Universidade Federal de Santa Catarina
DAS - Departamento de Automação e Sistemas
DAS410058 - Aprendizado de Máquina

Trabalho Prático
Aprendizado Indutivo

Classificação de páginas da web usando aprendizado de máquina

Alunos
Alan Kunz Cechinel
Eduardo Marschall
Henrique Raduenz
Igor de Oliveira Silvestre

Professores
Jomi F. Hubner
Marcelo Stemmer

Florianópolis - Setembro de 2018

Sumário

1 Introdução

Este trabalho tem como objetivo demonstrar a aplicação e implementação de conceitos de aprendizado de máquina vistos em sala de aula. Os conceitos relacionados à *Data Mining* são empregados para resolver um problema de classificação de páginas web acadêmicas em categorias distintas.

Para solucionar esse problema de classificação, quatro agentes classificadores, disponíveis no software Weka, são treinados, cada um empregando uma técnica diferente. O primeiro deles é a árvore de decisão, onde o algoritmo J48 é implementado. O segundo também utiliza árvore de decisão mas aplica o algoritmo *Random Forests*. A terceira técnica é a de redes neurais com o algoritmo *MultilayerPerceptron*. A quarta técnica é a de Redes Bayesianas com o algoritmo *BayesNet*.

Os quatro agentes são avaliados conforme a capacidade deles em fazer a classificação correta das páginas. Para a fase de treinamento são tomadas 70% de páginas. Os 30% restantes são usados para validação das regras de classificação encontradas.

Os softwares utilizados são o Matlab para a fase de pré-processamento dos dados, envolvendo a seleção de atributos e preparação dos exemplos de treinamento e validação, e o Weka para fazer o treinamento dos agentes e a validação deles.

Este relatório está dividido da seguinte maneira: Na Seção 2 o problema de classificação é apresentado. Na Seção 3 são descritas as técnicas de classificação empregadas com os respectivos algoritmos. Na Seção 4 é apresentado o pré-processamento dos dados onde são descritos os critérios utilizados para a seleção dos atributos que irão descrever os exemplos. Na Seção 5 é descrita a implementação das técnicas e algoritmos. Na Seção 6 são avaliados os resultados obtidos. Os resultados são discutidos na Seção 7. Finalmente, na Seção 8 são apresentadas as conclusões.

Todos os arquivos de programação e resultados estão disponíveis em: https://github.com/alankc/ReconhecimentoPaginasWeb.

2 Descrição do problema de classificação

O problema consiste em classificar páginas web de conteúdo acadêmico em diferentes categorias. As categorias, já previamente definidas, são:

- Professores;
- Documentos;
- Cursos;
- Alunos;
- Projetos
- Funcionários;
- Outros.

Os dados fornecidos para a classificação consistem em páginas web, que os agentes necessitam identificar a qual categoria pertencem, com base em um treinamento prévio.

Para treinar os agentes classificadores, é necessário definir quais são os atributos que podem ser usados para diferenciar as páginas entre cada categoria. Após definidos os atributos é necessário alimentar os dados de treinamento para os agentes classificadores para que eles sejam treinados.

A Tabela 1 abaixo mostra a quantidade de exemplos fornecida por cada classe existente.

Classe	Curso	Dpto	Prof	Outro	Proj	Func	Estud	Total
Exemplos	930	182	1124	3764	504	137	1641	8282
Porcent	11,2%	2,2%	13,6%	45,4%	6,1%	1,7%	19,8%	100%

Tabela 1: Quantidade de exemplos fornecidos por classe.

3 Técnicas e algoritmos empregados

Como mencionado anteriormente quatro técnicas de aprendizado supervisionado são implementadas e comparadas quanto a sua capacidade de classificar as páginas web. Árvores de decisão e o algoritmo J48 estão descritos na Seção 3.1, *Random Forestes* na Seção 3.2, Redes Neurais na Seção 3.3 e o algoritmo *BayesNet* na Seção 3.4.

3.1 Árvores de Decisão - Algoritmo J48

Árvores de decisão são algoritmos de aprendizado de máquina que objetivam criar regras relevantes para tomada de decisão. No caso de classificadores a árvore de decisão é treinada para, a partir da raiz, ir recursivamente submetendo a instância à diferentes regras de julgamento e guiando ela até uma folha, que indica a classe à qual aquela instância pertence. Uma boa regra interna de julgamento é aquela em que os resultados da saída definem perfeitamente uma instância como pertencendo a uma classe ou outra. Cada regra interna está associada a um atributo.

Um dos algoritmos do tipo árvore de decisão é o J48. Ele primeiro constrói a árvore de decisão e no fim poda os galhos que geram *outliers* ou resultados muito específicos.

3.2 Árvores de Decisão - Algoritmo *Random Forests*

Segundo Ponmani [4], o algoritmo *Random Forests* consiste em montar várias pequenas árvores de decisão e unir elas para formar uma floresta. Segundo os autores, selecionar pequenas quantidades de exemplos para formar árvores pequenas a partir de cada grupo de dados, resulta em uma floresta que é menos sensível à exemplos ruidosos do que uma árvore formada a partir de muitos exemplos. Outro detalhe é que as árvores menores podem ser formadas somente com alguns atributos e não todos.

A classificação acontece aplicando cada instância para todas as árvores, a floresta toma como classificação final aquela que mais ocorreu nas diversas árvores [4].

3.3 Redes Neurais Artificiais - Algoritmo *Multylayer-Perceptron*

Classificadores baseados em redes neurais utilizam a matriz de neurônios e pesos para determinar se uma instância pertence a uma classe ou não. Durante a fase de treinamento os pesos são ajustados conforme os exemplos

fornecidos [7]. O ajuste dos pesos é feito comparando o resultado dado pela rede com o esperado, gerando um sinal de erro.

3.4 Redes Bayesianas – *BayesNet*

Classificadores baseados em redes Bayesianas utilizam distribuições de probabilidade para determinar as classificações, por isso são classificadores estatísticos. Eles preveem a probabilidade de uma instância pertencer a uma determinada classe.

4 Pré-processamento dos dados - Seleção dos atributos

Talvez a parte mais importante para se obter um bom agente classificador seja a de fornecer ao algoritmo os atributos mais relevantes possíveis. Para isso, é necessário extrair dos dados os atributos que melhor descrevem cada categoria. Isso faz parte do chamado pré-processamento dos dados.

Neste trabalho, os dados fornecidos estão em formato de páginas html. Por conta disso, inicialmente, optou-se por buscar atributos representados através da frequência de palavras. Logo, o processo para definir os atributos consistiu em contar as palavras para em seguida definir a sua relevância.

4.1 Contagem das palavras

Num primeiro momento foi implementado um *script* em Matlab para varrer todas as páginas e montar um dicionário, contendo palavras com sua respectiva frequência, para cada uma das classes (*course, department, faculty, other, project, staff e student*), ordenado de forma decrescente de acordo com a frequência das palavras. Este *script* está disponível no Apêndice A. Vale ressaltar que a função do Matlab que conta as palavras considera plural e singular como um só, desta forma colocando no dicionário a palavra em sua flexão numeral de maior frequência.

Com o dicionário construído, deu-se início ao processo de seleção de palavras relevantes como possíveis atributos, com o intuito de eleger as que mais representassem cada uma das categorias.

4.2 Listagem de atributos

Palavras que aparecem em grande quantidade em todas as categorias podem não ser relevantes, pois resultariam em atributos muito generalizados. Exemplos desse tipo de palavra são: *the*, *university* e *computer*.

Por outro lado, palavras que aparecem somente em algumas páginas, talvez, também não sejam relevantes, visto que resultam em atributos que representam pequenos grupos de uma classe. Exemplos desse tipo de palavra são: nome de pessoas, endereços de e-mail e números.

Para selecionar possíveis palavras relevantes, foi implementado um *script* em Matlab que com base no dicionário de palavras, seleciona as N primeiras palavras de cada categoria, insere cada um destes conjuntos de palavras em uma *string* e realiza a contagem das palavras contidas na *string*, dessa forma,

se obtém uma lista de palavras e em quantas classes cada uma delas estão presentes. Este *script* pode ser visualizado no Apêndice B.

Com o objetivo de levantar possíveis palavras relevantes escolheu-se listar as 50 primeiras que apareceram em pelo menos duas e no máximo em todas as classes. O resultado obtido foi uma lista com 71 palavras, presente na Tabela 2.

computer	research	web	last
programming	language	tue	welcome
page	design	links	conference
gmt	department	interests	phd
project	new	publications	associate
systems	parallel	development	j
information	network	model	thu
date	software	applications	fax
server	work	class	data
contenttype	course	problem	processing
texthtml	office	cs	general
science	email	postscript	mimeversion
home	students	c	technology
lastmodified	engineering	schedule	support
contentlength	group	graduate	currently
nov	distributed	reports	phone
university	lab	technical	cornell
time	algorithms	college	

Tabela 2: Lista de palavras.

Foi decidido adicionar mais um possível atributo, responsável por representar a qual universidade o dado pertence, uma vez que eventualmente algumas características poderiam ser importantes apenas se associadas a sua origem. Este campo foi obtido através da leitura da URL da página. Sendo que no atributo, o valor 0 representa universidade desconhecida, 1 representa Cornell, 2 Texas, 3 Washington e 4 Wisconsin. Foi escolhido chamar este atributo como endereço.

Além das palavras e do atributo anterior, foi incorporada a contagem das *tags* html: $href, src$ e li.

Estes possíveis atributos foram utilizados para montar o arquivo no formato (.arff) aceito pelo Weka onde cada página é representada através de uma linha com a frequência de cada uma das palavras e das *tags* html, a universidade de origem e ao final a classe que pertence.

O *script* em MATLAB utilizado para montar o arquivo está disponível no Apêndice C.

4.3 Seleção de atributos

Não necessariamente todos os possíveis atributos que foram selecionados são relevantes para que seja efetuada a classificação de forma satisfatória. Por conta disso, uma terceira etapa de pré processamento foi realizada através dos algoritmos de seleção de *features* do Weka.

Três métodos de seleção de atributos do Weka, pertencentes a aba *Select Features*, foram utilizadas para filtrar os atributos, podem ser visualizadas juntamente com seus parâmetros na Tabela 3.

Avaliador de Atributo	**Método de Busca**
CfsSubsetEval -P 1 -E 1	GreedyStepwise -T -1.79E308 -N -1 -num-slots 1
CorrelationAttributeEval	Ranker -T -1.79E308 -N 21
InfoGainAttributeEval	Ranker -T -1.797E308 -N 21

Tabela 3: Lista de filtros de atributos.

Como a primeira ferramenta trouxe como resultado 21 atributos, escolheu-se para as outras duas, que utilizam método de busca *Ranker*, definir que deveriam trazer também 21 possíveis atributos. Após a filtragem agrupou-se os atributos selecionados por cada método de seleção para serem os atributos utilizados nos classificadores, resultando em um total de 29 atributos que podem ser visualizados na Tabela 4.

computer	department	associate
gmt	course	thu
server	office	fax
science	email	mimeversion
home	student	currently
lastmodified	tue	phone
contentlength	interest	(tag html) href
nov	classs	(tag html) src
university	welcome	(origem da página) endereco
research	phd	

Tabela 4: Lista de atributos finais.

Uma observação quanto a Tabela, como a palavra *class* é reservada pelo Weka, a contagem da palavra *class* é representada pelo atributo *classs*.

5 Treinamento dos agentes classificadores

Como a divisão dos dados foi definida como 70% para treinamento e 30% para validação com um método de divisão é randômico realizado pelo Weka, foi utilizado a mesma semente, número 1, para garantir que todos os agentes classificadores tivessem os mesmos conjuntos de dados. A seguir são apresentadas as telas de configuração para cada um dos classificadores utilizados no Weka.

A Figura 1 traz a tela de configuração do algoritmo J48, onde o único parâmetro modificado foi o *confidence factor*, este parâmetro recebeu um valor menor do que possuía para evitar *overfitting*.

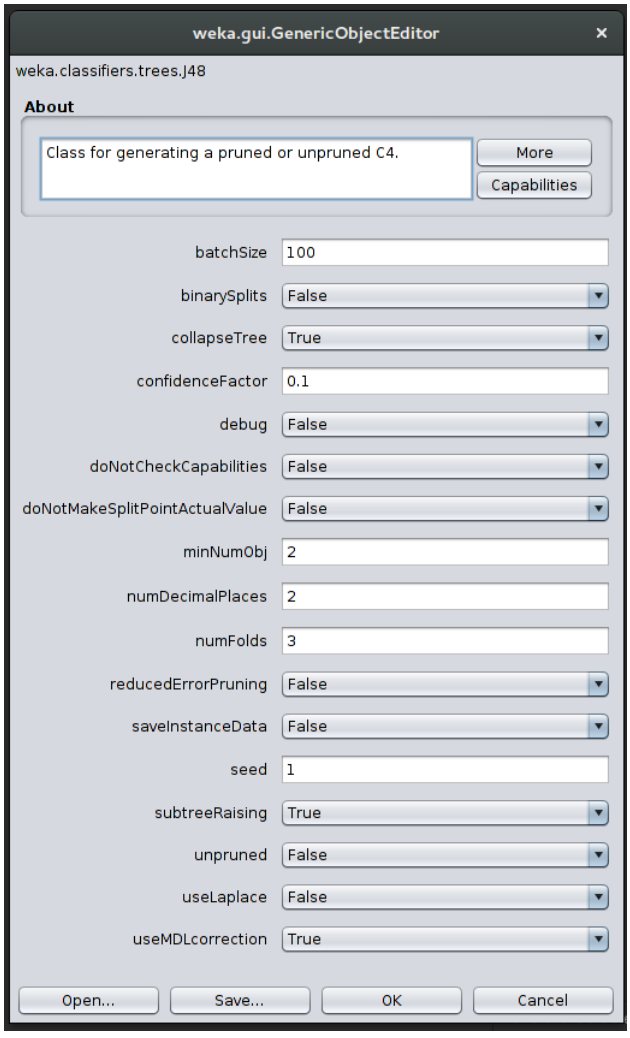

Figura 1: Tela de configuração do algoritmo J48.

Para o algoritmo *Random Forest*, a configuração apresentada na Figura 2 foi mantida exatamente como a sugerida pelo programa.

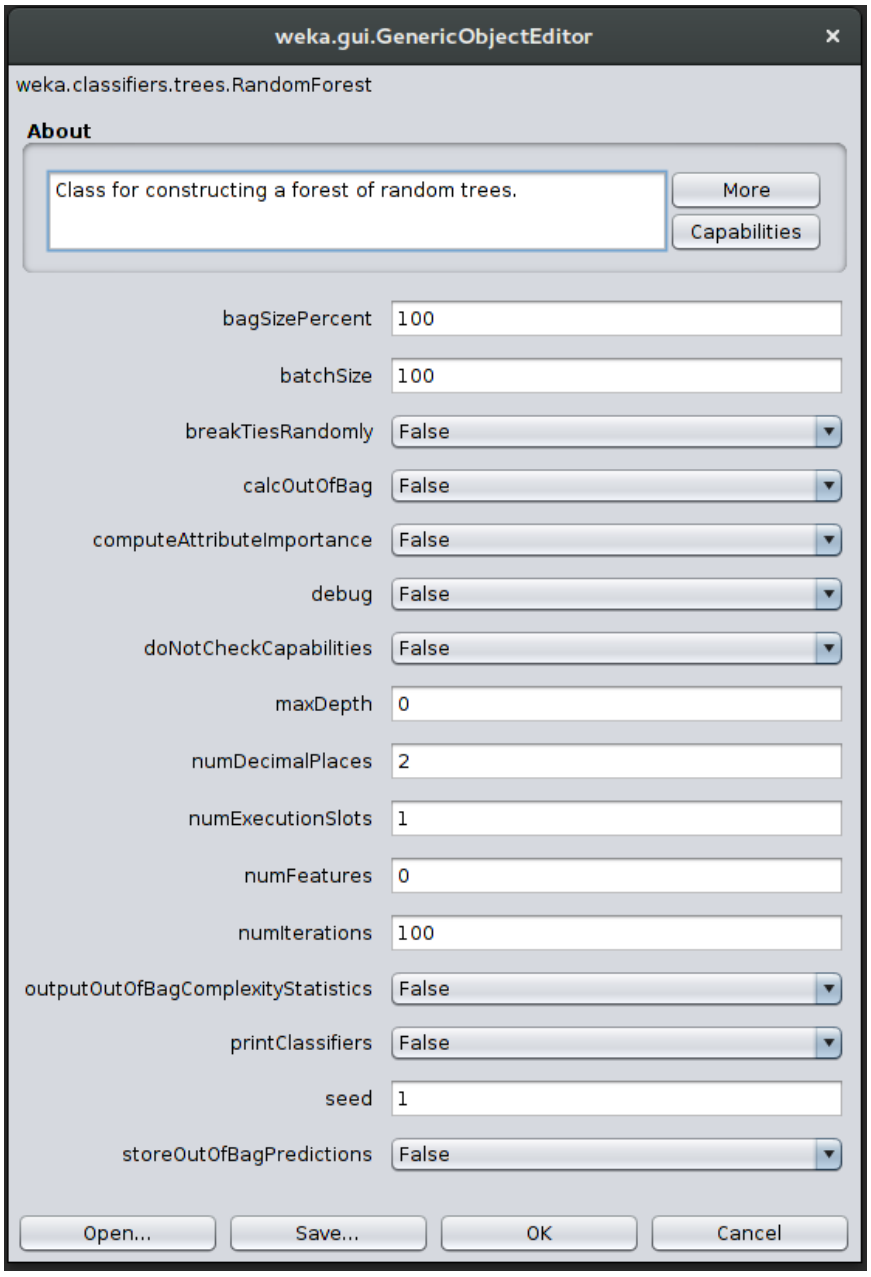

Figura 2: Tela de configuração do algoritmo Random Forest.

Para a rede neural artificial do tipo Multilayer Perceptron, a configuração presente na Figura 3 também foi mantida como o Weka sugere. Vale ressaltar

que o valor "a"no campo *hidden layers* significa que na camada intermediária a rede possui o número de neurônios dado pela Equação 1, onde a é o número de neurônios, A o número de atributos de entrada e C o número de classes de saída.

$$a = \frac{A + C}{2} \tag{1}$$

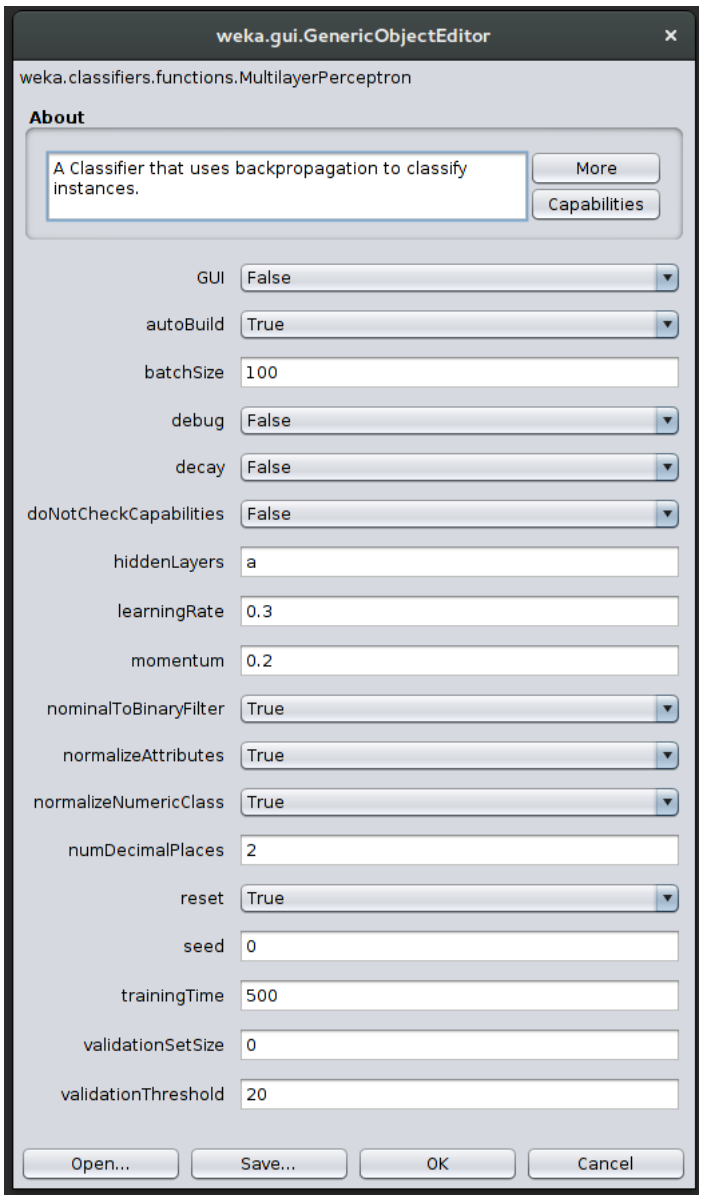

Figura 3: Tela de configuração do algoritmo Multilayer Perceptron.

Finalmente, na configuração do algoritmo *Bayes Net*, apresentada na Figura 4, o único parâmetro modificado foi o método de busca (campo *search algorithm*), pois utilizando o *Simulated Annealing* se obteve melhores resultados do que com o sugerido pela ferramenta.

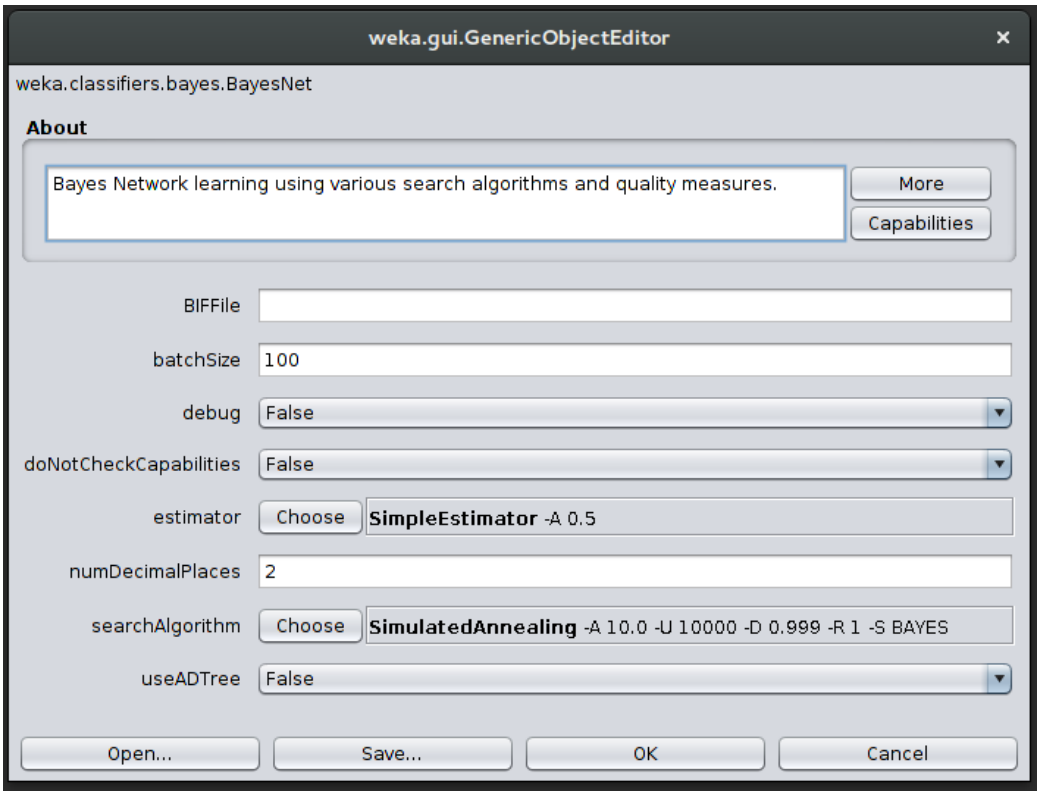

Figura 4: Tela de configuração do algoritmo Bayes Net.

6 Resultados

Nessa seção é apresentado os resultados obtidos utilizando cada método apresentado anteriormente. Utilizou-se para isso 70% dos dados para treinamento de cada algoritmo e os 30% restante como teste de verificação. Para se ter uma verificação mais detalhada e de fácil visualização, adicionou-se as matrizes de confusão de cada método. Essa matriz relaciona a classificação dada para cada instância em relação a classe verdadeira de cada caso.

6.1 Árvores de Decisão - Algoritmo J48

Com um total de 2485 instâncias, esse método foi capaz de classificar corretamente 1786 desses casos, obtendo uma taxa de acerto de 71,8712%. Os resultados mais detalhados podem ser vistos na sua matriz de confusão na Tabela 5

Classificação → Correto ↓	Curso	Dpto	Prof	Outro	Proj	Func	Estudante
Curso	**154**	0	9	67	7	0	23
Departamento	8	**25**	4	5	3	0	13
Professor	17	8	**204**	37	10	6	56
Outro	20	3	19	**1055**	6	1	44
Projeto	11	3	27	38	**45**	3	38
Funcionário	1	1	11	8	0	**1**	17
Estudante	17	11	55	82	7	3	**302**

Tabela 5: Matriz de confusão para o algoritmo J48.

A Figura 5 mostra qual foi a precisão atingida em cada classe através do método J48.

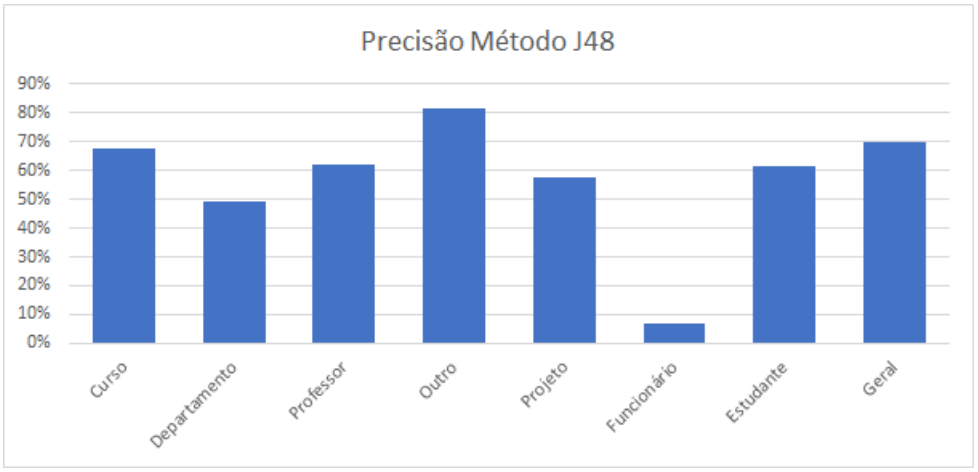

Figura 5: Gráfico de precisão do algoritmo J48

Verifica-se que para esse método houve classes que tiveram uma precisão muito boa, como na classe "outros"que teve esse quesito superior a 80%, enquanto a classe "funcionário"obteve-se uma precisão inferior a 10%.

6.2 Árvores de Decisão - Algoritmo *Random Forests*

Esse algoritmo foi capaz de classificar corretamente 1887 das 2485 instâncias testadas, obtendo assim uma taxa de acerto de 75,9356%. A matriz de confusão do método *Random Forests* pode ser visto na Tabela 6.

Classificação → Correto ↓	Curso	Dpto	Prof	Outro	Proj	Func	Estudante
Curso	**190**	0	4	46	2	1	17
Departamento	9	**22**	6	3	4	0	14
Professor	13	3	**223**	33	6	0	60
Outro	23	1	12	**1068**	8	0	36
Projeto	15	1	28	39	**51**	0	31
Funcionário	3	1	8	10	1	**2**	14
Estudante	16	1	52	66	8	3	**331**

Tabela 6: Matriz de confusão para o algoritmo *Random Forests*.

Observando a Figura 6 pode-se observar que com exceção da classe funcionários que obteve precisão próxima dos 30%, o restante das classificações obtiveram um resultado bastante parecido, obtendo somente outras três classes com precisão abaixo de 70%.

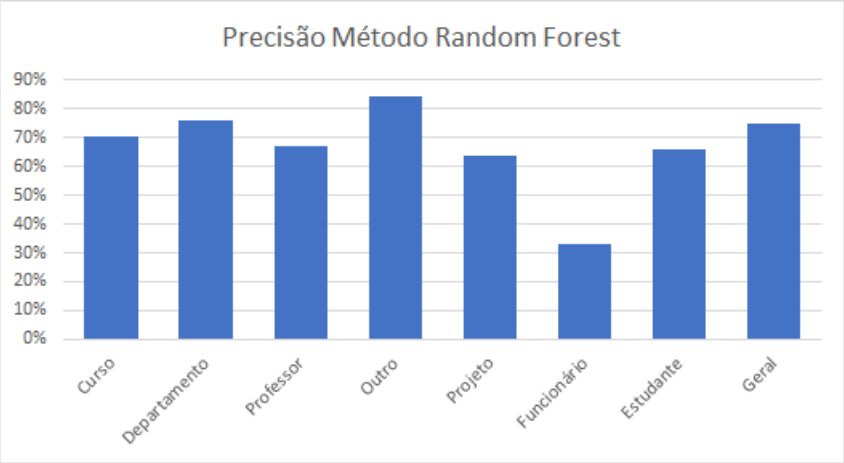

Figura 6: Gráfico de precisão do algoritmo *Random Forest*

6.3 Redes Neurais Artificiais - Algoritmo *Multylayer Perceptron*

A taxa de acerto desse método foi de 69,6982%, com uma classificação correta de 1732 atributos. Sua matriz de confusão e seu gráfico de precisão podem ser vistos na Tabela 7 e na Figura 7, respectivamente.

Classificação → Correto ↓	Curso	Dpto	Prof	Outro	Proj	Func	Estudante
Curso	**167**	3	2	66	8	0	28
Departamento	1	**24**	2	6	4	0	16
Professor	15	1	**150**	49	39	1	101
Outro	20	6	12	**1001**	21	0	49
Projeto	6	1	7	35	**58**	0	53
Funcionário	3	1	1	17	5	**0**	13
Estudante	19	0	13	115	14	0	**332**

Tabela 7: Matriz de confusão para a rede neural *Multylayer Perceptron*.

Percebe-se que com esse método não houve acertos para as páginas de funcionários. O restante das classes teve uma taxa de precisão variada, obtendo-se resultados com valores abaixo de 40% e acima de 80%.

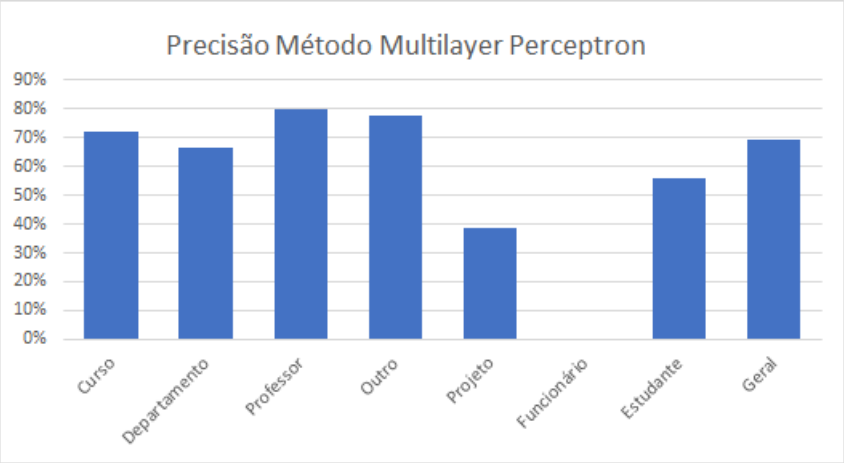

Figura 7: Gráfico de precisão do algoritmo *Multilayer Perceptron*

6.4 Redes Bayesianas – *BayesNet*

Com esse método, obteve-se uma taxa de acerto de 75.1308%, sendo que das 2485 instâncias, 1867 foram classificadas corretamente.A matriz de confusão e o gráfico de precisão obtidos com esse método podem ser vistos na Tabela 8 e na Figura 8, respectivamente.

Classificação → Correto ↓	Curso	Dpto	Prof	Outro	Proj	Func	Estudante
Curso	**175**	0	8	51	1	0	25
Departamento	1	**28**	6	4	4	0	15
Professor	14	3	**219**	22	15	3	62
Outro	27	2	19	**1043**	16	1	40
Projeto	13	3	20	31	**58**	3	37
Funcionário	2	0	6	5	1	**3**	22
Estudante	13	7	48	59	7	2	**341**

Tabela 8: Matriz de confusão para o algoritmo de Redes Bayesianas.

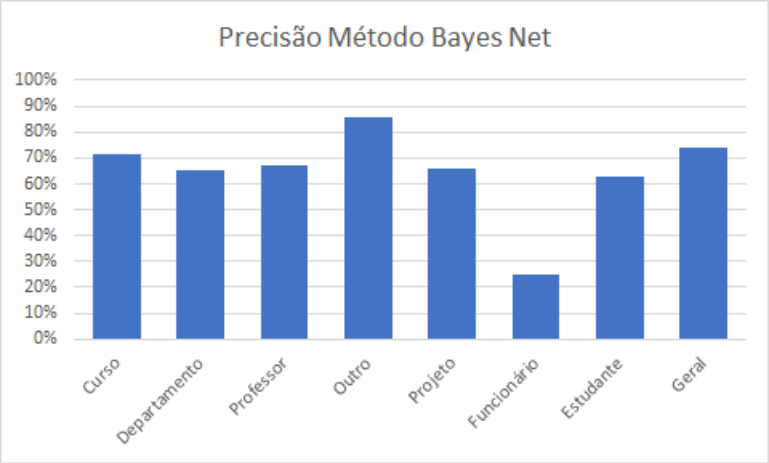

Figura 8: Gráfico de precisão do algoritmo *Bayes Net*

Assim como nos métodos anteriores, a taxa de acertos para a classe "funcionários"foi bastante baixa enquanto as precisões das outras classes ficaram todas acima de 60%.

7 Discussão dos resultados

Foi visto que a precisão de classificação obtida não foi tão expressiva. Algumas possíveis causas para isso foram identificadas.

Em relação ao pré-processamento dos dados, foi percebido que a aplicação dos algoritmos do Weka de seleção das *features* mais relevantes, não necessariamente resultaram em agentes mais bem treinados, pois a capacidade de classificação deles se mostrou similar aos casos em que estes algoritmos não foram aplicados. A principal vantagem é que eles encontraram um número menor de atributos que conseguem treinar os agentes e obter uma capacidade de classificação similar. Isso resulta numa maior eficiência do classificador.

Foi visto que alguns sites classificados na categoria *other* são na verdade sites que deveriam originalmente ter sido classificados como sendo de outras categorias, como por exemplo os sites:
"other/wiscosin/http_^^www.cae.wisc.edu^~ece552^conduct^conduct", que claramente é um site de um curso/disciplina;
"other/wiscosin/http_^^www.bocklabs.wisc.edu^mbinfo^profiles^Carroll,Sean", que claramente é um site de um professor;
"other/wiscosin/http_^^www.bocklabs.wisc.edu^mbinfo^profiles^White,John", que é novamente um site de professor. Estes exemplos demonstram que existe ruído nos dados. Esse tipo de ruído dificulta o treinamento dos agentes e também a validação deles.

Observou-se também que todos os algoritmos tiveram resultados muito ruins para a classe "funcionário". Isso pode ser explicado primeiramente pela quantidade inferior de páginas desse tipo quando comparada a outras classes (vide Tabela 1). Uma possível explicação para o fato de a classe "departamento"ter resultados bons, apesar de ter poucos exemplos pode ser pela existência de atributos nas páginas dessa classe que as diferenciam e classificam de forma eficiente em relação a outros tipos de páginas.

8 Conclusão

Em relação às técnicas utilizadas, foi visto que existe uma grande quantidade de algoritmos para gerar classificadores. De uma maneira generalizada pode ser afirmado que todos apresentam resultados satisfatórios. No entanto, para obter resultados excelentes é necessário ter grande conhecimento de cada técnica, de suas limitações e de quais fases de pré-processamento trazem mais qualidade aos dados.

Este trabalho prático propiciou o aprofundamento nos conhecimentos de *Data Mining* vistos em sala de aula. Além de permitir explorar na prática a aplicação de diferentes algoritmos, o que mostrou que o uso das técnicas de aprendizado de máquina requer muito esforço nas fases de pré-processamento e seleção de atributos relevantes. Dessa maneira, a grande disponibilidade de algoritmos e facilidade de uso pode instigar o analista de dados a partir rapidamente para a escolha da técnica de classificação, quando na verdade é crucial se ter dados representativos para cada uma das classes, assim como conhecer as características deles para escolher a técnica de classificação mais adequada.

Referências

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/webkb-data.gtar.gz. Acessado em: 28 de Setembro de 2018.

[2] Umadevi, S., Marseline, K. S. J.. A survey on Data Mining Classification Algorithms. International Conference on Signal Processing and Communications (ICSPC'17) July, 2017.

[3] Beniwal, S., Arora, J.. Classification and Feature Selection Techniques in Data Mining. International Journal of Engineering Research & Technology (IJERT) Agosto, 2012.

[4] Ponmani, S., Samuel, R., VidhuPriya, P.. Classification Algorithms in Data Mining - A Survey. International Journal of Advanced Research in Computer & Technology (IJARCET). Janeiro, 2017.

[5] Mitchell, T. M.. Machine Learning. McGraw-Hill Science/Engineering/-Math. Março, 1997.

[6] Zaki, M. J., Meira Jr., W.. Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press. Fevereiro, 2014. USA.

[7] Han, J., Kamber, M.. Data Mining: Concepts and Techniques. 2nd ed. Morgan Kaufmann Publishers. 2006. USA.

[8] https://machinelearningmastery.com/feature-selection-to-improve-accuracy-and-decrease-training-time/. Acessado em: 29 de Setembro de 2018.

[9] https://machinelearningmastery.com/an-introduction-to-feature-selection/. Acessado em: 29 de Setembro de 2018.

Appendices

A Contador de palavras

```
paths = {'course', 'department', 'faculty', 'other', 'project',
    'staff', 'student'};
root = 'C:\Users\cechi\Desktop\webkb\#\*\*';
wC = {};
for p = 1:1:size(paths,2)
    disp("Para a pasta: " + paths(1, p));

    str = "";

    rootTemp = replace(root,'#', strtrim(paths(1, p)));
    files = dir(fullfile(rootTemp));

    for i = 1:1:size(files,1)
        if files(i).isdir == false
            path = [files(i).folder '\\' files(i).name];
            str = str + lower(extractFileText(path));
        end
        if mod(i,100) == 0
            disp(i);
        end
    end

    %apaga tags html
    str = eraseTags(str);

    %variavel que contem os dicionarios
    %necessario matlab >= r2017b
    wC{p} = wordCloudCounts(str);
end
```

B Selecionador de palavras

Os parâmetros são respectivamente, as N primeiras palavras do dicionário, o dicionário, o mínimo número de classes em que uma palavra deve aparecer e o máximo número de categorias que uma palavra deve aparecer.

```
function [listaDePalavras] = definirPalavras(qtdPalavras, wCC,
    qtdInicio, qtdFim)
    palavras = {};
    for i = 1:1:size(wCC, 2)
        palavras{i} = wCC{1, i}{1:1:qtdPalavras, 1};
    end

    palavrasDisjuncao = "";
    for i = 1:1:size(palavras, 2)
        palavrasI = palavras{:, i};
         for j = 1:1:size(palavrasI, 1)
             palavrasDisjuncao = palavrasDisjuncao + " " +
                 palavrasI(j);
         end
    end
    palavrasArquivos = wordCloudCounts(palavrasDisjuncao);

    listaDePalavras =
        palavrasArquivos{find(palavrasArquivos{:,2} >= qtdInicio
        & palavrasArquivos{:,2} <= qtdFim), 1};
end
```

C Montador de arquivo de dados

```
root = 'C:\Users\cechi\Desktop\webkb\#\*\*';
paths = {'course', 'department', 'faculty', 'other', 'project',
    'staff', 'student'};
%campos '' significa que o plural foi desconsiderado
words = {'computer','computers','programming',''...};
universities = {'cornell', 'utexas', 'washington', 'wisc'};

for p = 1:1:size(paths,2)

    disp("Para a pasta: " + paths{1, p});
```

```
rootTemp = replace(root,'#', strtrim(paths{1, p}));
files = dir(fullfile(rootTemp));

for i = 1:1:size(files,1)
    if files(i).isdir == false

        path = [files(i).folder '\\' files(i).name];
        str = extractFileText(path);
        str = lower(str);

        %contando palavras
        wc = wordCloudCounts(str);

        %verificando se as palavras da lista estao
        %entre as que foram encontradas no arquivo
        data = {};
        for j = 1:1:size(words, 2)
            data{j} = 0;
            if ~strcmp(words{j},'')
                index = find(wc{:,1} == words{j});
                if (~isempty(index))
                    data{j} = wc{index, 2};
                end
            end
        end

        %agrupando plural e singular
        dataToPrint = {};
        for k = 1:1:(size(data, 2)/2)
            dataToPrint{k} = data{k * 2} + data{k * 2 - 1};
        end

        %nome da universidade
        dataToPrint{size(words, 2)/2 + 1} = 0; %uma qualquer
        for n = 1:1:size(universities, 2)
            if ~isempty(strfind(files(i).name,
                universities{n}))
                dataToPrint{size(words, 2)/2 + 1} = n;
                break;
            end
        end

        dataToPrint{size(words, 2)/2 + 2} = paths{p};
```

```
            printFile("dados.arff", dataToPrint);

        end
        if mod(i,100) == 0
            disp(i);
        end
    end
end
```