ANA PAULA DA SILVA LEMOS Nº USP 8122564
ENOLA J. MANGO Nº USP 6926132
WELLINGTON SILVA NºUSP 3465255

# RELATÓRIO - TRABALHO PRÁTICO

## INTRODUÇÃO

Segundo Rennie et al (2003), Naive Bayes é frequentemente utilizado como referencial para classificação de textos por ser rápido e de fácil implementação. Algumas aplicações dessa técnica, discutidas em aula, são: detecção de spam em e-mails, categorização de documentos, detecção de conteúdo sexual em páginas web, análise de sentimento em documentos, entre outros. A última aplicação mencionada é o tema proposto para esse trabalho. A análise de sentimento é uma importante ferramenta para distintas áreas da indústria (alimentícia, cinematográfica, têxtil etc) e de pesquisa(sociologia, psicologia, etc), pois capta a reação que os usuários possuem a respeito do seu produto e\ou serviço. Dada a dimensão da internet, é inviável analisar todos os relatos dos usuários manualmente. Por isso, o método de aprendizado de máquina é essencial para extração e identificação dos sentimentos transmitidos em um texto corrido. Possuindo esse conhecimento, abrem-se novas possibilidades, como por exemplo: análise de tendências -o que pode motivar novas estratégias dentro das empresas ou uma nova linha de pesquisa-, publicidade direcionada - foco no público alvo -, entre outros. Dessa maneira, percebemos o quão vantajoso é saber a opinião do público.

Para realizar a análise de sentimentos, utilizaremos o algoritmo de aprendizagem Naive Bayes e classificaremos cada tweet que compõe um documento em “sentimento positivo” ou “sentimento negativo”. O código foi desenvolvido em linguagem python e estruturado da seguinte maneira:

- NaiveBayesWordClassifier.py: separação dos conjuntos.
- humor-classifier.py: treinamento e classificador.

As seções a seguir compreendem a estrutura detalhada na especificação do trabalho prático.

Primeiro explicaremos como os dados foram preparados, posteriormente comentaremos como o classificador foi feito, em um terceiro momento o classificador será avaliado utilizando a abordagem holdout, na quarta etapa a avaliação será feita utilizando a abordagem crossvalidation, na quinta parte será usada a abordagem holdout com stop words e por fim, a conclusão.

## PRÉ-PROCESSAMENTO DOS DADOS

Os dados foram separados aleatoriamente em dois conjuntos: teste e treinamento. Para tanto, foram criados dois novos arquivos, o primeiro, “trainSetFile.csv”, é composto por dois terços dos dados que serão utilizados no treinamento, o restante foi armazenado em “testSetFile.csv”. A estratégia utilizada para divisão dos dados foi a aplicação de um método que desordenou o arquivo inicial. Possuindo os dados dispostos de modo aleatório, facilmente pudemos separar os primeiros dois terços do conjunto no arquivo de treinamento e o restante no arquivo de teste. Cabe ressaltar que o cabeçalho foi previamente retirado do arquivo do conjunto de dados. Sendo assim, para eventuais testes com arquivos de outra natureza, faz-se necessário sua adequação.

O código poderá ser executado via linha de comando da seguinte maneira: python nome_arquivo.py. Caso o sistema operacional utilizado seja OS X ou qualquer distribuição do Linux não será necessária nenhuma instalação prévia. Para Windows é necessário instalar o interpretador python antes da execução do código (encontrado em: https://www.python.org/downloads/windows/).

As palavras de uso comum e que geralmente não trazem muita relevância para um texto tais como conjunções, pronomes e artigos por exemplo são denominadas de stop words. Utilizamos uma base livre de stop-words (encontrada em: https://code.google.com/p/stop-words/) para que palavras da nossa base de tweets que estejam na lista de stop words sejam removidas em um dado momento da implementação.

## IMPLEMENTAÇÃO DO CLASSIFICADOR

Segundo Matos et al (2009), “classificadores bayesianos são classificadores estatísticos supervisionados, cuja função é predizer a probabilidade de um exemplo pertencer a uma determinada classe”. Os autores complementam a explicação mencionando que o termo “Naive” surge a partir da independência condicional das classes, ou seja, o efeito de um valor de atributo de uma dada classe é independente dos valores de outros atributos.

De acordo com Matos et al (2009), a função dos classificadores é predizer a qual classe pertence um documento (no nosso caso, um tweet) de acordo com a classe que possui maior probabilidade a posteriore condicionada ao documento. Em outras palavras, para cada classe existente, precisamos calcular a probabilidade dessa classe ocorrer, dado uma documento. Contudo, um documento, ou tweet, é formado por palavras, que como mencionado, são independentes. Essa independência de atributos é obtida através do produtório de cada atributo dado a classe c. Outro valor que precisamos saber é a probabilidade de cada classe ocorrer (nº de documentos que são classificados pela classe c dividido pelo número de documentos). Caso a probabilidade prévia não seja conhecida, ela deve ser retirada do cálculo.

Dessa maneira, ao final do treinamento, calculamos alguns dados que serão posteriormente utilizados para classificação de um exemplo de teste, são eles:

- |Vocabulário| = tamanho do vocabulário;
- P(pos), P(neg) = Probabilidades a priori para cada classe;
- ni = número total de palavras (incluindo repetições) em cada classe
- P(wi | ci) = termos de probabilidade condicional de ocorrência das palavras, dada uma classe.

Como esses dados são calculados antes do exemplo de teste, podemos considerá-los a priori do nosso classificador.

**DESEMPENHO DO CLASSIFICADOR COM HOLDOUT**

Utilizando a abordagem de particionamento holdout, sem remoção de stop words, obtivemos os seguintes valores (nas probabilidades condicionais utilizamos o exemplo de duas palavras):

- |Vocabulário| = 652795

- P(pos), P(neg) = 0.50083, 0.49917
- n(pos), n(neg) = 7009143, 7500390
- P(woodb|pos) = 0.0000002610
- P(woodb|neg) = 0.0000001227
- P(solouk|pos) = 0.0000002610
- P(solouk|neg) = 0.0000001227

Podemos destacar também um exemplo de matriz de confusão, matriz que confronta as classes obtidas na classificação com as classes esperadas. nela percebemos que na diagonal principal temos os acertos.

| | POS | NEG |
|---|---|---|
| POS | 0175383 | 0061538 |
| NEG | 0041928 | 0194739 |

Obtivemos aproximadamente 78.2% de acurácia, ou seja, os rótulos pré definidos foram os mesmos aplicados pelo classificador induzido em 78.2% das vezes em que os testes foram feitos. Matos et al (2009), nos fornece um método simples para o cálculo de desempenho do classificador:

$$erro(h) = \frac{1}{n}\sum_{i=1}^{n} \|y_i \neq h(x_i)\|$$

Dentro do somatório fazemos a comparação entre o rótulo verdadeiro do exemplo yi e o rótulo atribuído pelo classificador induzido, essa comparação devolve 0 se a condição for falsa ou 1 caso a condição seja verdadeira. A somatória dividida pelo número de exemplos testados nos dá o erro. E a precisão é calculada subtraindo esse resultado de um.

**DESEMPENHO DO CLASSIFICADOR COM 10-CROSS VALIDATION**

A segunda abordagem utilizada para o particionamento do conjunto de dados foi o cross validation. Segundo Matos et al (2009), essa abordagem divide os dados em partes iguais e os testa repetidas vezes, variando os dados de teste dentro do mesmo conjunto de dados. Seguindo essa regra, com o conjunto de dados previamente desordenado, pegamos o valor total de documentos e dividimos pelo número dado no enunciado para o particionamento que era igual a dez. O conjunto de teste foi separado ordenadamente para cada iteração correspondente. Ou seja, na primeira iteração o primeiro fold foi o de teste e o restante compôs o conjunto de treinamento, e assim sucessivamente.

De acordo com os autores acima mencionados, a vantagem do cross-validation é a divisão dos dados, pois cada fold acaba sendo testado exatamente uma vez para o conjunto de treinamento. A desvantagem é que o algorítmo de treinamento tem que repetir k (número de folds) vezes, o que significa um custo computacional elevado. A seguir será mostrada a porcentagem de acertos para cada uma das iterações:

| ITERAÇÃO | TAXA DE ACERTOS (em %) |
|---|---|
| 1 | 78.32 |
| 2 | 78.41 |
| 3 | 78.25 |
| 4 | 78.57 |
| 5 | 78.60 |
| 6 | 78.25 |
| 7 | 78.21 |
| 8 | 78.37 |
| 9 | 78.27 |
| 10 | 78.29 |

A estimativa final para o desempenho do modelo foi representado pela média de erros dos 10 folds que ficou em 78.35% com desvio padrão de 0.13. Matos et al (2009), afirmam que a

variância é reduzida à medida que k é aumentado. Posto isso, podemos utilizar o desvio padrão como segunda métrica para analisar o desempenho do classificador utilizando a abordagem cross validation, pois representa o grau de dispersão em relação à média.

## DESEMPENHO DO CLASSIFICADOR COM HOLDOUT E STOP WORDS

As stop words são palavras simples, de uso cotidiano, que geralmente não tem relevância no processo de classificação. O holdout aplicado nessa etapa foi constituído da mesma maneira como explicado anteriormente. A diferença é que as stop words foram retiradas do vocabulário, que passou de 652622 palavras para 651811 palavras. Esse valor também variava pois a cada execução do script as amostras de teste e treinamento era regeradas. Utilizando essa tática, o classificador obteve 74.3% de precisão, praticamente 4% a menos do que foi calculado no holdout com stop words. Dessa maneira, podemos concluir que as stop words tiverem relevância para uma classificação mais acertiva.

## CONCLUSÃO

Com a realização deste trabalho, conclui-se que o uso de técnicas probabilísticas é de grande valia ao processamento de linguagem natural. O resultado alcançado com o Naive Bayes, de 74% de acurácia (mínimo), é bastante relevante e pode ser usado como auxílio na análise sintática. Além disso, mostra que a técnica do Naive Bayes é de fato efetiva e que a hipótese de independência muitas vezes não influencia o resultado. Um fato interessante a ser observado é que houve uma diferença entre os métodos de particionamento de dados, contudo, essa discrepância é mínima para se escolher um método ou outro. Outro critério para realizar essa escolha é o tempo de processamento, que no cross validation é muito maior que nas outras abordagens. Portanto, a melhor escolha depende das necessidades de quem aplica o método.

## REFERÊNCIAS BIBLIOGRÁFICAS

MITCHEL, T. M. **Machine Learning**. McGraw-Hill International Editions,

Computer Science Series, 1997. 432 p.

RENNIE, J.D.M., et al. **Tackling the Poor Assumptions of Naive Bayes Text Classifiers.** Disponível em: < http://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf>. Acesso em: 24 nov. 2014.

**Machine Learning Blog & Software Development News.** Disponível em: < http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/>. Acesso em: 24 nov. 2014.

**Introduction to sentmental analysis.** Disponível em: <http://www.lct-master.org/files/MullenSentimentCourseSlides.pdf>. Acesso em: 24 nov. 2014.

MATOS, P.F .**Relatório técnico "Conceitos sobre aprendizado de máquina"**. Disponível em: <http://www.icmc.usp.br/pessoas/taspardo/techreportufscar2009b-matosetal.pdf>. Acesso em: 26 nov. 2014.

Brown Computer Science. **Twitter sentiment classification.** Disponível em: <http://cs.brown.edu/courses/csci1951-a/assignments/assignment3/>. Acesso em: 27 nov. 2014.

**Stop words**. Disponível em: <http://www.agenciamestre.com/seo/stop-words-como-funcionam-palavras-de-parada/>. Acesso em: 27 nov. 2014.