

# Trabalho Final de Inteligência Artificial

**Nomes:** Arthur Sudbrack Ibarra, Felipe Grozse Nipper de Oliveira, Miguel Torres de Castro e Willian Magnum Albeche

## 1. Funcionamento

Para que a análise de sentimentos pudesse ser feita, o grupo optou por utilizar a linguagem de programação Python, em conjunto com a ferramenta Weka. Dessa forma, um script Python chamado *generator.py* foi criado, o qual é responsável por ler o arquivo fornecido pela professora com os dados de entrada já pré-processados e gerar arquivos .arff válidos que o Weka possa interpretar.

É possível informar ao script quantas palavras serão consideradas para o bag of words, bem como quais palavras devem ser desconsideradas. Inclusive, algumas palavras como "dell", "notebook" e "inspiron" de fato foram desconsideradas, visto que, após uma análise no Weka, o grupo percebeu que se tratavam de palavras neutras, que não agregavam para o treino do modelo, uma vez que podem ser usadas tanto em contextos positivos como negativos.

O uso do script se dá da seguinte forma:

```
python generator.py <N_PALAVRAS>
```

N\_PALAVRAS = Quantas palavras serão usadas para o bag of words.

70% dos dados são usados para treino e 30% para teste, sendo que a distribuição dos dados é feita de forma aleatória. De 70% dos dados, o arquivo de treino somente inclui frases do dataset fornecido que contenham pelo menos 1 palavra do bag of words. Isso porque o grupo não viu sentido em treinar o modelo com linhas totalmente zeradas, já que estas não indicam/ensinam nada significativo ao modelo. Em contrapartida, no arquivo de teste, tal restrição não existe.

## Exemplo de execução do algoritmo:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [ ] [X] ^ X

PS C:\Users\ibarrart\Documents\Faculdade\TF-Inteligencia-Artificial> python generator.py 90

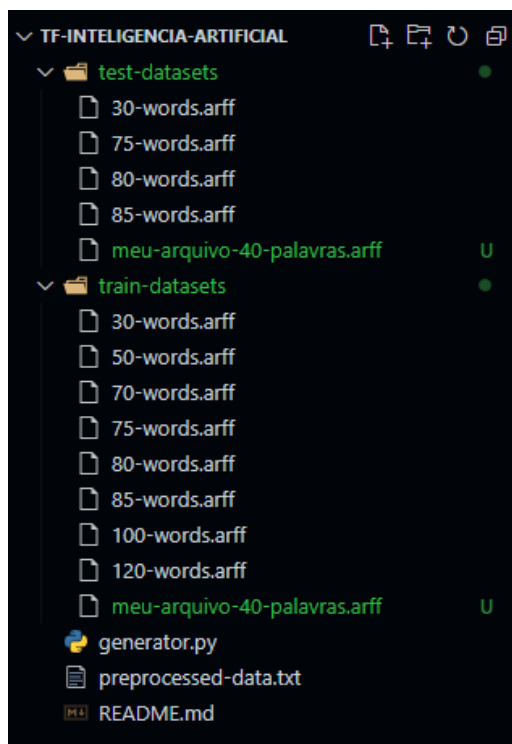
Palavras sendo desconsideradas: ['dell', 'not', 'notebook', 'http', 'window', '10', '3', '15', '2', 'inspiron', '1', '%', 's', 'tou']

Arquivos gerados com sucesso:

train-datasets/90-words.arff: Arquivo de treino.
test-datasets/90-words.arff: Arquivo de teste.

PS C:\Users\ibarrart\Documents\Faculdade\TF-Inteligencia-Artificial>
```

## Exemplo de arquivos ARFF sendo gerados:



Exemplo de arquivo de treino gerado com 30 palavras:

```
generator.py M 30-words.arff U x
train-datasets > 30-words.arff
1 @relation emotionAnalysis
2
3 @attribute compr {0,1}
4 @attribute pra {0,1}
5 @attribute nov {0,1}
6 @attribute problem {0,1}
7 @attribute reclam {0,1}
8 @attribute aqu {0,1}
9 @attribute quer {0,1}
10 @attribute brasil {0,1}
11 @attribute comput {0,1}
12 @attribute reclameaqu {0,1}
13 @attribute dellnobrasil {0,1}
14 @attribute nunc {0,1}
15 @attribute dá {0,1}
16 @attribute car {0,1}
17 @attribute q {0,1}
18 @attribute ta {0,1}
19 @attribute ano {0,1}
20 @attribute olh {0,1}
21 @attribute dellajud {0,1}
22 @attribute ach {0,1}
23 @attribute ness {0,1}
24 @attribute troc {0,1}
25 @attribute vou {0,1}
26 @attribute deu {0,1}
27 @attribute nao {0,1}
28 @attribute atual {0,1}
29 @attribute boa {0,1}
30 @attribute lind {0,1}
31 @attribute agor {0,1}
32 @attribute dess {0,1}
33 @attribute class {-1,1}
34
35 @data
36 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
37 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
38 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1
39 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1
40 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
```

## **2. Resultados**

Após gerados os arquivos .arff, fornecemos eles à ferramenta Weka para que pudéssemos utilizar diferentes modelos de classificação e ver o desempenho destes com variadas configurações.

### **2.1. KNN (IBk)**

**Quantidade de palavras no bag of words:** 85.

**Palavras sendo desconsideradas:** ['dell', 'not', 'notebook', 'http', 'window', '10', '3', 'inspiron', '1', '%', '"s"', '"tou"']

**Tipo do teste:** Conjunto de teste fornecido.

**Vizinhos próximos sendo considerados:** 5.

## Resultado:

The screenshot shows the Weka Explorer interface. The 'Classify' tab is active. The classifier selected is 'IBk -K 10 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""'.

**Test options:**

- ☐ Use training set
- ☒ Supplied test set (Set...)
- ☐ Cross-validation (Folds: 10)
- ☐ Percentage split (%: 66)

**(Nom) class:** Start Stop

**Result list (right-click for options):**

- 23:30:23 - lazy.IBk
- 23:33:49 - lazy.IBk
- 23:34:03 - lazy.IBk (selected)
- 23:35:59 - lazy.IBk
- 23:36:14 - lazy.IBk
- 23:36:24 - lazy.IBk
- 23:36:32 - lazy.IBk
- 23:36:38 - lazy.IBk
- 23:36:44 - lazy.IBk
- 23:36:54 - lazy.IBk
- 23:39:06 - lazy.IBk
- 23:39:26 - lazy.IBk
- 23:39:36 - lazy.IBk

**Classifier output:**

```
err
estrag
format
driv
lig
tÃ¡;
fod
assist
deslig
cheg
tÃ¡'
class

Test mode: user supplied test set: size unknown (reading incrementally)

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 5 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.05 seconds

=== Summary ===

Correctly Classified Instances      125      68.306 %
Incorrectly Classified Instances    58      31.694 %
Kappa statistic                    0.1328
Mean absolute error                 0.4158
Root mean squared error             0.4531
Relative absolute error             95.4667 %
Root relative squared error         96.4726 %
Total Number of Instances          183

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
0,927    0,817    0,699    0,927    0,797    0,166    0,693    0,832    -1
0,183    0,073    0,550    0,183    0,275    0,166    0,693    0,474    1
Weighted Avg.    0,683    0,573    0,650    0,683    0,626    0,166    0,693    0,715

=== Confusion Matrix ===

  a  b  <-- classified as
114  9  |  a = -1
 49 11 |  b = 1
```

**Status:** OK

Conforme pode ser observado na figura acima, o modelo acertou 68,306% das classificações, errando 31,694% dos palpites. Outros testes foram feitos com bag of words de tamanhos diferentes e conjuntos de treino e teste distintos, porém a taxa de acerto do modelo manteve-se na faixa de 64-68%. Ademais, o grupo notou que analisando os **5** vizinhos mais próximos para definir a classe de um elemento, o modelo obteve um resultado melhor do que quando esse valor era mais elevado, como **10**, ou mais baixo, como **2**.

## 2.2. MultilayerPerceptron

**Quantidade de palavras no bag of words:** 85.

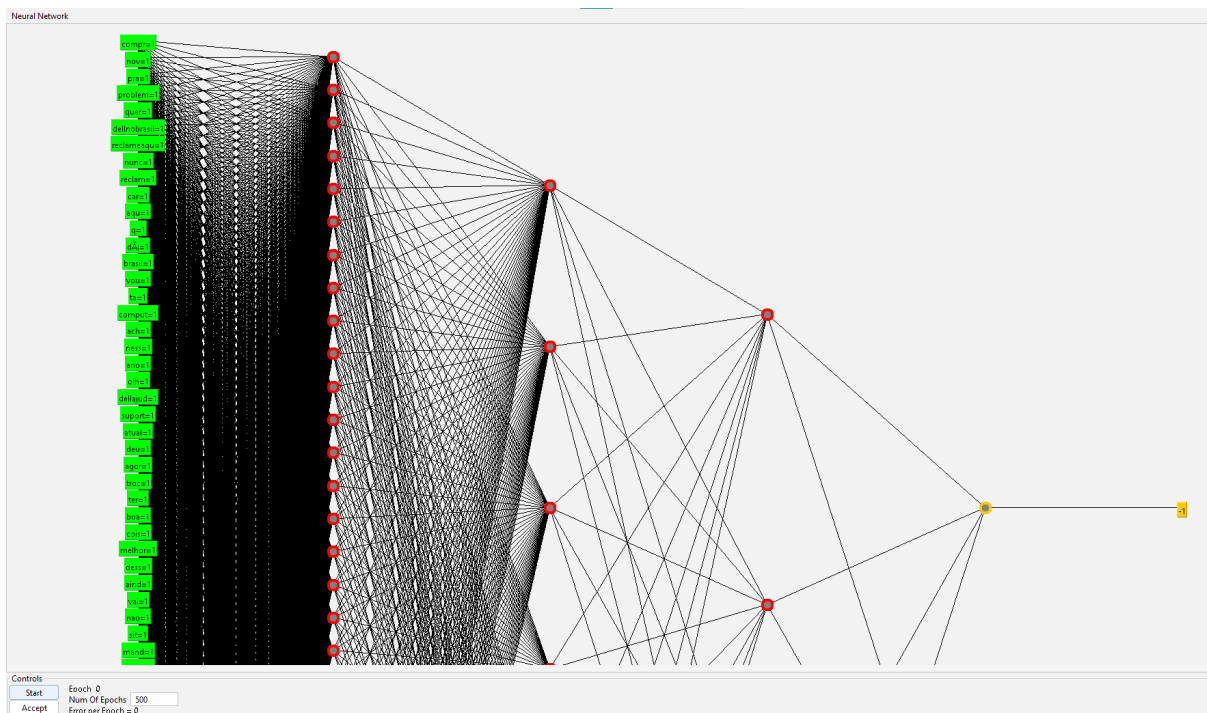
**Palavras sendo desconsideradas:** ['dell', 'not', 'notebook', 'http', 'window', '10', '3', 'inspiron', '1', '%', 's', 'tou']

**Tipo do teste:** Conjunto de teste fornecido.

**Quantidade de épocas:** 2000.

**Taxa de aprendizado:** 0,1

**Hidden layers:** (8,4) 1 hidden layer com 8 neurônios e 1 hidden layer com 4 neurônios.



## Resultado:

Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier

Choose **MultilayerPerceptron** -L 0.1 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H "a, 8, 4" -G -R

Test options

☐ Use training set

☒ Supplied test set

☐ Cross-validation Folds 10

☐ Percentage split % 66

(Nom) class

Result list (right-click for options)

- 23:30:23 - lazy.IBk
- 23:33:49 - lazy.IBk
- 23:34:03 - lazy.IBk
- 23:35:59 - lazy.IBk
- 23:36:14 - lazy.IBk
- 23:36:24 - lazy.IBk
- 23:36:32 - lazy.IBk
- 23:36:38 - lazy.IBk
- 23:36:44 - lazy.IBk
- 23:36:54 - lazy.IBk
- 23:39:06 - lazy.IBk
- 23:39:26 - lazy.IBk
- 23:39:36 - lazy.IBk
- 00:01:23 - functions.MultilayerPerceptron
- 00:02:45 - functions.MultilayerPerceptron
- 00:03:31 - functions.MultilayerPerceptron
- 00:04:32 - functions.MultilayerPerceptron
- 00:10:34 - functions.MultilayerPerceptron
- 00:10:56 - functions.MultilayerPerceptron
- 00:11:38 - functions.MultilayerPerceptron
- 00:12:20 - functions.MultilayerPerceptron
- 00:12:38 - functions.MultilayerPerceptron
- 00:13:12 - functions.MultilayerPerceptron
- 00:13:58 - functions.MultilayerPerceptron
- 00:14:15 - functions.MultilayerPerceptron
- 00:14:29 - functions.MultilayerPerceptron
- 00:15:08 - functions.MultilayerPerceptron
- 00:15:22 - functions.MultilayerPerceptron
- 00:15:44 - functions.MultilayerPerceptron
- 00:16:09 - functions.MultilayerPerceptron**

Classifier output

Node 52 -0.23312662898446804

Sigmoid Node 56

Inputs Weights

Threshold -0.40700909690849285

Node 45 -0.22547282893425166

Node 46 -0.15234897143320364

Node 47 -0.18827480922499382

Node 48 -0.15894022320830803

Node 49 -0.16929135303397194

Node 50 -0.2318890045908293

Node 51 -0.22166174467612781

Node 52 -0.23549659532279968

Class -1

Input

Node 0

Class 1

Input

Node 1

Time taken to build model: 40.72 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances	123	67.2131 %
Incorrectly Classified Instances	60	32.7869 %
Kappa statistic	0	
Mean absolute error	0.4338	
Root mean squared error	0.4699	
Relative absolute error	99.596 %	
Root relative squared error	100.0389 %	
Total Number of Instances	183	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1,000	1,000	0,672	1,000	0,804	?	0,664	0,822	-1
	0,000	0,000	?	0,000	?	?	0,664	0,447	1
Weighted Avg.	0,672	0,672	?	0,672	?	?	0,664	0,699	

=== Confusion Matrix ===

a	b	<-- classified as
123	0	a = -1
60	0	b = 1

Status

OK

Conforme mostrado na imagem acima, o modelo de Multilayer Perceptron obteve uma taxa de acerto de 67,213% e uma taxa de erro de 32,786%, obtendo, portanto, um resultado um pouco inferior ao do modelo KNN. O grupo pôde perceber que a configuração das **hidden layers** impactou muito o resultado do modelo, visto que quando só existia 1 hidden layer, a taxa de instâncias classificadas corretamente foi inferior à 50%. Além disso, um fator curioso de comentar é o fato de que o valor da **taxa de aprendizado** não influenciou nos resultados finais do modelo, o que chamou a atenção do grupo, dado que uma taxa de aprendizado pequena implica em saltos de ajuste de peso menores e em um refinamento mais preciso do modelo.

Todavia, quando a taxa de aprendizado foi configurada para **0,01** (baixo) e para **1** (alto), a taxa de acerto do modelo manteve-se a mesma.

## **2.3. RandomForest**

**Quantidade de palavras no bag of words:** 85.

**Palavras sendo desconsideradas:** ['dell', 'not', 'notebook', 'http', 'window', '10', '3', 'inspiron', '1', '%', '"s"', '"tou"']

**Tipo do teste:** Conjunto de teste fornecido.

**Profundidade máxima:** 9.



## Resultado:

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **RandomForest** -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1 -depth 9

Test options

☐ Use training set

☒ Supplied test set

☐ Cross-validation Folds 10

☐ Percentage split % 66

(Nom) class

Result list (right-click for options)

- 11:44:42 - trees.RandomForest
- 11:44:51 - trees.RandomForest
- 11:44:59 - trees.RandomForest
- 11:45:09 - trees.RandomForest
- 11:45:20 - trees.RandomForest
- 11:45:27 - trees.RandomForest
- 11:45:35 - trees.RandomForest**

Classifier output

```
tant
hp
aA-
resolv
faz
err
estrag
format
driv
lig
tA;
fod
assist
deslig
cheg
tA'
class

Test mode: user supplied test set: size unknown (reading incrementally)

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -depth 9 -do-not-check-capabilities

Time taken to build model: 0.03 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances      129           70.4918 %
Incorrectly Classified Instances     54           29.5082 %
Kappa statistic                     0.13
Mean absolute error                  0.3982
Root mean squared error              0.4389
Relative absolute error              91.4324 %
Root relative squared error          93.453 %
Total Number of Instances           183

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
          1,000    0,900    0,695     1,000    0,820     0,264    0,714    0,848    -1
          0,100    0,000    1,000     0,100    0,182     0,264    0,714    0,565     1
Weighted Avg.   0,705    0,605    0,795     0,705    0,611     0,264    0,714    0,755

=== Confusion Matrix ===

  a  b  <-- classified as
123  0  |  a = -1
 54  6  |  b = 1
```

Status  
OK

O último modelo testado foi o de RandomForest, o qual obteve uma taxa de acerto de 70,491% e uma taxa de erro de 29,508%, superando por pouco tanto o modelo de KNN quanto o modelo de Multilayer Perceptron. A configuração observada pelo grupo como a mais importante para o sucesso do algoritmo foi a de **profundidade máxima**, sendo que o valor ideal foi de **9**. Se o valor da profundidade máxima era < 9, a taxa de acerto do modelo caía para < 70%; para valores próximos de 9, como 10 e 11, o resultado mantinha-se em 70,491%; mas quando o valor da profundidade

máxima se distanciava de 9 para cima, as taxas de acerto do modelo voltavam a cair.

### **3. Conclusão**

Ao final dos testes realizados, foi possível perceber que as taxas de acerto dos três modelos testados não foram altas, girando em torno de 67%-70%, não sendo, portanto, suficientes para considerarmos os modelos confiáveis. Isso se dá, muito provavelmente, pelo dataset que foi utilizado para gerar os arquivos de treino e teste, visto que ele era pequeno (604 frases), e que existiam muito mais amostras com classes negativas do que positivas. Além disso, muitas das frases fornecidas, mesmo com um bag of words grande, não tinham nenhum termo do bag of words, resultando em linhas de dados zeradas como entradas durante a fase de teste, as quais acabavam prejudicando a taxa de acerto do modelo, uma vez que nessas situações o modelo acaba tendo de classificar a instância sem ter nenhuma informação relevante.