Classificação de textos: Processamento de linguagem natural e aprendizado de máquina

Arthur Aguiar Menezes de Souza, Emanoel Aurélio Vianna Fabiano, Mauricio Steinert

Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 - Partenon, RS, 90619-900 – Porto Alegre – RS – Brazil

{arthur.aguiar, emanoel.fabiano, mauricio.steinert}@acad.pucrs.br

Resumo

O presente relatório apresenta os resultados obtidos a partir de experimentos com processamento e classificação de textos utilizando aprendizado de máquina. Abordaremos a solução empregada para o pré-processamento dos dados, bem como os resultados obtidos com a execução dos algoritmos de aprendizado de máquina, utilizando como suporte a ferramenta Weka[1].

1. Introdução

O objetivo deste trabalho é, a partir de *corpus* de textos de classes distintas (“esportes”, “polícia”, “espaço do trabalhador” e “seu problema é nosso”) extraídos do Diário Gaúcho, treinar o computador para identificar, a partir do uso de algoritmos de aprendizagem de máquina, a qual classe determinado texto pertence.

O pré-processamento consiste, dado um conjunto de dados não estruturados, prepará-los para que possam ser utilizados para extração de informações. A partir dos dados do pré-processamento, foram gerados arquivos os quais servem como entrada para o Weka, a fim de realizar o processo de análise, aprendizado de padrões e classificação dos textos propriamente ditos.

2. Pré-processamento de dados

A etapa de pré-processamento é responsável pela leitura dos arquivos de textos (dados não estruturados) e sua preparação para processamento: retirada de informações indesejadas[2] , normalização morfológica e anotação linguística (identificação de classes gramaticais), bem como extração dos termos e seleção dos mais relevantes. A partir desta seleção, é gerada uma *Bag-of-Words*, composta pelos *k* termos mais relevantes do conjunto de textos de treino, sendo cada termo composto por n-gramas distintos (valores de 1 a 3).

O pré-processamento é realizado a partir de um programa Java, o qual recebe como entrada um conjunto de textos para cada uma das classes, separados em pastas de classes e finalidade (treino ou teste), conforme tabela 1. Como o *corpus* de texto estava organizado em

[1] http://www.cs.waikato.ac.nz/~ml/weka/

[2] No caso deste *corpus*, foram removidas tags HTML, as quais não são relevantes para o processo de aprendizado e de classificação.

um arquivo por classe, onde cada arquivo possui *n* textos, foi utilizado o programa csplit[3] para dividir o arquivo em um arquivos por texto.

textos/esporte/treino	textos/policia/treino	textos/trabalhador/treino	textos/problema/treino
textos/esporte/teste	textos/policia/teste	textos/trabalhador/teste	textos/problema/teste

Tabela 1: estrutura de pastas com textos (*corpus*)

O Cogroo é utilizado de forma integrada para processamento, normalização e classificação dos termos. Para cada texto é criado um objeto, o qual armazena sua classe, para qual finalidade o mesmo é empregado (treino, teste ou classificação), sua classe e listagem de termos, onde os termos já estão normalizados, classificados e contadas suas repetições. Para cada texto, é criada uma lista de seus termos, compostos de n-gramas, onde *n* varia de 1 a 3: quando o valor de *n* é igual a 1, são considerados apenas os termos classificados como verbos, substantivos, advérbios e adjetivos; para os demais casos, são consideradas também as preposições - termos como pontuação e conjunções são desconsiderados.

A partir destes dados, é criada uma lista com os *k* termos únicos e mais relevantes para cada número de n-gramas, onde é utilizado como critério de relevância a quantidade de repetições do termo dentro do conjunto de textos de treino. O arquivo resultante segue o formato ARFF[4], onde os atributos (tag @ATTRIBUTE) são as palavras e as classes, e as linhas de informações (tag @DATA) representa, para cada atributo, a presença (valor 1) ou não (valor 0) de cada atributo para cada texto fornecido como entrada. O resultado do pré-processamento são dois arquivos: um com os dados para treino e outro para testes.

3. Processo de aprendizagem de máquina

Para o processo de aprendizagem de máquina, utilizaremos os algoritmos de classificação k-NN (k-Nearest Neighbours), MultiLayer Perceptron e Redes Bayesianas. Para todos os testes, foram utilizadas 273 amostras para treino (80% do *corpus*) e 68 classificadas para teste (20%).

[3] https://www.gnu.org/software/coreutils/manual/html_node/csplit-invocation.html
[4] Attribute-Relation File Format (https://www.cs.waikato.ac.nz/ml/weka/arff.html)

3.1 Algoritmo k-NN

Para execução do algoritmo k-NN, foram realizados testes variando o valor de n-gramas, *k* (quantidade de palavras utilizadas para compor a Bag-of-Words), além do valor de *k-NN* (com quantos outros padrões um elemento é comparado para determinar sua classe). Os dados, presentes na forma de tabela no anexo 1, estão sintetizados na figura 1.

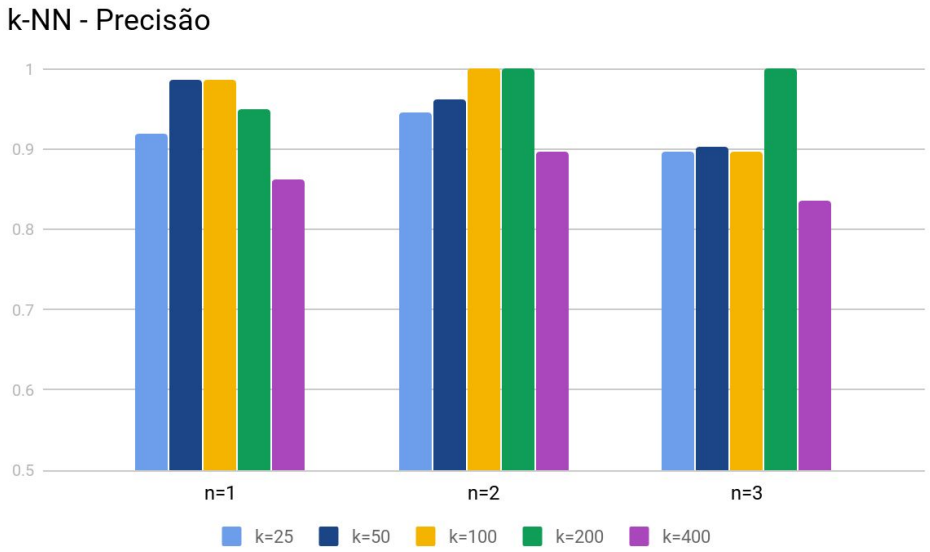

Figura 1:gráfico de precisão do algoritmo k-NN.

A partir do gráfico, é possível perceber que nem sempre o aumento do valor de *k* aumenta a precisão na classificação dos textos. Para todos os valores de *n*, existe uma perda de precisão quando o valor de *k* é igual ou superior a 200. Foram realizados experimentos alterando o valor de *k-NN*, o que demonstrou um certo ganho de precisão em certas circunstâncias.

Além disso, em alguns casos, o nível de precisão chegou a 1 (100%), não sendo possível avaliar se foi uma coincidência do conjunto de dados ou uma falha do software.

3.2 MultiLayer Perceptron

Para a execução do MultiLayer Perceptron foram mantidas as configurações do Weka. As variações de parâmetros foram realizadas nos valores de *k* e no número de *n-gramas*.

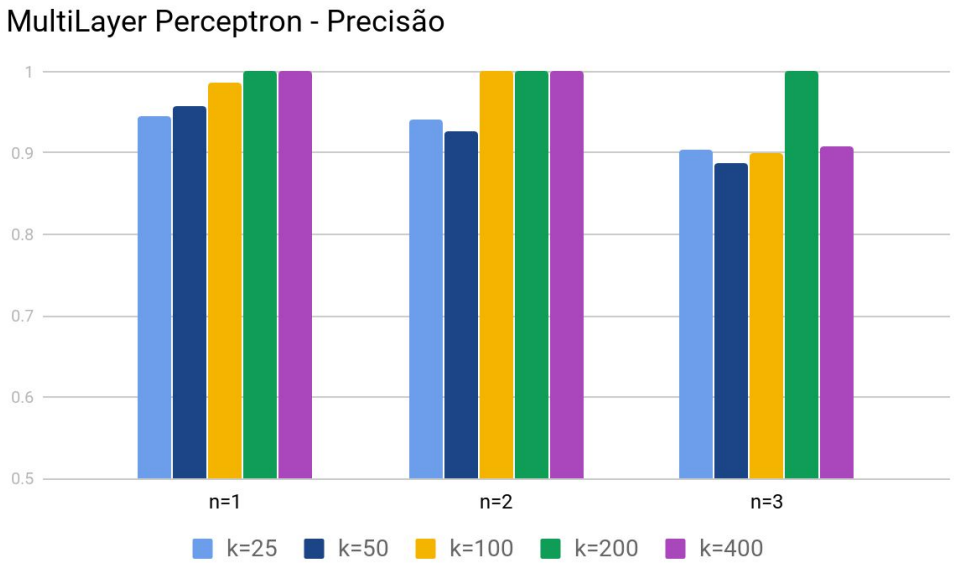

Figura 2:gráfico de precisão do algoritmo MultiLayer Perceptron.

Ao analisar o gráfico, o nível de precisão aumenta quando temos *n* igual a 1. Quando temos *n* igual a 2, temos um aumento de precisão com aumento de *k*, com resultado 1 (100%) para *k* igual a 100, 200 e 400. Para *n* igual a 3, percebemos um pico quando *k* é igual a 200, e uma queda quando *k* é igual a 400.

3.3 Redes Bayesianas

Para os testes com algoritmos de Redes Bayesianas, foram mantidas as configurações padrão adotadas pelo Weka. Novamente, as variações foram realizadas nos valores de *k* e no número de *n-gramas*.

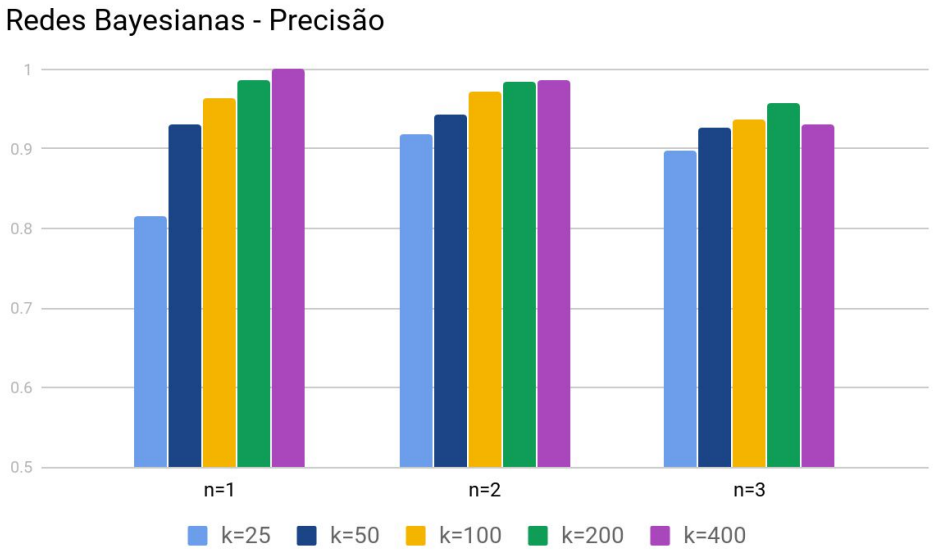

Figura 3:gráfico de precisão do algoritmo Redes Bayesianas.

Neste caso, com exceção de quando *n* é igual a 3, a precisão aumenta conforme o valor de *k* também aumenta.

Comparando os índices de precisão obtidos com os três algoritmos (figura 4), de maneira geral não é possível perceber a predominância de um algoritmo em relação ao outro enquanto variamos os valores de *n* e *k*:

- Algoritmos k-NN e MultiLayer Perceptron se alternam em maior precisão conforme o valor de *n* é igual a 1 e 2 e *k* varia.
- Aparentemente os algoritmos de Redes Bayesianas apresentam um nível de precisão igual ou maior quando *n* é igual a 3.

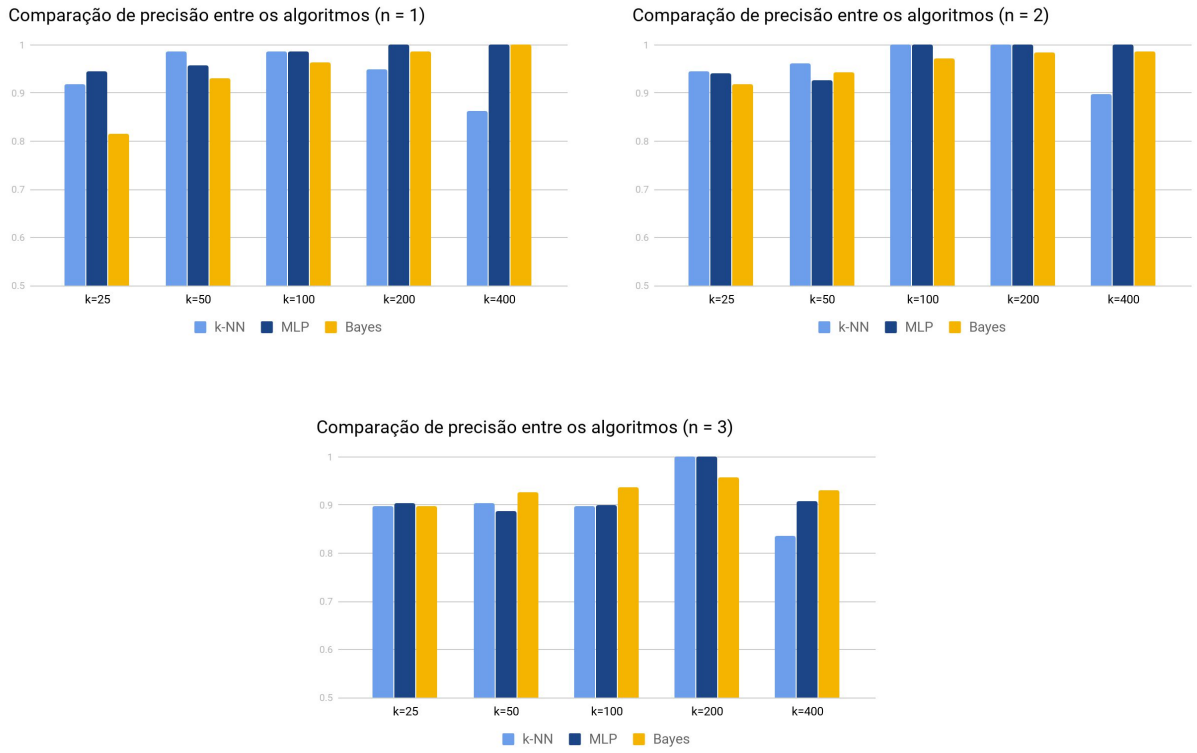

Figura 4: comparação da precisão entre os três algoritmos. Da esquerda para direita: *n* igual a 1, 2, *e* 3.

4. Considerações finais

A realização deste experimento permitiu uma compreensão dos procedimentos e dificuldades inerentes ao processamento e classificação de textos. Com relação ao programa de pré-processamento, em função do uso de listas simples para armazenamento dos termos, seria necessária a avaliação de estruturas de dados mais performáticas para inclusão e, principalmente, consulta de termos, a qual atualmente tem desempenho assintótico de *O(n)*.

Quanto à análise deste conjunto de dados, o qual está dentro de uma mesma linha editorial (mesmos autores, mesma linha de redação e linguagem institucional), constatamos que alguns algoritmos obtêm melhores resultados conforme o tamanho de n-gramas e o valor

de *k*. Não é possível afirmar, no entanto, que trata-se de uma tendência comportamental destes algoritmos, o que demandaria uma análise mais aprofundada do processo de aprendizado dos mesmos, assim como testes com outros conjuntos de dados distintos. Além disso, o Weka retornou alguns resultados com precisão 1 (100%), o que gera dúvidas quanto ao efetivo processo de aprendizado de máquina, até em função do pouco tempo disponível para explorar aspectos de depuração da ferramenta.

Referências

Corpus de textos. Disponível em
http://www.ufrgs.br/textecc/porlexbras/porpopular/download_do_corpus.php

ANEXO 1 - TABELAS COM RESULTADOS DE EXECUÇÃO DOS ALGORITMOS NO WEKA

k-NN

n-gramas	k	k-NN	Verdadeiro Positivo	Falso Positivo	Precisão	Recall	F-Measure
1	25	1	0,838	0,056	0,848	0,838	0,840
		5	0,897	0,035	0,898	0,897	0,897
		10	0,912	0,031	0,919	0,912	0,913
	50	1	0,956	0,016	0,962	0,956	0,956
		5	0,971	0,011	0,974	0,971	0,971
		10	0,985	0,005	0,986	0,985	0,985
	100	1	0,926	0,023	0,932	0,926	0,927
		5	0,985	0,005	0,986	0,985	0,985
		10	0,985	0,005	0,986	0,985	0,985
	200	1	0,926	0,023	0,930	0,926	0,926
		5	0,941	0,019	0,949	0,941	0,939
		10	0,912	0,029	0,926	0,912	0,910
	400	1	0,794	0,070	0,854	0,794	0,778
		5	0,809	0,064	0,862	0,809	0792
		10	0,779	0,072	0,852	0,779	0,765

n-gramas	k	k-NN	Verdadeiro Positivo	Falso Positivo	Precisão	Recall	F-Measure
2	25	1	0,941	0,022	0,945	0,941	0,942
		5	0,912	0,033	0,916	0,912	0,912
		10	0,926	0,027	0,929	0,926	0,927
	50	1	0,912	0,034	0,923	0,912	0,914
		5	0,926	0,029	0,942	0,926	0,928
		10	0,956	0,017	0,962	0,956	0,956
	100	1	1,000	0,000	1,000	1,000	1,000
		5	0,927	0,025	0,931	0,927	0,927
		10	0,908	0,031	0,918	0,908	0,909
	200	1	1,000	0,000	1,000	1,000	1,000
		5	0,886	0,040	0,899	0,886	0,883
		10	0,886	0,039	0,901	0,886	0,883
	400	1	0,824	0,060	0,850	0,824	0,816
		5	0,882	0,041	0,898	0,882	0,865
		10	0,853	0,050	0,882	0,853	0,824

n-gramas	k	k-NN	Verdadeiro Positivo	Falso Positivo	Precisão	Recall	F-Measure
3	25	1	0,838	0,063	0,898	0,838	0,842
		5	0,824	0,068	0,892	0,824	0,826
		10	0,794	0,079	0,860	0,794	0,794
	50	1	0,868	0,050	0,901	0,868	0,871
		5	0,853	0,057	0,904	0,853	0,859
		10	0,838	0,063	0,898	0,838	0,844
	100	1	0,838	0,063	0,898	0,838	0,846
		5	0,824	0,068	0,892	0,824	0,832
		10	0,721	0,108	0,860	0,721	0,726
	200	1	1,000	0,000	1,000	1,000	1,000
		5	0,806	0,075	0,886	0,806	0,808
		10	0,740	0,100	0,866	0,740	0,738
	400	1	0,544	0,177	0,827	0,544	0,533
		5	0,603	0,154	0,836	0,603	0,604
		10	0,574	0,165	0,831	0,574	0,570

MultiLayer Perceptron

n-gramas	k	Verdadeiro Positivo	Falso Positivo	Precisão	Recall	F-Measure
1	25	0,941	0,020	0,946	0,941	0,942
	50	0,956	0,014	0,958	0,956	0,956
	100	0,985	0,004	0,986	0,985	0,985
	200	1,000	0,000	1,000	1,000	1,000
	400	1,000	0,000	1,000	1,000	1,000
2	25	0,941	0,021	0,941	0,941	0,941
	50	0,926	0,026	0,927	0,926	0,926
	100	1,000	0,000	1,000	1,000	1,000
	200	1,000	0,000	1,000	1,000	1,000
	400	1,000	0,000	1,000	1,000	1,000
3	25	0,853	0,057	0,904	0,853	0,857
	50	0,853	0,054	0,888	0.853	0,855
	100	0,882	0,044	0,900	0,882	0,883
	200	1,000	0,000	1,000	1,000	1,000
	400	0,897	0,038	0,909	0,897	0,898

Redes Bayesianas

n-gramas	k	Verdadeiro Positivo	Falso Positivo	Precisão	Recall	F-Measure
1	**25**	0,794	0,068	0,816	0,794	0,796
	50	0,912	0,027	0,930	0,912	0,911
	100	0,956	0,014	0,963	0,956	0,957
	200	0,985	0,005	0,986	0,985	0,985
	400	1,000	0,000	1,000	1,000	1,000
2	**25**	0,912	0,032	0,919	0,912	0,912
	50	0,941	0,022	0,944	0,941	0,941
	100	0,971	0,011	0,971	0,971	0,971
	200	0,985	0,005	0,985	0,985	0,985
	400	0,985	0,006	0,986	0,985	0,985
3	**25**	0,868	0,051	0,898	0,868	0,870
	50	0,912	0,033	0,927	0,912	0,912
	100	0,926	0,027	0938	0,926	0,927
	200	0,952	0,018	0,957	0,952	0,953
	400	0,926	0,027	0,931	0,926	0,926