram de opinião, que poderiam ser consideradas como ruído, e diminuimos o número de palavras únicas para 500. Com bs = 300 e epoch = 100, foi obtida a accuracy de 64% no teste.

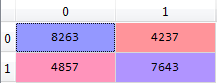
Para otimizar a rede, foi tentado uma classe de otimizador do Keras, que testou a rede com diferentes parâmetros - epochs de 50, 100 e 150, batch size de 50, 100 e 250 e funções adam, rmsprop e sgd. O otimizador acabou retornando como melhores parâmetros valores parecidos com os já utilizados - bs = 50, epochs = 100 e o otimizador adam, porém o resultado nos testes acabou sendo muito parecido ao do anterior, ao redor de 64%. Esse otimizador se encontra no arquivo bagofwords\_otimizer.py. Também foi tentado adicionar outras camadas, porém sem influência no resultado.

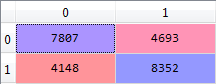
### Results

O melhor resultado ficou ao redor de 64%, onde se mantém de forma bastante estável.

Matrizes de resultados :







## B. Decision tree

The decision tree algorithm has been implemented using the open source software Weka. It uses specifically the J48 algorithm which is a Java adaptation of c4.5 (<https://en.wikipedia.org/wiki/C4.5_algorithm>)

### How does it work ?

J48 is an algorithm producing a decision tree based on the concept of information entropy, which measure the “the average amount of information produced by a stochastic source of data” (<https://en.wikipedia.org/wiki/Entropy_(information_theory)>)

Thus, it will measure the entropy of every attribute of the test data to figure out which attribute give the highest rate of information to create the best decision tree. J48 use some pruning technique to make the tree simpler, readable and more efficient.

### Implementation

In order to use Weka, we had to generate two .arff files, one for the training data, with the reviews stored in part1, and one for the test data with the reviews stored in part2. The “Arff\_generator.py” is a simple Python script that will generate those 2 files from the data folders.

### Attributes selected

For each review the following attributes have been calculated :

* The number of words
* The number of positive words
* The number of negative words
* The percentage of positive words over the total
* The percentage of negative words over the total
* The difference between the number of negative and positive words

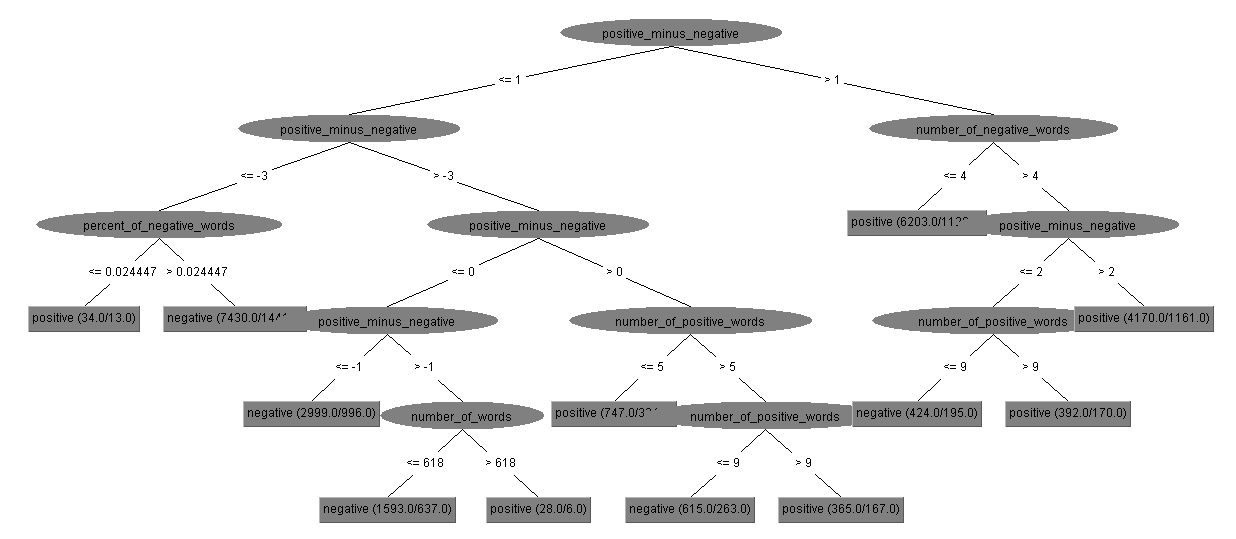
We have of course as a final attribute, the type of review that is either positive or negative, that will allow Weka with the J48 algorithm to build the data tree with the training data set and check its validity with the test data set.

In order to calculate the number of positive words and negative words in each review, we had to download a dictionary of negative english words and positive english words. We could find them at the following link : <http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>. The negative words dictionary contains around 4800 entries, and the positive around 2000.

Thus our Arff\_generator.py script compares each words of each reviews with those of these two dictionaries, count them and record the results in the dataset arff files.

### Experiences realised

We used the J48 algorithm to build the following tree:



We tried to play with the parameters of J48, as for example the minimum size of each leaf, the prune level of the tree, but it did not improve the results. Trying other decision tree algorithms did not improved the result neither.

As explained by the professor Ian Witten, in the MOOC “Data Mining with Weka” (<https://www.futurelearn.com/courses/data-mining-with-weka>), this j48 is the best decision tree algorithm of this software and its default parameters are well set in order to produce the best results.

### Results

Thus we could have the following results :

|  |  |
| --- | --- |
| Correctly Classified Instances 73.184 % | Incorrectly Classified Instances 26.816 % |

With the following confusion matrix :

|  |  |  |
| --- | --- | --- |
| Positive | Negative | Classified as |
| 8875 | 3625 | Positive |
| 3079 | 9421 | Negative |

We can see that there significantly more errors for negative reviews classed as positive than positive class as negative. Indeed, many negative reviews can contains positive words, sometime to empower the negative ones that are used.

This word to word analysis could be improved if we were analysing expressions or group of words. For example : if a positive word is used before a negative one, this word would not count as positive.

# IV. Conclusion

Ambos os métodos escolhidos mostraram sucesso, com o segundo tendo um resultado melhor. Isso pode se dever a implementação da rede neural ou ao segundo modelo se adaptar melhor ao nosso conjunto de dados. Também pode ter sido uma questão de limite de poder computacional, pois a rede neural poderia ter sido expandida ainda mais, mas esse fator foi limitante.