Categorização de Texto

Bruno Klein; Carlos Pakulski; Lorenzo Kniss; Wagner Schettert

December 19, 2017

1 Objetivos do trabalho

O presente trabalho procura experimentar técnicas de classificação em um corpus de notícias da Língua Portuguesa. Faz parte do escopo do trabalho processar os textos, classificá-los e analisar os resultados obtidos.

2 Pré-processamento

O pré-processamento é tido como o processo mais custoso e demorado de quase qualquer Aprendizado de Máquina. E tal tarefa ainda torna-se mais custosa em face a tarefas de processamento de texto, eis que os textos costumam ser desordenados e, por assim dizer, caóticos aos olhos de uma máquina. Os métodos citados nesta seção serão explicados na próxima seção.

Primeiramente, foram adquiridos os seguintes arquivos do site da UFRGS[1]:

1. CORPUS DG ESPACO DO TRABALHADOR - final.txt;
2. CORPUS DG ESPORTES - final.txt;
3. CORPUS DG POLICIA - final.txt;
4. CORPUS DG SEU PROBLEMA E NOSSO - final.txt.

Destes arquivos, utilizando o algoritmo *generate_testing_and_training_files()* ,foram separados 80% dos textos que seriam utilizados como dados de treinamento do categorizador e os outros 20% seriam utilizados como dados de teste. Então, foi criada uma rotina em *Python*, Figura 1 para que fossem retirados os caracteres especiais e as *tags* da linguagem *HTML*. Os comandos utilizados de "*re*", são comandos da biblioteca de expressões regulares do *Python*. Uma vez com

```
def clean_text(text):
    pattern1 = re.compile(r'[\x93\x94\x96\x97]')
    pattern2 = re.compile('<.*?>')

    regexes = [pattern1, pattern2]
    pattern_combined = '|'.join(x.pattern for x in regexes)

    clean_text = re.sub(pattern_combined, r'', text)

    return clean_text
```

Figure 1: Rotina que retira os caracteres especiais.

o texto editado, é necessário classificar as palavras por tipo gramatical. Para tanto, foi gerado um arquivo do tipo *JSON* que guarda o texto categorizado pela ferramenta *Cogroo*[2]. Isso é feito para todas as palavras de cada um dos Corpus. Além disso, foi necessário fazer a lematização

[1] *http : //www.ufrgs.br/textecc/porlexbras/porpopular/download_do_corpus.php*
[2] http://cogroo.sourceforge.net/

das palavras, processo que consiste em deflexionar uma palavra para determinar o seu lema, por exemplo, as palavras gato, gata, gatos, gatas são todas formas do mesmo lema: gato. A classificação gramatical e a lematização foram armazenadas em um arquivo JSON, gerado após a do método método *process_ text()*.

Com os *ngramas* já obtidos geramos a *Bag-of-Words* (BoW) com os termos gerais mais relevantes de todos os textos do Corpus localizados no diretório *original-files*. Obtendo a BoW, chama-se o método *generate_ arff_ files()* que gera os arquivos *Weka* (extensão .arff) de teste e treino para determinado *ngram* passado como parâmetro.

3 Algoritmos de Pré-processamento

Os algoritmos a seguir foram produzidos pelo próprio grupo, e são os responsáveis pelo pré-processamento dos textos do Corpus anteriormente adquirido. Algumas etapas do código demandam uma quantidade considerável de processamento, demorando alguns minutos para terminar a execução.

3.1 Generate_testing_and_training_files

Este método realiza a separação de todos os textos obtidos em dois arquivos distintos. Destes arquivos, 80% dos textos foram utilizados para a realização do treino do algoritmo e 20% para a realização dos testes. Ambos os arquivos foram armazenados em disco em diretórios distintos: */testing* e */training*.

3.1.1 Parâmetros

- **s_path**: Caminho do diretório do arquivo do corpus a ser lido.
- **d1_path**: Caminho do diretório do arquivo de treino a ser gerado (80% dos textos).
- **d2_path**: Caminho do diretório do arquivo de teste a ser gerado (20% dos textos).
- **f_encoding**: codificação dos arquivos do corpus.

3.2 Process_text

Algoritmo que possui como objetivo realizar a lematização e a classificação gramatical das palavras. Após sua execução, são gerados e armazenados em disco um total de 8 arquivos. Dois destes arquivos (um para treino e outro para teste, ambos em formato *JSON*) contêm as classes gramaticais de cada palavra lematizada, com as palavras separadas em suas respectivas seções (ex: TEXTO 1, TEXTO 2). Os demais arquivos gerados, são referentes aos *ngrams* (1, 2 e 3) dos arquivos de teste e treino. São armazenados nos diretórios: *ngrams-training s-training* e *ngrams-testing*.

3.2.1 Parâmetros

- **training_path**: Caminho do diretório onde está localizado o arquivo de treino.
- **testing_path**: Caminho do diretório onde está localizado o arquivo de teste.
- **category**: Especifica a categoria relacionada ao arquivo do corpus.
- **f_encoding**: Codificação do conjunto de caracteres dos arquivos de teste e treino.

3.3 Get_Bag_Of_Words

Este método realiza a leitura de todos os arquivos de treino com um determinado n-grama (ex: *ngrams-training/esporte-ngram1.json*, ngrams-training/esporte-ngram1.json) construindo uma lista geral com os *ngramas* mais comuns.

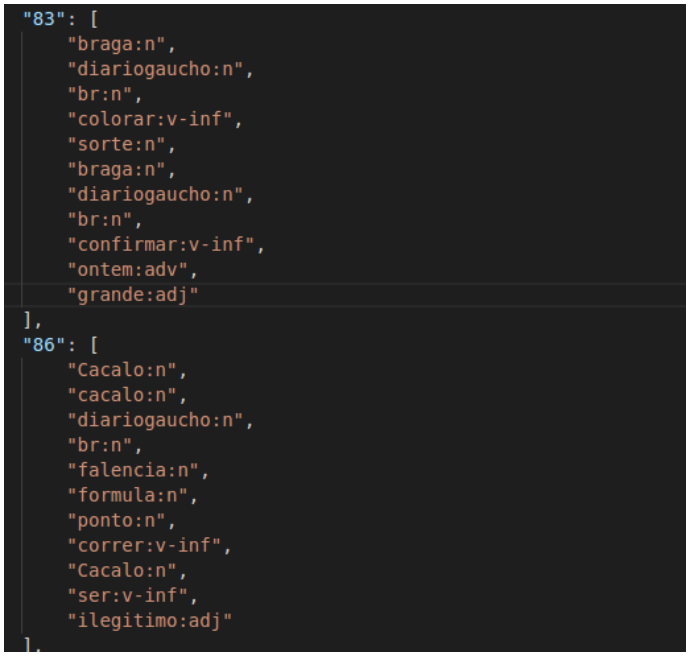


Figure 2: Arquivo JSON contendo as palavras lematizadas e suas classes gramaticais.

3.3.1 Parâmetros

- **ngrams_path**: Diretório onde se encontram todos os *ngramas*.
- **n**: Especifica se o *ngrama* é igual a 1, 2 ou 3.
- **bow_size**: Número de termos da lista geralmm

3.4 Generate_Arff_File

A rotina a seguir é executada essencialmente em duas iterações: a primeira que serve para gerar o arquivo de treinamento do *Weka*, e a segunda para criar o arquivo de testes. O trecho de código que acaba sendo repetido basicamente faz com que os arquivos do *Weka* sejam gerados com base nos arquivos de n-gramas localizados nos diretórios "ngrams_training" e "ngrams_testing".

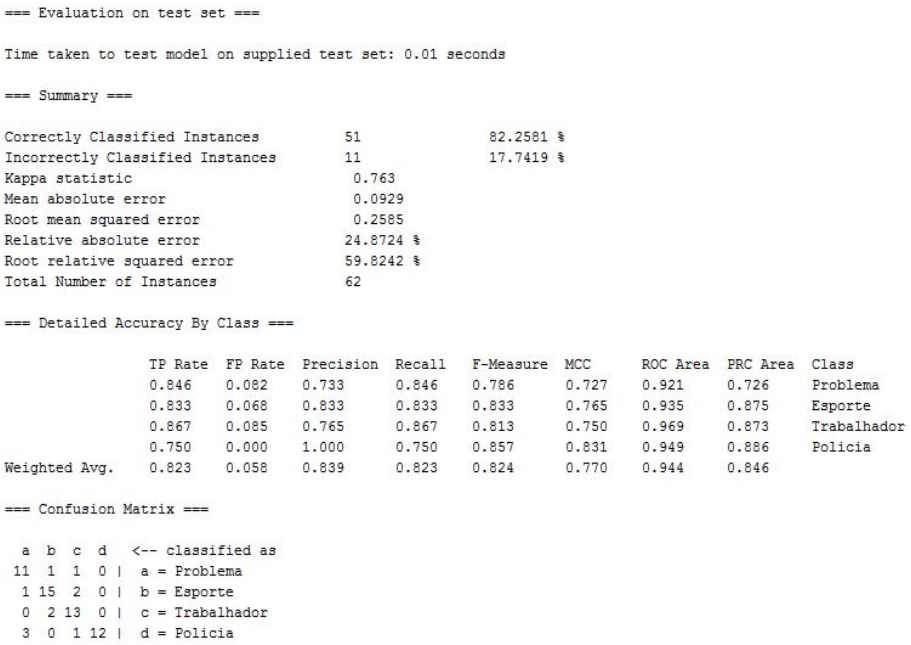
3.4.1 Parâmetros

- **n**: Determina o caminho do diretório onde está localizado o arquivo de treino.
- **filename_training**: Nome do arquivo de treinamento do *Weka*.
- **filename_testing**: Nome do arquivo de testes a ser processado pelo *Weka*.
- **f_encoding**: Codificação do conjunto de caracteres dos arquivos de teste e treino.
- **bow_size**: Tamanho da *bag-of-words*.

4 Algoritmos testados e análises

Para classificar os resultados obtidos pelos algoritmos acima, foi utilizada a ferramenta *Weka*[3]. Esta ferramenta possui uma coleção de algoritmos de aprendizagem de máquina. Dentro destes algoritmos, foi escolhido o algoritmo *K-nearest neighbour* (KNN), que é um algoritmo de classificação. Com isso, utilizamos o arquivo de treinamento para fazer o pré-processamento e o arquivo de testes para fazer a classificação. Os resultados obtidos pelo *Weka* podem ser vistos na Figura 3. É possível observar que o algoritmo KNN obteve um índice de 82.25% de sucesso ao classificar

[3]https://www.cs.waikato.ac.nz/ml/weka/



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

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

=== Summary ===

Correctly Classified Instances          51               82.2581 %
Incorrectly Classified Instances        11               17.7419 %
Kappa statistic                          0.763
Mean absolute error                      0.0929
Root mean squared error                  0.2585
Relative absolute error                 24.8724 %
Root relative squared error             59.8242 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.846    0.082    0.733      0.846    0.786      0.727    0.921     0.726     Problema
                 0.833    0.068    0.833      0.833    0.833      0.765    0.935     0.875     Esporte
                 0.867    0.085    0.765      0.867    0.813      0.750    0.969     0.873     Trabalhador
                 0.750    0.000    1.000      0.750    0.857      0.831    0.949     0.886     Policia
Weighted Avg.    0.823    0.058    0.839      0.823    0.824      0.770    0.944     0.846

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 11  1  1  0 |  a = Problema
  1 15  2  0 |  b = Esporte
  0  2 13  0 |  c = Trabalhador
  3  0  1 12 |  d = Policia
```

Figure 3: Resultados obtidos no *Weka* através do algoritmo *KNN* para unigramas.

as palavras. Foram 51 palavras corretamente classificadas, de um total de 62. Cabe salientar que os erros encontrados são provenientes das palavras cuja frequência de utilização era alta em todos os Corpus. É visto que o número de resultados falso-positivo, para a categoria Polícia, é zero, pois não existiram casos onde as palavras das outras categorias foram erroneamente categorizadas como Polícia. Isto denota que este grupo possui um valor maior de palavras únicas em relação aos outros.

Além deste algoritmo, o grupo realizou os testes com mais dois algoritmos: *Multilayer Perceptron* e *KStar*.

O algoritmo *KStar* também se mostrou bastante preciso quando foram realizados testes com arquivos que contem unigramas. Das 62 palavras analisadas, este classificador obteve um total de 85.48% de acerto, como poder ser observado através da Figura 4.

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

KStar Beta Verion (0.1b).
Copyright (c) 1995-97 by Len Trigg (trigg@cs.waikato.ac.nz).
Java port to Weka by Abdelaziz Mahoui (am14@cs.waikato.ac.nz).

KStar options : -B 20 -M a

Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.19 seconds

=== Summary ===

Correctly Classified Instances          53               85.4839 %
Incorrectly Classified Instances         9               14.5161 %
Kappa statistic                          0.8064
Mean absolute error                      0.0962
Root mean squared error                  0.2408
Relative absolute error                 25.7656 %
Root relative squared error             55.7168 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.846    0.061    0.786      0.846    0.815      0.764    0.958     0.814     Problema
                 0.833    0.023    0.938      0.833    0.882      0.841    0.977     0.952     Esporte
                 1.000    0.106    0.750      1.000    0.857      0.819    0.974     0.920     Trabalhador
                 0.750    0.000    1.000      0.750    0.857      0.831    0.974     0.963     Policia
Weighted Avg.    0.855    0.045    0.876      0.855    0.856      0.817    0.972     0.918

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 11  1  1  0 |  a = Problema
  0 15  3  0 |  b = Esporte
  0  0 15  0 |  c = Trabalhador
  3  0  1 12 |  d = Policia
```

Figure 4: Resultados obtidos no *Weka* através do algoritmo *KStar* para unigramas.

Os testes realizados com os arquivos que possuem duas palavras para testes se mostraram piores do que os que contem apenas uma palavra. A taxa de acertos após a execução destes testes diminuiu aproximadamente 35%, totalizando 50% de acertos e 50% de erros. Porém, da mesma maneira que o algoritmo KNN, o KStar se mostrou mais eficiente na classificação de três palavras do que com duas, como pode ser observado nas Figuras 5 e 6.

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

KStar Beta Verion (0.1b).
Copyright (c) 1995-97 by Len Trigg (trigg@cs.waikato.ac.nz).
Java port to Weka by Abdelaziz Mahoui (am14@cs.waikato.ac.nz).

KStar options : -B 20 -M a

Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.17 seconds

=== Summary ===

Correctly Classified Instances          31               50      %
Incorrectly Classified Instances        31               50      %
Kappa statistic                          0.3182
Mean absolute error                      0.2608
Root mean squared error                  0.3844
Relative absolute error                 69.8567 %
Root relative squared error             88.9545 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.438    0.130    0.538      0.438    0.483      0.330    0.804     0.624     Policia
               0.667    0.455    0.375      0.667    0.480      0.193    0.723     0.625     Esporte
               0.333    0.106    0.500      0.333    0.400      0.264    0.780     0.520     Trabalhador
               0.538    0.000    1.000      0.538    0.700      0.693    1.000     1.000     Problema
Weighted Avg.  0.500    0.191    0.578      0.500    0.507      0.350    0.816     0.678

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
  7  7  2  0 |  a = Policia
  4 12  2  0 |  b = Esporte
  0 10  5  0 |  c = Trabalhador
  2  3  1  7 |  d = Problema
```

Figure 5: Resultados obtidos no *Weka* através do algoritmo *KStar* para bigramas.

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

KStar Beta Verion (0.1b).
Copyright (c) 1995-97 by Len Trigg (trigg@cs.waikato.ac.nz).
Java port to Weka by Abdelaziz Mahoui (am14@cs.waikato.ac.nz).

KStar options : -B 20 -M a

Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.16 seconds

=== Summary ===

Correctly Classified Instances          35               56.4516 %
Incorrectly Classified Instances        27               43.5484 %
Kappa statistic                          0.4062
Mean absolute error                      0.2491
Root mean squared error                  0.3436
Relative absolute error                 66.7026 %
Root relative squared error             79.5181 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.077    0.000    1.000      0.077    0.143      0.249    0.717     0.356     Problema
               1.000    0.587    0.372      1.000    0.542      0.392    0.725     0.411     Policia
               0.000    0.000    0.000      0.000    0.000      0.000    0.713     0.357     Trabalhador
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Esporte
Weighted Avg.  0.565    0.151    0.596      0.565    0.460      0.444    0.800     0.558

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
  1 12  0  0 |  a = Problema
  0 16  0  0 |  b = Policia
  0 15  0  0 |  c = Trabalhador
  0  0  0 18 |  d = Esporte
```

Figure 6: Resultados obtidos no *Weka* através do algoritmo *KStar* para trigramas.

Além disso, foram feitos testes do KNN utilizando as BoWs de bigramas e trigramas, como visto nas Figuras 7 e 8.

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

IB1 instance-based classifier
using 1 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: 0.02 seconds

=== Summary ===

Correctly Classified Instances          28               45.1613 %
Incorrectly Classified Instances        34               54.8387 %
Kappa statistic                          0.2488
Mean absolute error                      0.2647
Root mean squared error                  0.4381
Relative absolute error                 70.8859 %
Root relative squared error            101.3814 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.500    0.217    0.444      0.500    0.471      0.272    0.766     0.584     Policia
               0.667    0.455    0.375      0.667    0.480      0.193    0.689     0.599     Esporte
               0.333    0.085    0.556      0.333    0.417      0.302    0.777     0.478     Trabalhador
               0.231    0.000    1.000      0.231    0.375      0.438    0.881     0.689     Problema
Weighted Avg.  0.452    0.209    0.568      0.452    0.440      0.291    0.770     0.585

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
  8  6  2  0 |  a = Policia
  5 12  1  0 |  b = Esporte
  0 10  5  0 |  c = Trabalhador
  5  4  1  3 |  d = Problema
```

Figure 7: Resultados obtidos no *Weka* através do algoritmo *KNN* para bigramas.

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

IB1 instance-based classifier
using 1 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: 0.01 seconds

=== Summary ===

Correctly Classified Instances          35               56.4516 %
Incorrectly Classified Instances        27               43.5484 %
Kappa statistic                          0.4062
Mean absolute error                      0.2338
Root mean squared error                  0.3401
Relative absolute error                 62.6084 %
Root relative squared error             78.7018 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.077    0.000    1.000      0.077    0.143      0.249    0.717     0.356     Problema
               1.000    0.587    0.372      1.000    0.542      0.392    0.725     0.411     Policia
               0.000    0.000    0.000      0.000    0.000      0.000    0.713     0.357     Trabalhador
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Esporte
Weighted Avg.  0.565    0.151    0.596      0.565    0.460      0.444    0.800     0.558

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
  1 12  0  0 |  a = Problema
  0 16  0  0 |  b = Policia
  0 15  0  0 |  c = Trabalhador
  0  0  0 18 |  d = Esporte
```

Figure 8: Resultados obtidos no *Weka* através do algoritmo *KNN* para trigramas.

Ao comparar os resultados obtidos pelo *Weka*, é possível observar que a classificação das palavras piorou. Isto ocorre pois as combinações tornam a classificação mais complexa. Como os valores de k não são dados por uma heurística, o desempenho do *KNN* piora ao classificar *ngramas* maiores que um. Os resultados para bigramas duas vezes piores que unigramas. Já o resultado para a

classificação de trigramas foi levemente superior ao arquivo de teste que utiliza duas palavras. Isto é dado devido ao fato de os trigramas possuírem muitas combinações com sentido único para sua classificação, como, por exemplo, qualquer combinação que utilize a palavra "Cacalo" é classificada como Esporte.

O Algoritmo *Multilayer Perceptron* (MLP) é uma rede neural que possui mais de uma camada de neurônios em alimentação direta. Este tipo de rede é composto por camadas de neurônios ligadas entre si por sinapses com pesos e geralmente o seu aprendizado se dá através de retro-propagação do erro.

O MLP foi o algoritmo que obteve o maior índice de acerto após a execução dos arquivos de teste. Na execução dos unigramas, representada através da Figura 9, o índice foi de 90.32%, acertando um total de 56 de 62 palavras analisadas e também uma taxa muito baixa de falsos positivos.

```
Time taken to build model: 3.19 seconds

=== Evaluation on test set ===

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

=== Summary ===

Correctly Classified Instances          56               90.3226 %
Incorrectly Classified Instances         6                9.6774 %
Kappa statistic                          0.871
Mean absolute error                      0.0584
Root mean squared error                  0.1805
Relative absolute error                 15.6402 %
Root relative squared error             41.7679 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.923    0.061    0.800      0.923    0.857      0.819    0.995     0.986     Problema
               0.833    0.000    1.000      0.833    0.909      0.883    0.980     0.963     Esporte
               0.933    0.064    0.824      0.933    0.875      0.835    0.984     0.949     Trabalhador
               0.938    0.000    1.000      0.938    0.968      0.958    1.000     1.000     Policia
Weighted Avg.  0.903    0.028    0.915      0.903    0.905      0.877    0.989     0.974

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 12  0  1  0 |  a = Problema
  1 15  2  0 |  b = Esporte
  1  0 14  0 |  c = Trabalhador
  1  0  0 15 |  d = Policia
```

Figure 9: Resultados obtidos no *Weka* através do algoritmo *MLP* para unigramas.

Analisando os testes realizados com os arquivos que contem os bigramas, este algoritmo em relação ao KNN se mostrou superior no que tange a taxa de acertos da classificação. Ainda assim, o índice de acertos comparados aos testes do unigrama diminuiu muito, chegando em um valor de 56.45% de acerto, o que representa apenas 35 das 62 palavras analisadas, como visto na Figura 10. Por fim, o índice de acertos de classificação após a execução dos testes com trigramas, embora que pouco, diminuiu novamente, totalizando uma taxa de 54.83% de acertos.

```
Time taken to build model: 3.11 seconds

=== Evaluation on test set ===

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

=== Summary ===

Correctly Classified Instances          35               56.4516 %
Incorrectly Classified Instances        27               43.5484 %
Kappa statistic                          0.413
Mean absolute error                      0.1983
Root mean squared error                  0.3801
Relative absolute error                 53.1019 %
Root relative squared error             87.9654 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.500    0.087    0.667      0.500    0.571      0.457    0.840     0.732     Policia
               0.556    0.318    0.417      0.556    0.476      0.221    0.794     0.671     Esporte
               0.400    0.191    0.400      0.400    0.400      0.209    0.810     0.536     Trabalhador
               0.846    0.000    1.000      0.846    0.917      0.902    1.000     1.000     Problema
Weighted Avg.  0.565    0.161    0.599      0.565    0.575      0.422    0.853     0.723

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
  8  5  3  0 |  a = Policia
  3 10  5  0 |  b = Esporte
  0  9  6  0 |  c = Trabalhador
  1  0  1 11 |  d = Problema
```

Figure 10: Resultados obtidos no *Weka* através do algoritmo *MLP* para bigramas.

```
Time taken to build model: 3.12 seconds

=== Evaluation on test set ===

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

=== Summary ===

Correctly Classified Instances          34               54.8387 %
Incorrectly Classified Instances        28               45.1613 %
Kappa statistic                          0.3934
Mean absolute error                      0.2362
Root mean squared error                  0.3405
Relative absolute error                 63.2558 %
Root relative squared error             78.7966 %
Total Number of Instances               62

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.077    0.000    1.000      0.077    0.143      0.249    0.717     0.356     Problema
               0.000    0.000    0.000      0.000    0.000      0.000    0.725     0.411     Policia
               1.000    0.596    0.349      1.000    0.517      0.376    0.713     0.357     Trabalhador
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Esporte
Weighted Avg.  0.548    0.144    0.584      0.548    0.445      0.433    0.800     0.558

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
  1  0 12  0 |  a = Problema
  0  0 16  0 |  b = Policia
  0  0 15  0 |  c = Trabalhador
  0  0  0 18 |  d = Esporte
```

Figure 11: Resultados obtidos no *Weka* através do algoritmo *MLP* para trigramas.

5 Conclusão

A construção de um *software* baseado em aprendizado de máquina não constitui tarefa trivial. O grupo observou - como comumente dito pela literatura especializada no assunto - que realmente

o pré-processamento dos dados é o processo mais complexo. Após o mesmo ter sido concluído, a categorização dos textos, em si, não foi tão complicado quanto o procedimento inicial de se organizar e limpar os textos. Basicamente, após os textos terem sido lematizados e as palavras divididas nas *bag-of-words*, o restante do trabalho se resumiu em apenas escrever o arquivo .arff para o *Weka* e rodá-lo em seguida. Após a execução de todas as análises comparando diversos algoritmos, também foi possível visualizar com clareza que o algoritmo *MLP* foi o que obteve o maior índice de acertos nas classificações nas palavras, totalizando 90.32% de precisão. Neste algoritmo, a diferença entre a resposta desejada e a resposta observada na saída é denominada sinal de erro, e de acordo com esse erro, os parâmetros da rede são ajustados.