

# Trabalho Final de Inteligência Artificial

**Nomes:** Arthur Sudbrack Ibarra, Felipe Grozse Nipper, 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 "quer" 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> <NOME_DOS_ARQUIVOS> |
```

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

NOME\_DOS\_ARQUIVOS = Nome dos arquivos .arff que serão gerados (treino e teste).

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, todas as palavras do dataset são incluídas, mesmo aquelas que já estavam presentes no arquivo de treino.

## Exemplo de execução do algoritmo:

```
PS C:\Users\ibarrart\Documents\Faculdade\TF-Inteligencia-Artificial> python generator.py 40 meu-arquivo-40-palavras

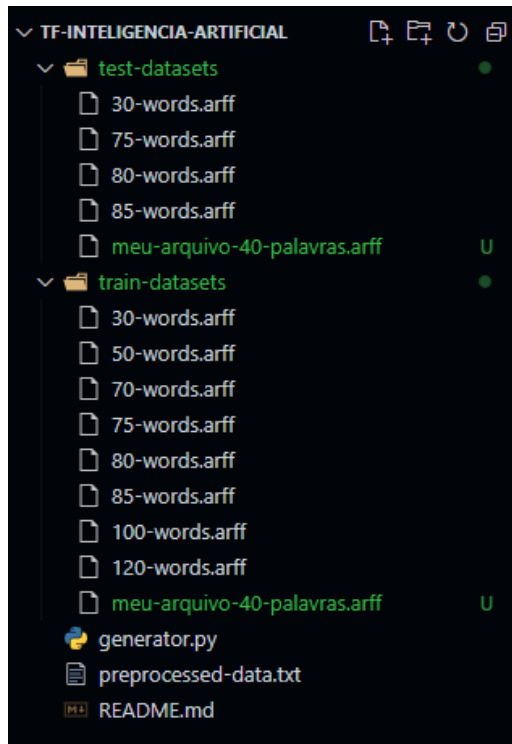
Palavras sendo desconsideradas: ['dell', 'notebook', 'not', 'nov', 'quer', 'q', 'ter', 'ach', 'agon', 'vai', 'dess', 'inspiron', 'precis', 'cas', '%', 's', 'tou']

Arquivos gerados com sucesso:

train-datasets/meu-arquivo-40-palavras.arff: Arquivo de treino, não considera linhas totalmente zeradas.
test-datasets/meu-arquivo-40-palavras.arff: Arquivo de teste, considera todo o dataset.

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

## Arquivos .arff gerados:





## 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:** 80.

**Palavras sendo desconsideradas:** dell, notebook, not, nov, quer, q, ter, ach, agor, vai, dess, inspiron, precis, cas, %, 's, 'tou.

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

**distanceWeighting:** Weight by 1/distance.

**Vizinhos próximos sendo considerados:** 5.

## Resultado:

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'Ibk - K 5 - W 0 - I - A "weka.core.neighboursearch.LinearNNSearch - A \"weka.core.EuclideanDistance - R first-last\"'". The test options are set to 'Supplied test set' with a 'Set...' button. The classifier output is displayed in the right pane, showing the model and evaluation results.

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

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

IBk instance-based classifier  
using 5 inverse-distance-weighted 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: 0.14 seconds

**=== Summary ===**

Correctly Classified Instances	517	85.596 %
Incorrectly Classified Instances	87	14.404 %
Kappa statistic	0.6356	
Mean absolute error	0.1926	
Root mean squared error	0.2966	
Relative absolute error	44.8016 %	
Root relative squared error	63.1651 %	
Total Number of Instances	604	

**=== Detailed Accuracy By Class ===**

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,988	0,416	0,831	0,988	0,902	0,671	0,942	0,965	-1
	0,584	0,012	0,958	0,584	0,726	0,671	0,942	0,873	1
Weighted Avg.	0,856	0,284	0,872	0,856	0,845	0,671	0,942	0,935	

**=== Confusion Matrix ===**

a	b	<-- classified as
402	5	a = -1
82	115	b = 1

Conforme pode ser observado na figura acima, o modelo acertou 85,596% das classificações, errando somente 14,404% dos palpites. O grupo notou que um dos fatores mais importantes para alcançar tal resultado foi a configuração **distanceWeighting**, que por padrão vem configurada como **No distance weighting**. Quando o campo foi mudado para **Weight by 1/distance**, os acertos do algoritmo aumentaram consideravelmente. Além disso, é de relevância comentar que para o caso em questão, diversos valores de números de vizinhos a se considerar (1, 2, 5...) foram testados, porém a taxa de acerto sempre manteve-se em 85,596%.

## 2.2. MultilayerPerceptron

**Quantidade de palavras no bag of words:** 80.

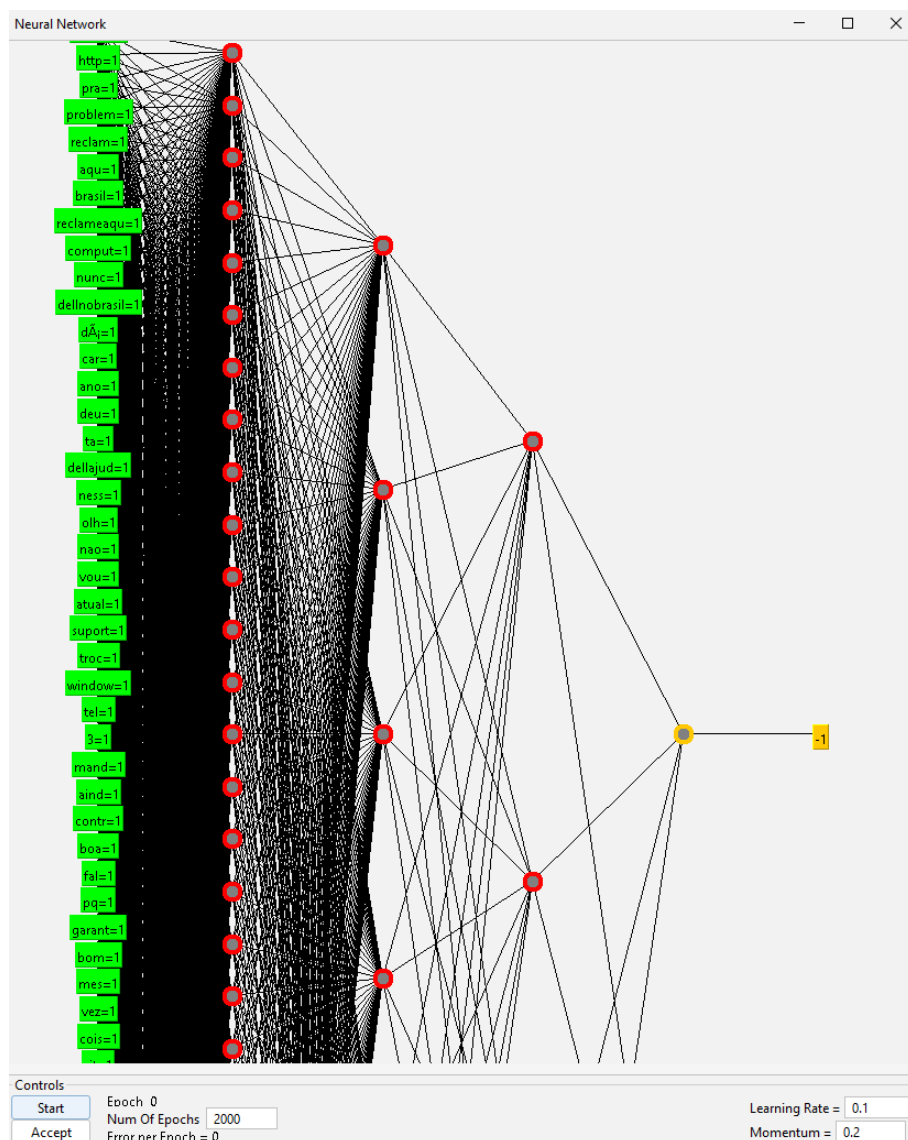
**Palavras sendo desconsideradas:** dell, notebook, not, nov, quer, q, ter, ach, agor, vai, dess, inspiron, precis, cas, %, 's, 'tou.

**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 2000 -V 0 -S 0 -E 20 -H "a, 8, 4" -R**

Test options

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

More options...

(Nom) class

Start Stop

Result list (right-click for options)

- 16:39:36 - lazy.JBk
- 16:50:40 - lazy.JBk
- 16:50:50 - lazy.JBk
- 16:51:02 - lazy.JBk
- 16:51:09 - lazy.JBk
- 16:52:07 - lazy.JBk
- 16:52:19 - lazy.JBk
- 16:53:34 - lazy.JBk
- 16:53:54 - lazy.JBk
- 16:54:00 - lazy.JBk
- 16:54:06 - lazy.JBk
- 16:54:18 - lazy.JBk
- 17:00:40 - functions.MultilayerPerceptron
- 17:03:48 - functions.MultilayerPerceptron
- 17:05:08 - functions.MultilayerPerceptron
- 17:12:18 - functions.MultilayerPerceptron
- 17:14:55 - functions.MultilayerPerceptron
- 17:16:58 - functions.MultilayerPerceptron
- 17:18:52 - functions.MultilayerPerceptron
- 17:20:39 - functions.MultilayerPerceptron**

Classifier output

Node 50 -1.7172862207809825

Sigmoid Node 54

Inputs Weights

Threshold 1.864675393785267

Node 43 -2.117716818928248

Node 44 -2.120559207626934

Node 45 -2.1111945053001917

Node 46 -2.0404649420213956

Node 47 -2.0149747143514705

Node 48 -1.716375098062046

Node 49 -2.523833727350775

Node 50 -1.4194547630492418

Class -1

Input

Node 0

Class 1

Input

Node 1

Time taken to build model: 66.91 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.02 seconds

=== Summary ===

Correctly Classified Instances	506	83.7748 %
Incorrectly Classified Instances	98	16.2252 %
Kappa statistic	0.5924	
Mean absolute error	0.2145	
Root mean squared error	0.3438	
Relative absolute error	49.9028 %	
Root relative squared error	73.2144 %	
Total Number of Instances	604	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,968	0,431	0,823	0,968	0,889	0,621	0,893	0,928	-1
	0,569	0,032	0,896	0,569	0,696	0,621	0,893	0,769	1
Weighted Avg.	0,838	0,301	0,847	0,838	0,826	0,621	0,893	0,876	

=== Confusion Matrix ===

a	b	-- classified as
394	13	a = -1
85	112	b = 1

Conforme mostrado na imagem acima, o modelo de Multilayer Perceptron obteve uma taxa de acerto de 83,774% e uma taxa de erro de 16,225%, 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 configurações diferentes foram testadas **(4,2)**, **(8,4,2)** e **(8,8,4)**, a taxa de acerto ficou perto de 67,5%, com ~15% de diferença em relação ao resultado final obtido com a configuração **(8,4)**.

## 2.3. RandomForest

Quantidade de palavras no bag of words: 80.

Palavras sendo desconsideradas: dell, notebook, not, nov, quer, q, ter, ach, agor, vai, dess, inspiron, precis, cas, %, 's, 'tou.

Tipo do teste: Conjunto de teste fornecido.

Profundidade máxima: 0.

Resultado:

The screenshot shows the Weka Explorer interface with the RandomForest classifier selected. The 'Test options' section shows 'Supplied test set' is chosen. The 'Classifier output' section displays the following results:

```
0.09 ( 146) volt
0.09 ( 137) ness
0.09 ( 109) carreg
0.09 ( 75) tent
0.09 ( 93) l
0.09 ( 176) sit
0.08 ( 134) 10
0.08 ( 164) dã;
0.08 ( 181) coils
0.07 ( 146) bom
0.07 ( 124) dand
0.07 ( 144) obrig
0.06 ( 135) vend
0.06 ( 153) ta
0.06 ( 102) err
0.06 ( 130) reclameagu
0.04 ( 84) comput
0.02 ( 14) contr
```

Time taken to build model: 0.26 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.15 seconds

=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	517	85.596 %
Incorrectly Classified Instances	87	14.404 %
Kappa statistic	0.6387	
Mean absolute error	0.229	
Root mean squared error	0.3129	
Relative absolute error	53.2608 %	
Root relative squared error	66.6354 %	
Total Number of Instances	604	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
Weighted Avg.	0,980	0,401	0,835	0,980	0,902	0,668	0,937	0,961	-1
	0,599	0,020	0,937	0,599	0,731	0,668	0,937	0,868	1

=== Confusion Matrix ===

a	b	-- classified as
399	8	a = -1
79	118	b = 1

O último modelo testado foi o de RandomTree, o qual obteve uma taxa de acerto de 85,596% e uma taxa de erro de 14,404%, empatando com o modelo de KNN e superando o modelo de Multilayer Perceptron. A configuração observada pelo grupo como a mais importante para o sucesso do algoritmo foi a de **profundidade máxima**, que conforme tinha o seu valor incrementado, a acurácia do modelo



diminuía, porém quando a profundidade alcançava um determinado valor X, a acurácia do algoritmo voltava a aumentar até chegar em 85,595% novamente.

### **3. Conclusão:**

Ao final dos testes realizados, foi possível perceber que as taxas de acerto dos três modelos testados foram razoáveis, porém o grupo acredita que não são taxas altas o suficiente para considerar os modelos confiáveis. Isso se dá, muito provavelmente, pelo dataset que foi utilizado para gerar os arquivos de treino, visto que ele era pequeno (apenas 604 instâncias), 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.