4
Experimentos

A estratégia `V-Wrapper` descrita por Zheng et. al (ZSW07), resumida no Capítulo 2, foi implementada com a finalidade de comparar um método baseado em atributos visuais com o algoritmo proposto `NCE`, descrito no Capítulo 3.

A Seção 4.1 descreve as métricas utilizadas para aferir a qualidade do `V-Wrapper` e do `NCE`. Em seguida, na Seção 4.2, o ambiente onde os algoritmos foram desenvolvidos e testados é descrito. Descrevemos na Seção 4.3 o processo de construção do corpus de exploração e teste e suas características. A Seção 4.4 detalha os experimentos com os algoritmos `V-Wrapper` e `NCE`.

4.1
Métricas

As métricas utilizadas para aferir a qualidade utilizam o conceito de *bag* definido na Seção 3.1. Para construir a *bag* de um texto, as palavras são separadas por espaços e tabulações. Por exemplo, considere o seguinte parágrafo que representa o conteúdo relevante de uma página D:

```
Começou oficialmente hoje (29) o Rally de Dubai (UAE
Desert Challenge), nos Emirados Árabes, válido como etapa
do Campeonato Mundial. O brasileiro Jean Azevedo, que corre
nas motos, marcou o 8º melhor tempo na etapa que teve 444
quilômetros.
```

A *bag* associada a esta página é o conjunto:

B = {Começou,oficialmente,hoje,(29),o,Rally,de,Dubai,(UAE,Desert, Challenge),nos,Emirados,Árabes, válido,como,etapa,do,Campeonato, Mundial.,O,brasileiro,Jean Azevedo,que,corre, nas,motos, marcou, o,8º,melhor,tempo,na,etapa,que,teve,444,quilômetros.}

Para definir as fórmulas seja $Rel(D)$ o conteúdo relevante de uma página D, então o *recall* de um texto C com relação a D é definido pela Fórmula 4-1.

$$Recall(C, D) = \frac{|bag(C) \cap bag(Rel(D))|}{|bag(Rel(D))|} \tag{4-1}$$

O *precision* de C com relação a D é definido pela Fórmula 4-2.

$$Precision(C, D) = \frac{|bag(C) \cap bag(Rel(D))|}{|bag(C)|} \tag{4-2}$$

A medida $F1$ de C com relação a D é a média harmônica do *precision* e do *recall* e é definida pela Fórmula 4-3.

$$F1(C, D) = \frac{2 \times Precision(C, D) \times Recall(C, D)}{Precision(C, D) + Recall(C, D)} \tag{4-3}$$

Seja o seguinte texto o conteúdo extraído C:

`Começou oficialmente hoje (29) o Rally de Dubai (UAE Desert Challenge), nos Emirados Árabes, válido como etapa do Campeonato Mundial.`

Perceba que neste caso C esta contido em D. Assim o *recall* será dado por

$$Recall(C, D) = \frac{20}{39} = 0.51.$$

O *Precision* para o exemplo dado é

$$Precision(C, D) = \frac{20}{20} = 1.0.$$

A medida $F1$ será calculada como na seguinte equação:

$$F1(C, D) = \frac{2 \times 1 \times 0.51}{1 + 0.51} = 0.67.$$

Veja que a intuição da medida *precision* é medir a taxa de acerto do texto que foi extraído pela heurística em relação ao conteúdo relevante. Enquanto o *recall* mede o quanto do conteúdo relevante está contido no que foi extraído pela heurística.

No exemplo dado todas as palavras extraídas estavam corretas, por esta razão a taxa do *precision* foi de 1.0. O *recall* mostra que foi extraído aproximadamente metade do conteúdo relevante.

Assim essas medidas são bem intuitivas de se lidar. Além disso, elas são práticas, pois dispensam a inspeção manual do resultado da heurística. No entanto algumas vezes elas podem ser rigorosas demais. Se por exemplo o texto de uma notícia é pequeno e a heurística extraiu também uma legenda de direitos autorais logo abaixo da notícia, então pode ficar a impressão que se

capturou muito a mais e uma inspeção humana poderia julgar que o resultado foi bem satisfatório.

4.2 Ambiente

A máquina utilizada para testes dos algoritmos `V-Wrapper` e `NCE` possui as seguintes características técnicas:

- Processador Intel Core 2 Duo 1.73Ghz;
- Memória 2Gb;
- Sistema Operacional Debian Linux Kernel 2.6.26-1-686;
- Interpretador Python 2.5.2;
- Software weka 3.5.7;
- xulrunner 1.9.

O *software* weka é uma coleção de algoritmos para tarefas de mineração de dados[1]. O projeto *weka* é código aberto e portanto seu código pode ser diretamente utilizado.

A plataforma xulrunner [2] é mantida pelo grupo mozilla. Através desta plataforma é possível utilizar aplicações que o navegador mozilla contém. Neste trabalho utilizamos as bibliotecas que produzem a árvore DOM renderizada em um navegador baseado na plataforma mozilla.

4.3 Preparação do Corpus

O corpus é um conjunto de páginas *web* de notícias. Para coletar estas páginas foi utilizado um *crawler* que obteve diversas páginas de uma lista composta de portais de notícias selecionados manualmente. A partir do conjunto de páginas que foi obtido pelo *crawler* foi feita uma seleção manual das páginas que eram notícias para compor o corpus.

O corpus foi separado em corpus de exploração (CE) e o corpus de avaliação (CA). O corpus de exploração foi utilizado para ajustar os parâmetros do `NCE` apresentados no Capítulo 3. O algoritmo `NCE` foi apenas testado no corpus CA com seus parâmetros já fixos pelo corpus CE.

O corpus CE possui 140 páginas de 95 portais diferentes e o corpus CA tem 339 notícias de 23 portais - veja a lista dos portais de notícias do corpus de exploração e avaliação no Apêndice A.

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] https://developer.mozilla.org/en/XULRunner

O conteúdo relevante das notícias foi selecionado manualmente. Esse conteúdo é comparado com a saída do algoritmo para que se possa aferir a qualidade do `NCE`.

4.4 Experimentos

Os experimentos foram executados no algoritmo `NCE` e no algoritmo `V-Wrapper`. Nas subseções 4.4.1 e 4.4.2 são apresentados respectivamente os resultados do `NCE` e do `V-Wrapper`. Finalmente, na subseção 4.4.3 analisamos os resultados obtidos.

4.4.1 Resultados NCE

Dois testes foram feitos para o algoritmo NCE: um no conjunto CE e o outro no conjunto CA. Veja na Tabela 4.1 a média aritmética das métricas para cada conjunto de dados.

Tabela 4.1: Resultados para o algoritmo NCE.

Conjunto	Recall	Precision	$F1$
CE	0.93	0.9	0.9
CA	0.94	0.88	0.88

A Tabela 4.2 mostra o resultado do `NCE` quando o módulo de busca por título é desabilitado. Veja que o resultado é ligeiramente alterado.

Tabela 4.2: Resultados para o algoritmo NCE sem o módulo de busca por título.

Conjunto	Recall	Precision	$F1$
CE	0.92	0.9	0.89
CA	0.95	0.87	0.87

O resultado `NCE` executado sem os módulos de remoção de comentário e busca por título pode ser visto na Tabela 4.3. Veja que estes módulos fizeram diferença no resultado do corpus CA.

A Tabela 4.4 mostra os resultados do `NCE` por *site*. Veja que o resultado para o portal `www.watfordobserver.co.uk` está bem inferior em relação aos demais. Isso ocorre porque cada linha da notícia neste portal é separada por uma *tag* `<p>` e não a *tag* `<br>`, como ocorre comumente para quebra de linha

Tabela 4.3: Resultados para o algoritmo NCE sem os módulos de remoção de comentários e busca por título.

Conjunto	Recall	Precision	$F1$
CE	0.93	0.9	0.9
CA	0.95	0.83	0.86

em páginas *web*. Assim a notícia fica em uma subárvore muito fragmentada. Esta fragmentação dificulta a busca pelo nó separador, visto que cada folha, ou seu pai, deve ter um número mínimo de 200 caracteres (parâmetro α_0). A subárvore acaba sendo descartada porque suas folhas tem um número muito menor que α_0. A variação mostra que mesmo se diminuir o valor do parâmetro a medida $F1$ não aumenta, pois o *precision* diminui consideravelmente.

A Tabela 4.4 também mostra que apenas 3 *sites* testados obtiveram uma medida $F1$ menor que 0.8. O *site* que alcançou o melhor resultado foi o `www.iht.com`.

A Tabela 4.5 mostra o tempo de execução para cada conjunto de dados. Este tempo esta dividido nas duas fases principais; na leitura das páginas (isso inclui a montagem da estrutura de dados) e no algoritmo.

Veja na Figura 4.1 um histograma do tempo do algoritmo por página. O tempo do `NCE` fica bastante concentrado entre 0 e 0.2.

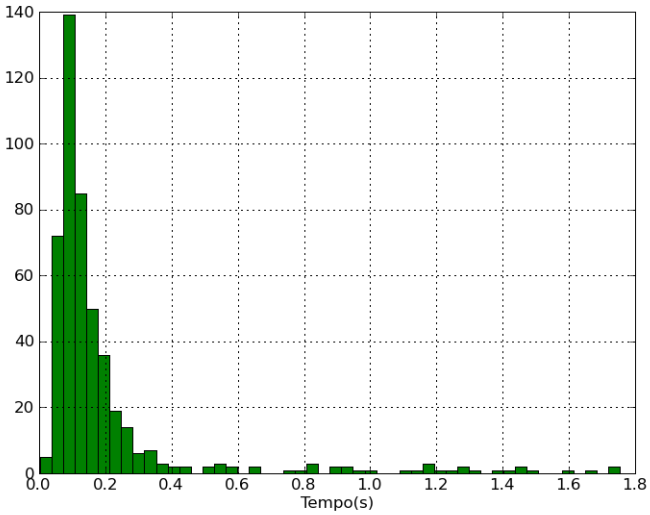


Figura 4.1: Histograma de Tempo do NCE no corpus CE

4.4.2

Tabela 4.4: Resultado do NCE por *site* do conjunto CA

Site	*Recall*	*Precision*	F1
www.theadvertiser.com	0.97	0.93	0.94
www.dublinpeople.com	0.87	0.94	0.88
www.wspa.com	0.93	0.86	0.86
www.delmarvanow.com	0.96	0.86	0.89
www.nationaudio.com	1.00	0.95	0.98
www.watfordobserver.co.uk	0.93	0.54	0.67
www.thedailyreporter.com	0.99	0.63	0.76
www.nysun.com	0.99	0.94	0.96
www.baltimoresun.com	0.83	0.89	0.80
www.eastandard.net	0.99	0.97	0.98
articles.lancasteronline.com	0.97	0.90	0.93
seattletimes.nwsource.com	0.96	0.97	0.97
www.bradenton.com	0.96	0.91	0.91
www.beaumontenterprise.com	0.94	0.81	0.86
www.greenvilleonline.com	0.69	0.98	0.74
www.thehawkeye.com	0.98	0.83	0.88
www.gtowntimes.com	0.90	0.77	0.82
www.iht.com	**0.99**	**0.99**	**0.99**
news.sky.com	0.92	0.86	0.88
www.thevalleychronicle.com	0.97	0.96	0.96
www.news24.com	0.99	0.91	0.95
dailyunion.com	0.86	0.99	0.86
www.silive.com	0.92	0.87	0.89

Resultados V-Wrapper

Antes de relatar os resultados do algoritmo `V-Wrapper` é importante destacar algumas decisões de implementação. A primeira diz respeito a representação das folhas bloco da estrutura de dados. Os autores descrevem que o texto da página fica sempre nas folhas bloco, porém em alguns casos no HTML não fica explícito que decisão tomar. Por exemplo, seja o seguinte código HTML:

```
< html >
<body>
<div>
Texto1 <b> Texto2 </b> Texto3
</div>
</body>
</html>
```

Quando o nó `<div>` é renderizado por um navegador, ele alcança um

Tabela 4.5: Tempo para o algoritmo NCE no corpora

Fase	Tempo(s)
Leitura dados	25.85
Algoritmo	104.27
Total	132.12

tamanho de 1256 por 20 pixels e o nó `<b>` 63 por 20 pixels. Como a área do nó `<b>` é diferente de 0 ele será transformado em um novo bloco, que será filho do bloco formado pela *tag* `<div>`. As strings *Texto1* e *Texto3* devem ficar em uma folha como os autores explicam, porém neste caso `<div>` não será um folha. Assim quando ocorre um caso deste, a decisão na nossa implementação foi criar um bloco que possui os mesmos atributos do nó pai com o texto. No exemplo dado cria-se um nó com os mesmo atributos do nó `<div>`, porém contendo as *strings* *Texto1* e *Texto3*.

Outra informação que não esta clara no artigo foi em relação aos atributos estatísticos. Assim quando é dito número da imagem assumimos, por exemplo, que é o número de imagens nos blocos abaixo do nó e este número representa o atributo número de imagens do nó.

Foi implementado um processo automático para rotular o conjunto de treino, pois o seu gabarito difere do algoritmo `NCE`. O gabarito para `V-Wrapper` são os rótulos dos blocos indicando a qual classe pertence. A fim de alcançar este objetivo cada gabarito do `NCE` é lido e transformado em um saco de palavras e comparado com cada nó da sua respectiva estrutura em blocos. Quando 85% das palavras do conteúdo de um nó pertencerem ao conteúdo do gabarito ele é rotulado como conteúdo ou título, dependendo do caso.

Para aferir a qualidade do processo automático de rotulação os nós rotulados em uma página, como título e conteúdo, tiveram seu textos unidos e comparados com o texto selecionado manualmente como conteúdo relevante da página. Para o corpus CE a comparação de cada página resultou em uma média da medida $F1$ de 0.96.

Após a rotulação automática foi realizado um processo de correção manual do rótulo de cada nó.

O classificador utilizado para treinar e testar foi a árvore de decisão J48 do *software* weka [3] com os parâmetros nos valores padrão. Os autores utilizaram Adaboost, mas a árvore de decisão na nossa implementação apresentou resultados melhores.

[3]http://www.gsi.dit.upm.es/ lssii/soft/papr/weka/doc/weka.classifiers.j48.J48.html

Tabela 4.6: Lista de Testes para o algoritmo V-Wrapper

Número	Conjunto de Treino	Conjunto de teste
1	CE	CA
2	CA	CE

Tabela 4.7: Validação cruzada para o algoritmo V-Wrapper

Número	Conjunto de Dados
3	CE
4	CA

Tabela 4.8: Resultados para o algoritmo V-Wrapper

Número	Recall	Precision	$F1$
1	0.7540	0.8356	0.7439
2	0.6888	0.8687	0.7107
3	0.85	0.71	0.76
4	0.34	0.49	0.34

Os autores usam a ferramenta `MSHTML`[4] para a renderização das páginas e na implementação deste trabalho usamos a plataforma Mozilla[5].

Após feita estas considerações os testes apresentados na Tabela 4.6 e a validação cruzada para cada linha na Tabela 4.7 foram realizados. A coluna *Número* é um identificador para o teste.

A validação cruzada foi feita dividindo o corpus em três subconjuntos de tamanhos aproximadamente iguais. Em cada iteração da validação cruzada um subconjunto foi utilizado para treino e os outros dois para teste. Por fim, é realizada a média dos resultados das três iterações da validação cruzada.

A Tabela 4.8 mostra a média das métricas *recall*, *precision* e $F1$ das páginas do conjunto de teste.

O resultado do `V-Wrapper` por *site* está na Tabela 4.9. O *site* que apresentou melhores resultados foi `www.iht.com`. Entretanto cinco *sites*, devido a problemas de renderização ou classificação, não apresentaram bons resultados.

A Tabela 4.10 mostra o tempo de execução do `V-Wrapper` para os testes da Tabela 4.8 nas duas fases principais do teste que é a renderização da página e o algoritmo, incluindo o teste pelo classificador, e também o tempo total.

[4]http://msdn.microsoft.com/en-us/library/aa741317.aspx
[5]https://developer.mozilla.org/en/Gecko

Tabela 4.9: Resultado `V-Wrapper` no corpus CA

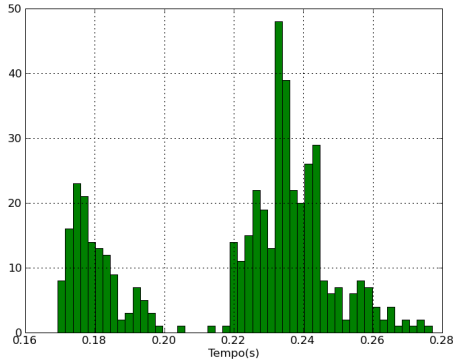
Site	*Recall*	*Precision*	F1
www.theadvertiser.com	0,9416	0,9778	0,9582
www.dublinpeople.com	0,9677	0,8028	0,8686
www.wspa.com	0,9146	0,7297	0,7917
www.delmarvanow.com	0,8618	0,9067	0,8826
www.nationaudio.com	0,1466	0,7411	0,1713
www.watfordobserver.co.uk	0,8715	0,9615	0,9085
www.nysun.com	0,7140	0,9957	0,7483
www.baltimoresun.com	0,8015	0,9525	0,8509
www.eastandard.net	0,9534	0,9710	0,9620
articles.lancasteronline.com	0,8607	0,9659	0,8794
seattletimes.nwsource.com	0,9460	0,9553	0,9490
www.bradenton.com	0,6280	0,9870	0,6821
www.beaumontenterprise.com	0,9603	0,7993	0,8702
www.greenvilleonline.com	0,8751	0,8612	0,8421
www.thehawkeye.com	0,9610	0,8107	0,8565
www.gtowntimes.com	0,1630	0,3282	0,1850
www.iht.com	0,9654	0,9955	0,9800
news.sky.com	0,0641	0,2880	0,1007
www.thevalleychronicle.com	0,8608	0,9941	0,8917
www.news24.com	0,2229	1	0,3483
dailyunion.com	0,7991	0,9979	0,8682
www.silive.com	0,8762	0,4584	0,5541

Este tempo não inclui o tempo de treino.

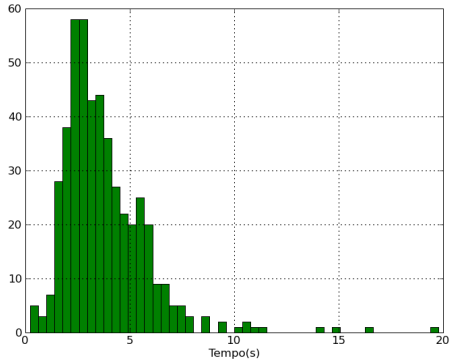
Para dar uma idéia da distribuição do tempo do algoritmo por página no corpora veja a Figura 4.2(a). Veja que o tempo do algoritmo no corpus se concentra entre $0.17s$ e $0.20s$ e entre $0.22s$ e $0.25s$. A distribuição do tempo de renderização pode ser vista na Figura 4.2(b), veja que o tempo varia aproximadamente entre $1s$ e $7s$, porém a maioria se concentra na região entre $2s$ e $3s$.

Tabela 4.10: Tempo para o algoritmo `V-Wrapper`

Fase	Tempo(s)
Renderização	1817.65
Algoritmo	105.81
Total	1923.46



4.2(a): Histograma de Tempo do algoritmo V-Wrapper no corpora



4.2(b): Histograma de Tempo de renderização do V-Wrapper no corpora

4.4.3 Análise dos Resultados

A implementação do algoritmo `V-Wrapper` neste trabalho diferiu em alguns aspectos da implementação original, como destacado na Seção 4.4.2. Isto reflete na diferença dos resultados reportados em (ZSW07) e nos resultados aqui encontrados. O melhor resultado na nossa implementação do algoritmo `V-Wrapper` foi uma medida $F1$ de 0.76. No entanto alguns resultados ficaram bem abaixo do limite de 0.76.

A renderização das páginas ocupa um tempo considerável no algoritmo `V-Wrapper`, podendo alcançar mais de 80% do tempo total de execução do algoritmo. Quando a página possui mais recursos, tal como uma propaganda em *flash*, sua renderização pode demorar ainda mais. Embora a plataforma de renderização utilizada neste trabalho seja diferente da utilizada por Zheng et. al em (ZSW07), a utilização de recursos gráficos do sistema operacional certamente exige mais memória e portanto mais tempo de execução.

Apesar da extração dos atributos gráficos consumir muito tempo de execução do algoritmo eles são boas descrições da página, pois as páginas exibem o seu conteúdo de forma a ser facilmente identificável pelo usuário. Por esta razão o conteúdo de muitas páginas de notícia são comumente exibidos na região central do navegador.

O tempo do algoritmo `V-Wrapper` é aproximadamente o mesmo que o do algoritmo `NCE`, desconsiderando a renderização. No entanto ao considerar a renderização do algoritmo `V-Wrapper` o tempo total do algoritmo se torna até dezoito vezes maior que o tempo do algoritmo `NCE`.

A representatividade do corpora foi comprovada através dos testes com os parâmetros do `NCE`. Os testes com os parâmetros mostram que ambos os corpus se comportam da mesma maneira quando o parâmetro é variado.

Finalmente, o algoritmo `NCE` utiliza apenas uma árvore DOM como sua estrutura de dados, portanto a replicação do código é facilitada.