Trabalho de Inteligência Artificial

Marlon Baptista de Quadros[1], Eurico Saldanha Lemos Neto[1]

[1]Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande Do Sul (PUCRS)

{marlon.quadros, eurico.neto}@acad.pucrs.br

Resumo. *Este relatório busca mostrar os resultados e técnicas de aprendizado de máquina utilizando o software Weka em cima de um corpus de notícias contendo dados não estruturados e separados por conteúdo. Também será detalhado as técnicas utilizadas para a criação de uma estrutura de arquivos e mineração de textos utilizando algumas técnicas simples de processamento de linguagem natural.*

1. Introdução

Os dados utilizados para o processo de aprendizado de máquina são notícias do ano de 2010 do jornal Diário Gaúcho e está separado por categorias, que são: Polícia, Esportes, Seu Problema é Nosso e Espaço do Trabalhador. Os textos se encontram em arquivos *.txt* sem nenhuma estrutura, como por exemplo: *XML*, *HTML* ou *JSON*. A primeira etapa para que se possa trabalhar, facilitar e melhorar os resultados obtidos no processo de aprendizado e máquina é estruturar os documentos e criar os arquivos *.arff* para que o Weka consiga trabalhar de forma eficiente.

Neste processo foi utilizado a linguagem de programação *Python,* a biblioteca NLTK (Natural Language Toolkit), o Cogroo (Corretor Gramatical para OpenlLibre Office), uma biblioteca que permite a integração do Cogroo com o Python e a biblioteca BeautifulSoup para remover marcações *HTML.*

O primeiro passo do algoritmo é separar os textos de cada arquivo/categoria e coloca-los em um dicionário (*dict* Python). Depois de separados os textos, são removidos elementos que não são necessários no processo de aprendizagem de máquina, como tags HTMLs, e-mails e realizado o processo de lemmatização dos verbos com o auxílio do Cogroo.

Com os textos "limpos" é realizado o processo de anotação das palavras através da técnica de Part of Speech (POS), para essa classificação foi utilizado o conjunto de dados da língua portuguesa que podem ser baixados pelo NLTK, aqui foi utilizado um código auxiliar o Nltk-Tagger-Portuguese que permite a criação de um tagger melhorado com base nos dados do NLTK.

Com POS inserido nos textos é feito a separação dos *n-grams,* a biblioteca NLTK possui um método que chamado *ngrams* que permite receber uma lista de palavras e um número *n,* indicando a quantidade de separações de termos que se deseja realizar. Antes da criação dos *n-grams* só são deixados os verbos, adjetivos, advérbios e substantivos, no caso dos *unigrams,* já para os *bigrams* e os *trigrams* foram incluídas as preposições.

Todos os textos estão separados e estruturados em um dicionário do Python, com

isso é realizado a separação dos textos de teste e os textos de treino, sendo 80% treino e 20% teste. Por fim, nessa etapa de estruturação são criados às Bag of Words (BoW) contendo a palavra ou termo e a quantidade de ocorências nos textos, sendo separados em vários arquivos separados por categorias ou misturados, com casos de unigrams, ou unigrams e bigrams, ou unigrams, bigrams e trigrams.

Com os ngrams é possível a realização do último algoritmo que cria os arquivos .arff que podem ser utilizados pelo Weka. Esse algoritmo lê os arquivos estruturados gerados pelas etapas anteriores e com as BoW ele verifica se existe a presença da palavra/termo no texto e vai criando uma estrutura de atributos com as palavras e de ocorrência ou não em determinado texto da categoria.

2. Resultados

Com os dados pré-processados e com os arquivos .arff sendo eles separados em casos de treinos e casos de testes, utilizamos a ferramenta Weka para o processo de aprendizado de máquina com um série de algoritmos, os resultados obtidos serão analisados abaixo.

Por conta de que nem todos os termos presentes no arquivo de treino estavam presentes no modelo de teste, o Weka sugeriu a utilização do algoritmo *InputMappedClassifier,* que segundo a sua documentação é utilizada nos casos em que existem inconsistência de atributos entre o modelo de treino e o de teste:

> *"Wrapper classifier that addresses incompatible training and test data by building a mapping between the training data that a classifier has been built with and the incoming test instances' structure. Model attributes that are not found in the incoming instances receive missing values, so do incoming nominal attribute values that the classifier has not seen before. A new classifier can be trained or an existing one loaded from a file."* [Class InputMappedClassifier: http://weka.sourceforge.net/doc.dev/weka/classifiers/misc/InputMappedClassifier.html - Acessado em: 10:42 - 27/11/2017]

2.1. Multilayer Perceptron

Utilizando o algoritmo de Multilayer Perceptron em um conjunto K=50 de termos unigrams da Bag of Words e usando a configuração padrão do Weka dos valores de *learningRate* em 0.3 e *momentum* 0.2 chegamos aos seguintes valores:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.34 seconds

=== Summary ===

Correctly Classified Instances          61               89.7059 %
Incorrectly Classified Instances         7               10.2941 %
Kappa statistic                          0.8617
Mean absolute error                      0.2181
Root mean squared error                  0.3115
Relative absolute error                 58.3282 %
Root relative squared error             72.0447 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,122    0,760      1,000    0,864      0,817    0,982     0,959     corpus_esportes
                 0,933    0,000    1,000      0,933    0,966      0,957    0,958     0,948     corpus_seu_problema_e_nosso
                 0,813    0,000    1,000      0,813    0,897      0,876    0,922     0,894     corpus_espaco_do_trabalhador
                 0,833    0,020    0,938      0,833    0,882      0,846    0,983     0,951     corpus_policia
Weighted Avg.    0,897    0,040    0,916      0,897    0,899      0,869    0,963     0,939

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 19  0  0  0 |  a = corpus_esportes
  0 14  0  1 |  b = corpus_seu_problema_e_nosso
  3  0 13  0 |  c = corpus_espaco_do_trabalhador
  3  0  0 15 |  d = corpus_policia
```

Imagem 1. Resultados obtidos do algoritmo Multilayer Perceptron. K=50 - Momentum=0.2 - LearningRate=0.3

Como pode-se observar nos resultados obtidos o percentual de instâncias classificadas corretamente foram de 89.7059% e incorretamente foram de 10.2941% e com o Kapp statistic chegou-se ao valor de 0.8617 onde o indice indica um valor substancial da qualidade do aprendizado.

No detalhamento da precisão por classe (Detailed Accuracy By Class) observa-se que as classes com maiores precisões foram o corpus_seu_problema_e_nosso e o corpus_espaco_do_trabalhador e o de menor precisão foi o corpus_esportes.

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances          61               89.7059 %
Incorrectly Classified Instances         7               10.2941 %
Kappa statistic                          0.8628
Mean absolute error                      0.0634
Root mean squared error                  0.2008
Relative absolute error                 16.9604 %
Root relative squared error             46.4312 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,875    0,058    0,824      0,875    0,848      0,801    0,980     0,913     corpus_espaco_do_trabalhador
                 1,000    0,038    0,882      1,000    0,938      0,921    0,997     0,992     corpus_seu_problema_e_nosso
                 0,944    0,020    0,944      0,944    0,944      0,924    0,992     0,980     corpus_policia
                 0,789    0,020    0,938      0,789    0,857      0,814    0,973     0,947     corpus_esportes
Weighted Avg.    0,897    0,033    0,900      0,897    0,896      0,864    0,985     0,958

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 14  1  0  1 |  a = corpus_espaco_do_trabalhador
  0 15  0  0 |  b = corpus_seu_problema_e_nosso
  1  0 17  0 |  c = corpus_policia
  2  1  1 15 |  d = corpus_esportes
```

Imagem 2. Resultados obtidos do algoritmo Multilayer Perceptron. K=50 unigram e bigram - Momentum=0.2 - LearningRate=0.3

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances          61               89.7059 %
Incorrectly Classified Instances         7               10.2941 %
Kappa statistic                          0.8627
Mean absolute error                      0.0726
Root mean squared error                  0.2134
Relative absolute error                 19.4219 %
Root relative squared error             49.353  %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,058    0,842      1,000    0,914      0,891    0,989     0,959     corpus_espaco_do_trabalhador
                 0,933    0,019    0,933      0,933    0,933      0,914    0,994     0,983     corpus_seu_problema_e_nosso
                 0,889    0,060    0,842      0,889    0,865      0,815    0,982     0,955     corpus_policia
                 0,789    0,000    1,000      0,789    0,882      0,854    0,971     0,944     corpus_esportes
Weighted Avg.    0,897    0,034    0,906      0,897    0,896      0,866    0,983     0,959

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 16  0  0  0 |  a = corpus_espaco_do_trabalhador
  0 14  1  0 |  b = corpus_seu_problema_e_nosso
  1  1 16  0 |  c = corpus_policia
  2  0  2 15 |  d = corpus_esportes
```

Imagem 3. Resultados obtidos do algoritmo Multilayer Perceptron. K=50 unigram, bigram e trigram - Momentum=0.2 - LearningRate=0.3

2.2. K-nn algoritmo LinearNNSearch

Com o algoritmo K-nn LinearNNSearch (IBk), com o conjunto K=50 de unigrams usando o parâmetro default de KNN=1 e o método de distância Euclidiana, obtivemos os seguintes resultados:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.11 seconds

=== Summary ===

Correctly Classified Instances          64               94.1176 %
Incorrectly Classified Instances         4                5.8824 %
Kappa statistic                          0.9213
Mean absolute error                      0.0378
Root mean squared error                  0.151
Relative absolute error                 10.1085 %
Root relative squared error             34.9294 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,947    0,041    0,900      0,947    0,923      0,893    0,982     0,965     corpus_esportes
                 0,933    0,000    1,000      0,933    0,966      0,957    0,982     0,957     corpus_seu_problema_e_nosso
                 0,938    0,019    0,938      0,938    0,938      0,918    0,993     0,961     corpus_espaco_do_trabalhador
                 0,944    0,020    0,944      0,944    0,944      0,924    0,993     0,967     corpus_policia
Weighted Avg.    0,941    0,021    0,943      0,941    0,941      0,921    0,988     0,963

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 18  0  1  0 |  a = corpus_esportes
  0 14  0  1 |  b = corpus_seu_problema_e_nosso
  1  0 15  0 |  c = corpus_espaco_do_trabalhador
  1  0  0 17 |  d = corpus_policia
```

Imagem 4. Resultados obtidos com LinearNNRegresion (IBk) para K=50 e KNN=1

O percentual de instâncias classificadas corretamente foi de 94.1176% e o de incorretamente classificadas foi de 5.8824%.

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.03 seconds

=== Summary ===

Correctly Classified Instances          62               91.1765 %
Incorrectly Classified Instances         6                8.8235 %
Kappa statistic                          0.8819
Mean absolute error                      0.0514
Root mean squared error                  0.1854
Relative absolute error                 13.7366 %
Root relative squared error             42.889  %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,875    0,038    0,875      0,875    0,875      0,837    0,960     0,922     corpus_espaco_do_trabalhador
                 0,933    0,000    1,000      0,933    0,966      0,957    1,000     1,000     corpus_seu_problema_e_nosso
                 1,000    0,040    0,900      1,000    0,947      0,930    0,997     0,982     corpus_policia
                 0,842    0,041    0,889      0,842    0,865      0,815    0,965     0,884     corpus_esportes
Weighted Avg.    0,912    0,031    0,913      0,912    0,911      0,882    0,980     0,944

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 14  0  0  2 |  a = corpus_espaco_do_trabalhador
  1 14  0  0 |  b = corpus_seu_problema_e_nosso
  0  0 18  0 |  c = corpus_policia
  1  0  2 16 |  d = corpus esportes
```

Imagem 5. Resultados obtidos com LinearNNRegresion (IBk) para K=50 unigrams e bigrams, com KNN=1

```
=== Evaluation on test set ===

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

=== Summary ===

Correctly Classified Instances          61               89.7059 %
Incorrectly Classified Instances         7               10.2941 %
Kappa statistic                          0.8623
Mean absolute error                      0.0588
Root mean squared error                  0.206
Relative absolute error                 15.7203 %
Root relative squared error             47.6517 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,875    0,019    0,933      0,875    0,903      0,875    0,978     0,933     corpus_espaco_do_trabalhador
                 0,933    0,019    0,933      0,933    0,933      0,914    0,986     0,962     corpus_seu_problema_e_nosso
                 1,000    0,060    0,857      1,000    0,923      0,898    0,991     0,956     corpus_policia
                 0,789    0,041    0,882      0,789    0,833      0,776    0,968     0,914     corpus_esportes
Weighted Avg.    0,897    0,036    0,899      0,897    0,896      0,862    0,981     0,940

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 14  0  0  2 |  a = corpus_espaco_do_trabalhador
  0 14  1  0 |  b = corpus_seu_problema_e_nosso
  0  0 18  0 |  c = corpus_policia
  1  1  2 15 |  d = corpus_esportes
```

Imagem 6. Resultados obtidos com LinearNNRegresion (IBk) para K=50 unigrams, bigrams e trigrams, com KNN=1

2.3. NaiveBayes

Com o algoritmo NaiveBayes e um conjunto de K=50 unigrams, obtivemos os seguintes resultados:

```
Time taken to build model: 0.02 seconds

=== Evaluation on test set ===

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

=== Summary ===

Correctly Classified Instances          63               92.6471 %
Incorrectly Classified Instances         5                7.3529 %
Kappa statistic                          0.9014
Mean absolute error                      0.0439
Root mean squared error                  0.1824
Relative absolute error                 11.7519 %
Root relative squared error             42.1815 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,947    0,082    0,818      0,947    0,878      0,830    0,986     0,969     corpus_esportes
                 1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     corpus_seu_problema_e_nosso
                 0,813    0,000    1,000      0,813    0,897      0,876    0,989     0,970     corpus_espaco_do_trabalhador
                 0,944    0,020    0,944      0,944    0,944      0,924    0,991     0,980     corpus_policia
Weighted Avg.    0,926    0,028    0,934      0,926    0,927      0,904    0,991     0,979

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 18  0  0  1 |  a = corpus_esportes
  0 15  0  0 |  b = corpus_seu_problema_e_nosso
  3  0 13  0 |  c = corpus_espaco_do_trabalhador
  1  0  0 17 |  d = corpus_policia
```

Imagem 7. Resultados obtidos com NaiveBayes para um conjunto K=50 unigrams

Com o algoritmo NaiveBayes chegou-se a 92.6471% instâncias corretamente classificadas e 7.3529% instâncias incorretamente classificadas.

```
Time taken to build model: 0.01 seconds

=== Evaluation on test set ===

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

=== Summary ===

Correctly Classified Instances          61               89.7059 %
Incorrectly Classified Instances         7               10.2941 %
Kappa statistic                          0.8621
Mean absolute error                      0.0603
Root mean squared error                  0.2236
Relative absolute error                 16.1283 %
Root relative squared error             51.7142 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,750    0,000    1,000      0,750    0,857      0,835    0,971     0,928     corpus_espaco_do_trabalhador
                 1,000    0,019    0,938      1,000    0,968      0,959    0,996     0,987     corpus_seu_problema_e_nosso
                 0,944    0,040    0,895      0,944    0,919      0,889    0,991     0,980     corpus_policia
                 0,895    0,082    0,810      0,895    0,850      0,790    0,982     0,960     corpus_esportes
Weighted Avg.    0,897    0,038    0,905      0,897    0,896      0,864    0,985     0,964

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 12  1  0  3 |  a = corpus_espaco_do_trabalhador
  0 15  0  0 |  b = corpus_seu_problema_e_nosso
  0  0 17  1 |  c = corpus_policia
  0  0  2 17 |  d = corpus_esportes
```

Imagem 8. Resultados obtidos com NaiveBayes para um conjunto K=50 unigrams e bigrams

```
Time taken to build model: 0.01 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances          61               89.7059 %
Incorrectly Classified Instances         7               10.2941 %
Kappa statistic                          0.8621
Mean absolute error                      0.06
Root mean squared error                  0.2227
Relative absolute error                 16.0537 %
Root relative squared error             51.5071 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,750    0,000    1,000      0,750    0,857      0,835    0,972     0,931     corpus_espaco_do_trabalhador
                 1,000    0,019    0,938      1,000    0,968      0,959    0,996     0,987     corpus_seu_problema_e_nosso
                 0,944    0,040    0,895      0,944    0,919      0,889    0,991     0,980     corpus_policia
                 0,895    0,082    0,810      0,895    0,850      0,790    0,982     0,960     corpus_esportes
Weighted Avg.    0,897    0,038    0,905      0,897    0,896      0,864    0,985     0,964

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 12  1  0  3 |  a = corpus_espaco_do_trabalhador
  0 15  0  0 |  b = corpus_seu_problema_e_nosso
  0  0 17  1 |  c = corpus_policia
  0  0  2 17 |  d = corpus_esportes
```

Imagem 9. Resultados obtidos com NaiveBayes para um conjunto K=50 unigrams, bigrams e trigrams

3. Conclusão

A etapa que mais deu trabalho foi a criação de estruturação dos textos, o pré-processamento, o algoritmo leva um tempo considerável para realizar todas as etapas do pré-processamento, desde a separação de cada texto até o procedimento de criação dos ngrams. Após a realização do pré-processamento são criados os arquivos .arff que são utilizados pelo Weka.

Com base nos resultados obtidos, o algoritmo que mais obteve instâncias classificadas corretamente foi o LinearNNSearch com percentuais entre 94.1176% ~ 89.7059% e manteve uma taxa elevada de Kappa Statistic variando entre 0.9213 ~ 0.8623.

Quando utilizado somente os unigrams, foi o que mais obteve taxa de instâncias classificadas corretamente e o Kappa Statistic elevado em todos os três algoritmos testados, quando inserido os bigrams os algoritmos apresentaram uma pequena queda na taxa de classificações corretas e o mesmo ocorreu com acréscimo dos trigrams.

Referências

Bird, S., Klein, E. and Loper, E. (2015) “Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit”, http://www.nltk.org/book/, Novembro/2017.

Weka, The University of Waikato “Documentation Weka”, http://weka.sourceforge.net/doc.dev/, Novembro/2017.

Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

Python, Foundation “Documentation Python 3.5”, https://docs.python.org/3.5/reference/index.html, Novembro/2017.