TÉCNICO LISBOA

Extração de Informação de Páginas Web

Tiago Miguel Jacinto Fernandes

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Orientadores: Prof. Pável Pereira Calado
Prof. Manuel Fernando Cabido Peres Lopes

Júri

Presidente: Prof. Miguel Nuno Dias Alves Pupo Correia
Orientador: Prof. Pável Pereira Calado
Vogal: Prof. José Luís Brinquete Borbinha

Janeiro 2021

Agradecimentos

Primeiramente, quero agradecer a toda a gente do campus Taguspark que me acompanhou nesta fase de universidade. Desde a licenciatura até ao fim do mestrado nunca me faltou apoio seja de colegas, funcionários ou mesmo professores.

Quero agradecer também a uma das pessoas mais importantes neste processo universitário, a Liliana Oliveira, que esteve sempre presente desde o ínicio até ao fim, todas as barreiras encontradas foram ultrapassadas com a sua ajuda, foi super importante nas grandes decisões tomadas, sempre que estava mais em baixo, foi ela a puxar para cima e foi claramente o grande pilar desta fase da vida.

Depois, quero deixar um agradecimento especial à Raquel Prata, por me ter apoiado na fase inicial da tese quando as dúvidas sobre os caminhos a seguir eram muitas, mostrou sempre o seu lado sincero e da experiência ajudando-me imenso a seguir em frente sem medos.

Quero agradecer também ao excelente grupo de amigos que a faculdade me deu, em especial ao Diogo, ao Henrique, à Mariana e à Filipa, que me trouxeram sempre a visão mais divertida e ponderada dentro deste mundo complicado, foram as pessoas perfeitas para passar os tempos de forma mais leve e tranquila.

Agradeço ainda à minha irmã Joana, que me auxiliou imenso na fase final da tese, tendo sido super fundamental em termos de organização e trabalho, graças a ela consegui terminar tudo a tempo e horas.

Para finalizar, quero ainda agradecer à minha família mais chegada, por me ter apoiado e motivado sempre para os problemas que ia encontrando, nunca deixaram de me dar aquelas palavras de apoio sempre boas de ouvir, agradecer aos Sku Nalama, os meus grandes amigos de infância de Odivelas que estão e estarão sempre presentes na minha vida para tudo e todos e por fim agradecer obviamente, ao professor Pável Calado por meter dado a orientação, o suporte e partilhado o conhecimento necessário para a realização desta tese.

Para todos vocês, um grande Obrigado e um bem haja aqui do TigasFer.

Resumo

O problema da extração de conteúdo de páginas Web tem sido objeto de estudo desde a expansão da World Wide Web. O seu objetivo é separar o conteúdo principal de uma página, como o texto de uma notícia, do conteúdo irrelevante, como anúncios e links de navegação.

A maioria das abordagens de extração de conteúdo opera ao nível do bloco, ou seja, a página Web é segmentada em blocos e, em seguida, cada um desses blocos é determinado como parte do conteúdo principal ou do conteúdo irrelevante da página Web.

Nesta tese, tentamos aplicar a extração de conteúdo a um nível mais profundo, ou seja, a elementos HTML. Durante o decorrer da tese, investigamos a noção de conteúdo principal mais de perto, criamos um conjunto de dados de páginas Web cujos elementos foram marcados manualmente como parte do conteúdo principal ou como conteúdo irrelevante e aplicamos Aprendizagem Automática (Machine Learning) a esse conjunto de dados para separar o conteúdo principal do conteúdo irrelevante. Propomos um algoritmo denominado X-CEX para resolver este problema de extração de conteúdo, baseado no Algoritmo Content Extractor. Finalmente, este método e os seus processos são avaliados a usar um conjunto de dados diferente de páginas Web, rotulados manualmente.

Palavras Chave

Extração de Informação; Páginas Web; Aprendizagem Automática; Aprendizagem Supervisionada; Content Extractor;

Abstract

The content extraction problem has been a subject of study ever since the expansion of the World Wide Web. Its goal is to separate the main content of a webpage, such as the text of a news, from the noisy content, such as advertisements and navigation links.

Most content extraction approaches operate at a block level; that is, the webpage is segmented into blocks and then each of these blocks is determined to be part of the main content or the noisy content of the webpage.

In this thesis, we try to apply content extraction at a deeper level, namely to HTML elements. During the course of the thesis, we investigate the notion of main content more closely, create a dataset of webpages whose elements have been manually labeled as either part of the main content or the noisy content, and apply machine learning to this dataset in order to separate the main content and the noisy content. We proposed an algorithm called X-CEX to solve this content extraction problem, it was based on the Content Extractor Algorithm. Finally, this method and it's processes are evaluated using a different dataset of manually labeled webpages.

Keywords

Information Extraction; Wep Pages; Machine Learning; Supervised Learning; Content Extractor;

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

IE	Information Extraction
IR	Information Retrieval
DOM	Document Object Model
BTE	Body Text Extraction
W3C	World Wide Web Consortium
SST	Site Style Tree
VIPS	Vision-Based Page Segmentation Algorithm
TF	Frequência da Palavra
IBDF	Inverse Block Document Frequency
tp	Positivo Verdadeiro
fp	Positivo Falso
tn	Negativo Verdadeiro
fn	Negativo Falso
WWW	World Wide Web

1

Introdução

Conteúdo

Hoje em dia, a tecnologia assume um papel muito forte na sociedade ao ponto de automatizar muitos processos existentes realizados por humanos. A extração de informação de páginas Web não é exceção e, como tal, começou-se a desenvolver formas de melhorar a visualização das mesmas facilitando assim a vida às pessoas que diariamente acedem à Web.

1.1 Motivação e Problema

Este trabalho é de facto um desafio interessante, pois visa a análise e melhoria dos sistemas existentes para visualização de notícias, publicações, artigos, entre outros, na Web. Apesar de ser todo um novo mundo com bastante adesão, existem problemas que não podem ser escondidos, a verdade é que o público mais idoso e o público mais novo sofre muito de ignorância neste mundo da tecnologia, e acaba por ser enganado e se deixar ir pelo que vê na Web. Nem tudo o que vemos é verdade, pelo que, este trabalho tem também bastante importância relativamente a estes pontos, desta forma, ajudamos à visualização de conteúdo na Web e assim a facilitar a vida às pessoas com maior dificuldade, sendo esta uma grande motivação para a realização do trabalho. As páginas na World Wide Web (WWW), hoje em dia, estão compostas por bastante conteúdo irrelevante chamado de ”ruído”, Lan Yi [1] separou este ”ruído”em 2 grupos de acordo com a sua granularidade, o *Ruído Global* e o *Ruído Local*. Ruído Global é relacionado com os conteúdos irrelevantes em páginas com grande granularidade, que, por norma, não é menor que uma página. Tipicamente mais associado a páginas copiadas e/ou duplicadas, páginas antigas sem utilização, entre outras. Ruído Local é relacionado com pequenas regiões dentro de uma página Web, tipicamente, associado a publicidade, foto de capa, links de navegação, imagens decorativas, entre outras. Neste trabalho, o foco é remover ruído local de páginas com conteúdo informativo como notícias, artigos e publicações.

As páginas Web, atualmente não mostram os conteúdos com clareza, pois, interesses externos são levantados, principalmente as receitas provenientes de publicidade, como tal, as páginas Web deixam de ser totalmente informativas e passam a conter muito conteúdo irrelevante e sem importância. Grande parte das páginas Web ficam compostas por esta categoria de conteúdos que, não podem ser classificados como conteúdos informativos. A esta categoria de blocos são chamados blocos sem conteúdo (Non-content blocks). Os blocos sem conteúdo são tipicamente publicidades, imagens, plug-ins, logótipos, caixas de pesquisa, links de navegação, links relacionados, cabeçalhos, rodapés e ainda informação de direitos de autor, estes blocos são bastante comuns em páginas dinamicamente geradas. Por outro lado, chamamos blocos com conteúdos (content blocks) aos blocos que apresentam o conteúdo informativo que a página em questão assim propõe. Estes blocos são os blocos que se pretendem preservar e manter para assim obter uma visualização mais clara da notícia. Tipicamente estes blocos são compostos por texto, ou seja, ”strings”, no entanto, poderão eventualmente conter alguma

imagem que seja considerada também relevante para o corpo da página em questão.

O problema principal deste trabalho é, portanto, identificado e relacionado com a dificuldade da extração de conteúdo importante para os artigos e notícias publicadas na Web, este é o grande desafio por trás deste projeto, pois, envolve toda uma análise dos blocos das páginas de forma a poder separar o que é informativo e o que é irrelevante. Este trabalho deve responder a estes problemas com um método capaz de dar resultados eficientes ao extrair o conteúdo relevante de cada página.

1.2 Hipótese e Metodologia

Para a resolução do problema descrito acima, é proposto realizar uma análise do algoritmo Content Extractor, de modo a poder perceber as várias melhorias necessárias a realizar, os problemas e ainda os reais benefícios deste método. Desta forma pode-se ter uma ideia clara do que tem de ser feito e melhorado no que toca à extração de informação das páginas Web.

Após a análise, implementar algumas melhorias que sejam, de facto, uma evolução para o algoritmo em questão. A escolha deste algoritmo baseia-se no poder que tem nesta área, sendo considerado como um algoritmo com grande fiabilidade. A análise proposta tem o objetivo de podermos concluir mais detalhadamente no que toca à eficácia, precisão, complexidade de implementação e tempo de execução, sendo que, à partida, este já terá grandes resultados nestes pontos. Desta forma, podemos considerar diferentes maneiras de dividir a página, de analisar o conteúdo da mesma e ainda, de realizar a extração do conteúdo relevante.

Assim sendo, neste relatório, é apresentado o algoritmo X-CEX, implementado por mim com base no algoritmo Content Extractor, para a solução do problema mencionado. Apercebi-me que o Content Extractor é um algoritmo que, apesar dos bons resultados que obteve e da baixa complexidade computacional que tem nunca foi muito explorado, existe muito pouca pesquisa e testes aplicados neste algoritmo, o que faz com que tenha muitos pontos que podem ser melhorados. Assim sendo, pretendo com o algoritmo X-CEX (capítulo 4), apresentar várias alternativas e várias melhorias ao algoritmo base e poder compará-las de forma a apresentar o melhor método possível. É também o meu objetivo poder dar mais informação sobre estas variantes para um possível futuro trabalho baseado neste algoritmo, desta forma qualquer projeto que queira partir deste como base terá bastantes informações e vários caminhos possíveis a percorrer.

1.3 Objetivos Gerais

Este trabalho visa, conseguir apresentar uma solução prática para a visualização clara e simples dos conteúdos relevantes das páginas Web. Com esta solução é suposto alcançar outros objetivos como:

- **Conhecer outros trabalhos** - Desta forma conhecer e aprender os vários trabalhos e métodos já existentes para este propósito;

- **Avaliação de outros trabalhos** - Entender que trabalhos são mais acessíveis ou não para a realização do objetivo principal avaliando-os conforme algumas métricas;

- **Métricas de Avaliação** - Ler e analisar as várias formas de poder analisar os algoritmos e métodos existentes, e, perceber que estas métricas tendo o mesmo nome podem ser consideradas de forma diferente por cada pessoa;

- **Datasets** - Procurar e adquirir datasets que possam ser usados para avaliar os vários métodos existentes;

- **Machine Learning** - Entender melhor a área de Machine Learning aplicada a extração de informação e, poder passar esses conhecimentos;

- **Análise de páginas** - Perceber como funciona uma página Web e como pode ser dividida em secções independentes. Primeiro entender como gerir e trabalhar com a estrutura dessas páginas através das etiquetas HTML.

No final deste trabalho, o objetivo é conseguir dar respostas às perguntas ”Quais são as componentes que mais influenciam uma boa extração?”e ”De que forma posso acrescentar valor aos métodos existentes?”.

1.4 Contribuições

Acredito que este trabalho possa ter uma relevante contribuição para todos os utilizadores da Web, a visualização de notícias, artigos, publicações, entre outras é uma constante atualmente, pelo que, para todas as pessoas que usam a Web para este efeito poderá ser uma mais-valia a curto prazo.

Na prática, este trabalho, será uma grande contribuição para todos os que pretenderem saber mais sobre a área e, como funciona este processo de extração que vivemos.

Futuramente, tenho a certeza que haverá outras soluções mais práticas e ainda mais fiáveis, no entanto, apenas é possível lá chegar a melhorar e contribuindo gradualmente com o que podemos ajudar neste trajeto. Este trabalho é uma contribuição para quem puder e quiser desenvolver algo mais

no futuro e desta forma dar continuidade a uma evolução e melhoria contínua da área. O mundo não para, e como tal, a evolução destas áreas tem de acontecer com a evolução do tempo, chegar cada vez mais a um futuro mais automatizado.

1.5 Estrutura do Documento

Este trabalho está dividido e organizado em 6 secções:

1. **Introdução** - É apresentado o contexto do trabalho, os problemas encontrados e qual a abordagem sugerida para a solução do problema, são ainda apresentados os objetivos e contribuições que se pretendem alcançar com a realização do trabalho;

2. **Conceitos** - Neste capítulo, é apresentada uma visão geral sobre as páginas Web, sobre a área de Extração de Informação e ainda uns conceitos sobre Aprendizagem Automática;

3. **Trabalho Relacionado** - Neste terceiro capítulo, é pretendido apresentar os mais variados trabalhos, métodos, algoritmos e ferramentas existentes que, hoje em dia produzem resultados eficientes;

4. **Proposta de Solução** - Nesta secção pretende-se mostrar o que será desenvolvido e de que maneira de forma a satisfazer os objetivos propostos, apresentando assim a arquitetura da abordagem sugerida e todas as suas componentes;

5. **Testes e Resultados** - Neste penúltimo capítulo, são apresentados os datasets usados para a avaliação do trabalho, as métricas usadas para avaliar, todos os testes e respetivos resultados obtidos e ainda uma análise detalhada dos mesmos;

6. **Conclusão** - No último capítulo é realizado uma conclusão do trabalho, são apresentadas as limitações do mesmo e ainda o trabalho futuro a realizar;

2

Conceitos

Conteúdo

Neste capítulo são apresentados alguns conceitos relativos às Páginas Web, como estão estruturadas e como são extraídas as informações das mesmas, falo ainda sobre conceitos relativos à Extração de Informação e por fim apresento algumas características da Aprendizagem Automática (Machine Learning).

2.1 Páginas de Internet

As páginas Web [2] são subdivididas em pequenas secções baseadas no seu conteúdo, a estas secções chamam-se blocos, sendo assim chamado até ao fim do relatório. Posto isto, uma página Web é constituída por um conjunto de blocos, sendo que, estes blocos não são todos informativos. Na Figura 2.1[1], tem-se um exemplo de como é dividido uma página Web em blocos, no caso uma página da Yahooligans, estes blocos estão sempre entre uma etiqueta de abertura e uma etiqueta de fecho do bloco sendo estas etiquetas identificadas na estrutura HTML da página em questão.

Figura 2.1: Página da Yahooligans com os seus blocos identificados por caixas delineadas a vermelho

[1] http://people.cs.uchicago.edu/~xiaofei/WebImageSearch.html

Posto isto, todas as páginas Web acabam por ser representadas como uma árvore, tendo os seus blocos e sub-blocos que correspondem, no caso da árvore, a nós pai e nós filhos. Por fim, os nós finais que não têm nenhum filho acabam por ser chamados de nós folha, como, por exemplo, os elementos de texto de uma página.

As estruturas de árvore, são as formas mais fáceis de poder navegar dentro de uma página, dessa forma consegue-se organizar e modificar facilmente as marcações do código HTML. Se as páginas forem todas escritas corretamente, com todos os elementos, abertura e fecho de etiquetas e com um código HTML totalmente validado, será, no futuro muito mais fácil apresentá-las e trabalhá-las.

2.2 Extração de Informação

Nesta secção, são exploradas e descritas as várias abordagens usadas para a extração de informação e ainda a arquitetura de um sistema destes.

2.2.1 Abordagens para Extração de Informação

Um sistema de Extração de Informação pode ser elaborado através de inúmeras abordagens:

- **Abordagem baseada em dicionário**

  Ao aplicar uma lista de nomes relativos ao domínio em questão, efetua-se uma pesquisa dos mesmos no texto livre obtendo como soluções os nomes que se deparam em ambos. Este processo apresenta uma desvantagem crucial que é a necessidade de existir uma lista extenuante de nomes e termos a procurar, englobando erros ortográficos, variantes de nomes e ambíguos, abreviaturas, sinónimos.

- **Abordagem baseada em regras**

  São estabelecidas regras que são empregues ao texto livre e quando um determinado padrão é identificado, a ação é executada, como, por exemplo, Expressões Regulares [3], compostas por um padrão e uma ação a desenvolver [4]. As regras podem ser concebidas mediante duas metodologias: através da codificação manual em que é fundamental a existência de humanos peritos do domínio que definam as regras ou da utilização de algoritmos de Aprendizagem Automática onde o sistema adquire as regras de extração a partir de exemplos estruturados já existentes [4].

  A desvantagem que sobressai neste método está presente nas normas originadas segundo o parâmetro de codificação manual, uma vez que são tão específicas em domínio que esporadicamente são extensíveis a outros. Perante outra perspetiva torna a elaboração do sistema bastante extenso e pode excluir termos significativos que não equivalem exatamente aos padrões predefinidos.

- **Abordagem baseada em Aprendizagem Automática (Machine Learning)**

  A abordagem mencionada tem como finalidade o desenvolvimento de algoritmos para detetar automaticamente padrões em dados. Os algoritmos de Aprendizagem Automática podem ser fragmentados consoante a sua categoria de aprendizagem: aprendizagem supervisionada que é regularmente executada através da indução de um modelo apto para prever ocorrências futuras baseando-se num extenso conjunto de dados de treino [4]; aprendizagem semi-supervisionada distinguindo-se da anterior, pois necessita de uma quantidade de amostras excecionalmente menores; por fim a aprendizagem não supervisionada, onde não há diferenciação entre dados de treino e dados de teste, sendo processada toda a informação de entrada com o objetivo de formar uma espécie de resumo ou aglomeração, normalmente são algoritmos de agrupamento ou redução de dimensionalidade.

- **Abordagem Híbrida**

  Uma abordagem híbrida é posta em prática para conciliar, primeiro, as vantagens das díspares abordagens definidas anteriormente. Cada contexto onde são adotadas algumas técnicas de Extração têm especificidades distintas e nelas existe a necessidade de adaptar a solução.

2.2.2 Arquitetura de sistemas de Extração de Informação

Quando se trata de um sistema de Extração de Informação, os documentos dos quais se pretende extrair a informação correspondem aos dados de entrada. Os respetivos documentos devem-se encontrar em formato digital de modo que a informação e os dados neles integrantes possam ser analisados. É essencial, para a execução desta análise, a utilização de tecnologias de Reconhecimento Ótico de Caracteres para proceder à conversão e fornecer também um modelo com a definição das entidades e os campos a serem extraídos, com o propósito do sistema saber exatamente o que extrair [5]. Facultativamente é possível recorrer a Bases de Conhecimento, dicionários, glossários ou ontologias para identificar identidades. No final do método de Extração de Informação são adquiridos dados num formato estruturado que, a 'posteriori', podem ser analisados.

Cada sistema de Extração de Informação pretende responder a questões de domínios distintos, e que independentemente de terem algumas disparidades significativas, todos os sistemas possuem certas componentes em comum.

Segundo Piskorski e Yangarber [6], a arquitetura de um sistema de Extração de Informação, representada na Figura 2.2, deve considerar dois grupos principais de componentes linguísticos, os independentes do domínio e os dependentes. Os componentes independentes são:

Análise de Metadados (Meta-data Analysis): Extração do título, o corpo e estrutura do texto assim como a data do documento.

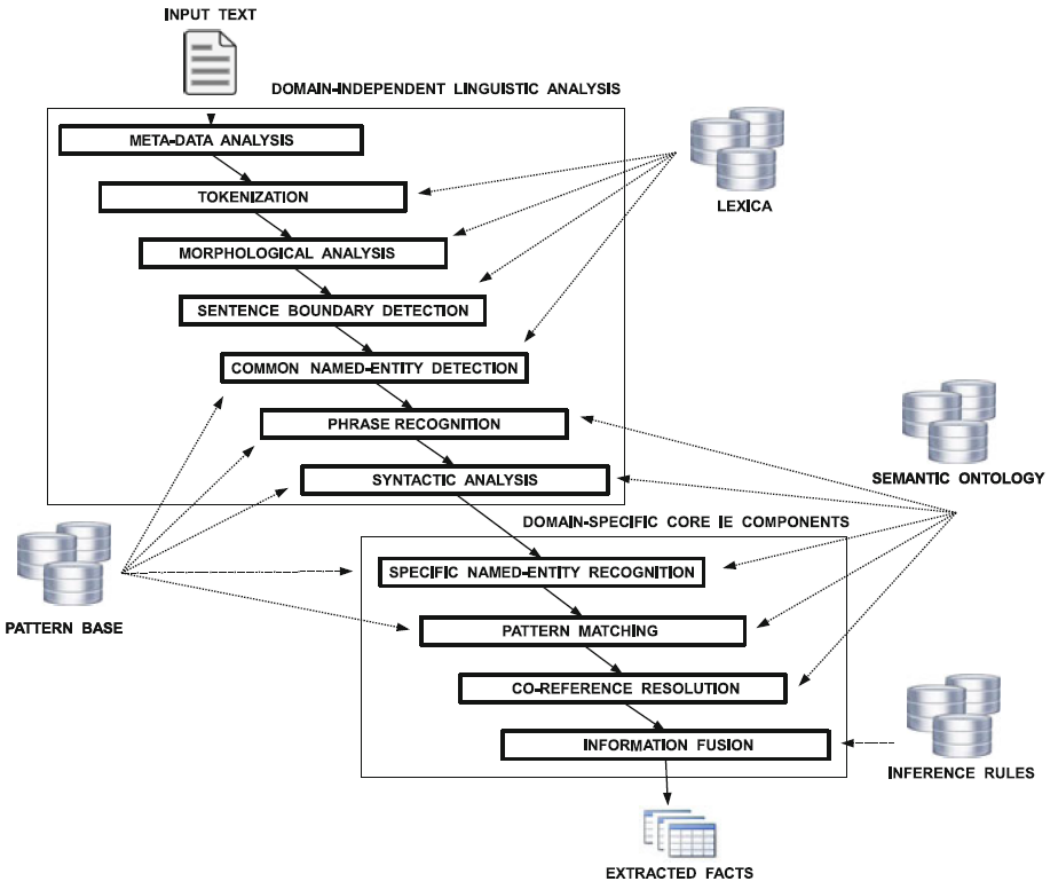

Figura 2.2: Arquitetura de um sistema de Extração de Informação

Tokenização (Tokenization): Fragmentação do texto em token (palavras) e o seu respetivo tipo de classificação.

Análise Morfológica (Morpholoical Analysis): Análise e Extração de Informação morfológica de cada token através da análise gramatical.

Deteção de frases (Sentence Boundary Detection): Segmentação do texto em frases, em que cada uma é representada por uma sequência de itens lexicais.

Deteção de entidades nomeadas (Common Named-entity Detection): Detetar as entidades, tais como, expressões temporais, numéricas ou referências geográficas.

Reconhecimento de frases (Phrase Recognition): Identificação de estruturas locais como grupos nominais, grupos verbais, frases proposicionais, siglas e abreviaturas.

Análise sintática (Syntactic Analysis): Análise da estrutura de frases, sendo identificados sujeito, predicado e complemento direto ou indireto. A análise pode ser profunda quando são analisadas todas as interpretações possíveis e as relações gramaticais dentro de uma frase, ou pode ser uma análise superficial, em que análise se limita à identificação de estruturas e fenómenos linguísticos

não recursivos e não tratados, assim como problemas de ambiguidade.

Já para os componentes linguísticas dependentes do domínio, destacam-se quatro tarefas principais que variam consoante os requisitos da aplicação:

Reconhecimento de Entidades Nomeadas (Name Entity Recognition): Tem como finalidade reconhecer e classificar nomes encontrados num texto livre em tipos predefinidos, como nomes de pessoas, de locais ou expressões numéricas, sendo que as entidades a extrair dependem do domínio da Extração de Informação.

Resolução de Correferência (Coreference Resolution): Corresponde à identificação da presença da mesma identidade num texto com variantes de nome, resumidamente, palavras diferentes que se referem à mesma entidade. A palavra pode estar representada como frase nominal, pronominal, nome ou implícito.

Extração de Relações (Relation Extraction): Obtém-se a identificação e classificação das relações entre identidades.

Extração de Eventos (Event Extraction) No decorrer das inúmeras tarefas é viável a aplicação de outros métodos com finalidade de melhorar os resultados adquiridos. São utilizados, por exemplo, padrões para identificar segmentos do texto, para extrair atributos de modo a preencher os campos no modelo predefinido e podem ainda ser aplicadas regras de inferência [7] para deduzir conclusões válidas de modo a preencher os campos definidos [6].

2.3 Aprendizagem Automática (Machine Learning)

A aprendizagem é o resultado da transformação da experiência em qualquer conhecimento e é nisto em que a Aprendizagem Automática se baseia. A Aprendizagem Automática consiste na criação de um algoritmo capaz de aprender através da construção de um modelo a partir de argumentos fornecidos com o objetivo de realizar previsões ou decisões com base na mesma data [8].

A principal utilidade da Aprendizagem Automática é a possibilidade de computorizar tarefas que os humanos realizam sem pensar e que não conseguem traduzir para código, como compreensão de imagem ou reconhecimento de fala, uma vez que se tratam de tarefas que podem aprendidas de forma eficaz se houver exemplos suficientes [8].

Ao longo dos tempos têm-se desenvolvido inúmeros tipos de aprendizagem, das quais três delas irão ser mencionadas no presente documento.

2.3.1 Aprendizagem Supervisionada

Baseia-se em algoritmos dependentes de exemplos fornecidos externamente, como o dataset de Treino, de modo a criar modelos onde seja possível fazer previsões acerca de outros exemplos. Assim, o objetivo da aprendizagem supervisionada é a construção de um modelo conciso na distribuição dos tipos de classificação. Seguidamente, o algoritmo é utilizado para atribuir rótulos a novos exemplos ou ao dataset de Teste, onde todos os valores são conhecidos à exceção dos rótulos.

De acordo com o autor, a parte inicial anterior à escolha do algoritmo é bastante importante, pois é aquela que realmente define a eficácia da previsão. Uma vez que podem faltar valores no processamento da informação e no dataset em estudo, a recolha de informação, incluída nesta primeira parte, vai ser integrada em ambos. O tratamento de informação vai ser preferencialmente desenvolvido por especialistas que usam métodos adequados, além disso, é também importante escolher os dados relevantes para a previsão que mais tarde promovam a rapidez e eficácia [9].

2.3.2 Aprendizagem Não Supervisionada

Esta variante de aprendizagem tem por base um processo em que não é necessário estabelecer um dataset de treino ou um dataset de teste, pois o algoritmo processa todo o dataset de maneira a criar um resumo da informação introduzida [8]. O tradicional é agregar o dataset em grupos de objetos semelhantes [10].

2.3.3 Aprendizagem Semi-Supervisionada

Por fim, a aprendizagem semi-supervisionada é uma associação entre as duas últimas aprendizagens mencionadas e pode ser aplicada de diferentes formas, sendo uma delas fornecer ao algoritmo não só argumentos não identificados, mas também alguma informação supervisionada, além disso, pode ser tratada como aprendizagem não-supervisionada acompanhada por algumas restrições.

Perante outra abordagem, é ver esta categoria de aprendizagem como se fosse uma aprendizagem supervisionada com informação adicional na distribuição de dados [11].

2.4 Sumário

Neste capítulo, foram então dados alguns conceitos bases e muito resumidos sobre as áreas de Extração de Informação e de Aprendizagem Automática, mas, acima de tudo, foi dada uma visão geral de como são representadas as páginas Web que visualizamos todos os dias nos nossos navegadores.

3

Trabalho Relacionado

Conteúdo

Com a evolução e a expansão da área da Internet, foram propostos vários algoritmos direcionados à extração de conteúdo de páginas. Muitos destes métodos baseiam-se em heurísticas que podem ser aplicadas no código HTML das páginas Web, nas suas respetivas árvores Document Object Model (DOM) ou mesmo no visual das páginas.

Nesta capítulo, os trabalhos referenciados estão agrupados separadamente em 5 categorias sendo que as categorias mais exploradas são as que têm maior relevância para o desenvolver do trabalho.

São descritos mais detalhadamente os algoritmos e métodos existentes relacionados com os sistemas de Information Retrieval (IR) e que mais se adaptam à resolução do problema. Caso seja do interesse do leitor, é recomendado que este leia mais sobre sistemas de Information Extraction (IE) [12], sobre as técnicas usadas [13] e ainda ter uma visão geral dos vários algoritmos e métodos que existem [14, 15].

3.1 Extração de Texto

Existem alguns métodos criados tendo como objetivo a identificação do texto principal de uma página Web [14, 15]. O método escolhido para ser descrito nesta secção é o Body Text Extraction (BTE) [16].

3.1.1 Body Text Extraction

Em 2001 foi proposto o algoritmo BTE por Finn, Kushmerick e Barry Smith [16]. O algoritmo BTE começa por separar todo o conteúdo de uma página em etiquetas HTML e palavras, sendo que, por consequência a página web é vista como uma sequência de bits. O bit é igual a 1 quando o token respetivo é uma etiqueta e é igual a zero quando é uma palavra. Assim sendo, esta sequência de bits é representada pela *document slope curve* que pode ser vista na Figura 3.1, onde o eixo do x representa o número de tokens vistos e o eixo do y representa o número de etiquetas HTML vistas. Nesta curva podemos detetar uma zona plana destacada a negrito, essa zona corresponde ao corpo principal de texto, pois é uma zona que não deteta nenhuma etiqueta HTML.

O Algoritmo BTE pretende encontrar um segmento na curva em que a inclinação é menor, no entanto, este segmento não pode ser demasiado curto, ou seja, precisa de ser suficientemente longo para poder corresponder a uma secção longa de texto. Mais concretamente, o algoritmo pretende encontrar dois índices *i* e *j* sendo que, entre eles o número de palavras encontrado é maximizado, enquanto antes do índice *i* e depois do índice *j* o número de etiquetas encontradas também será maximizado. Este segmento pode ser descrito pela seguinte função:

$$T_{i,j} = \sum_{n=0}^{i-1} B_n + \sum_{n=i}^{j} (1 - B_n) + \sum_{n=j+1}^{N-1} B_n, \tag{3.1}$$

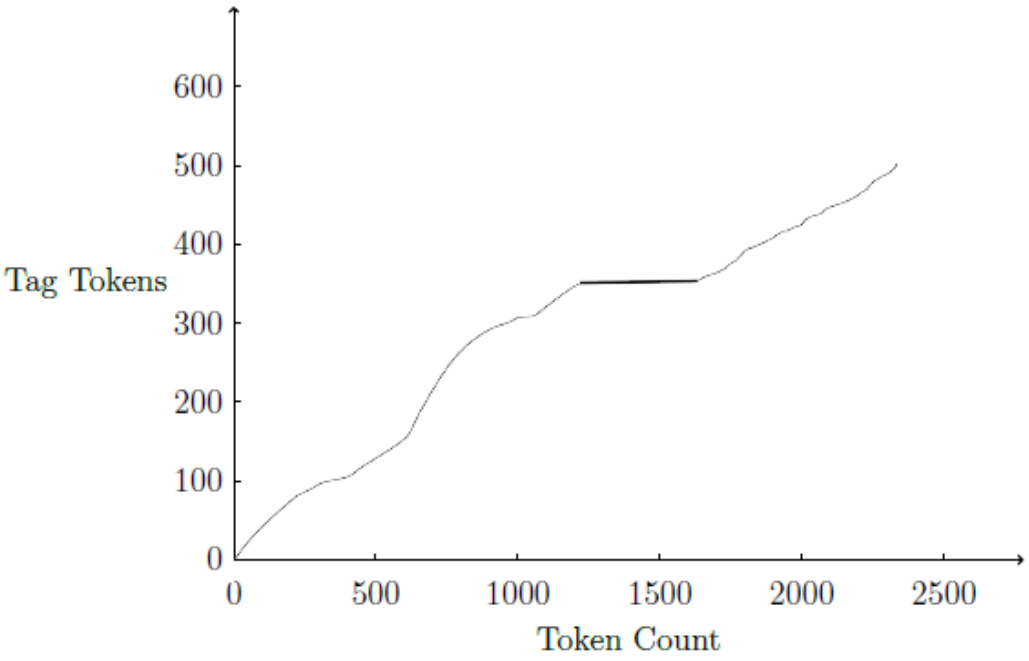

Figura 3.1: Document Slope Curve

onde *N* é o número total de tokens no documento.

O maior problema deste algoritmo é assumir que o corpo principal de texto se encontra todo junto, o que está errado, pois, pode haver várias frações de texto relevante separadas pela página. Entretanto, foi proposta uma melhoria a este método [17] para procurar vários segmentos de corpo principal de texto na curva, em vez de apenas 1 como foi originalmente proposto.

3.2 Extração de Conteúdo através de árvores DOM

DOM é uma 'interface' de linguagem neutra de programação para documentos HTML [18], ou seja, é uma forma de representação da página Web. Serve para facilitar o manuseamento da página, seja com alterações a efetuar ou para retirar qualquer informação da mesma tornando a tarefa muito mais facilitada. Normalmente usa-se DOM para atualizar uma página Web ou para se construir uma 'interface' de utilizador, avançada. Com o DOM pode-se mover itens dentro de uma página ou criar efeitos bastante interessantes sem precisar de recarregar a página.

Posto isto, o DOM representa documentos HTML numa estrutura de árvore, isso deve-se ao facto de, podermos ver uma página Web como uma árvore, sendo a raiz o elemento HTML com os seus respetivos filhos até chegar aos nós folha, isto é, nós que não têm nenhum filho, como se pode ver na Figura 3.2.

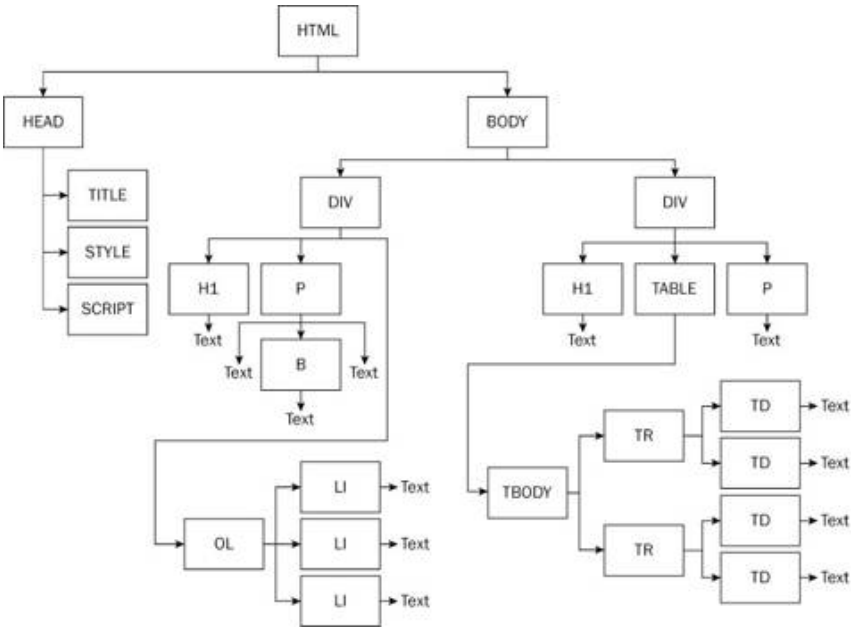

Figura 3.2: Árvore DOM de um documento HTML

As especificações do DOM são publicadas pela World Wide Web Consortium (W3C)[1]. Portanto, o DOM é um padrão de fato. Baseados no DOM foram publicados alguns métodos de extração de texto como, por exemplo um método usando as Site Style Trees (SSTs) ou ainda um método proposto por Gupta.

3.2.1 Site Style Tree

Este método proposto por Jadhav e Badhan [19] é baseado em criar uma SST a partir de um DOM para cada página Web, de forma a analisar corretamente o conteúdo da mesma e conseguir identificar os blocos com conteúdo relevantes.

Este método baseia-se nas árvores DOM e combina-as de forma a criar assim uma SST tal como mostra na Figura 3.3.

Nesta figura podemos observar 2 árvores DOM combinadas resultando assim na árvore SST onde é possível ver que apenas o fim da árvore difere face às progenitoras, isto porque, são analisadas ambas as árvores e combina-se todas as secções em comum, as que não são comuns passam a ser outro ramo da árvore. São usados dois valores de importância (importância da apresentação e importância do conteúdo) para encontrar a importância de um ”nó elemento”, quanto mais alto for este valor, mais similar com os blocos armazenados será, logo, menos relevante. Desta forma encontramos assim quais os nós com informação relevante da página Web.

Este método tem uma grande eficiência relativamente ao estilo de apresentação das páginas Web e

[1] https://www.w3.org/

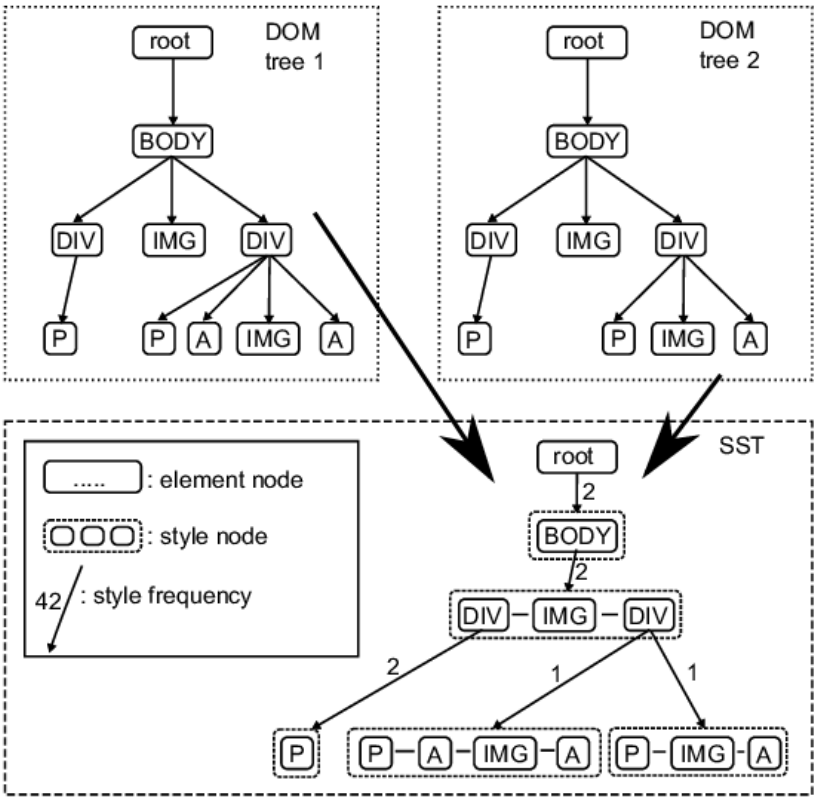

Figura 3.3: Árvores DOM 1 e 2 e SST respetiva

ao conteúdo do HTML. Apesar de identificar de forma eficiente estes blocos com conteúdo, este método apresenta níveis de complexidade de implementação muito elevados no que toca à construção destas SSTs.

3.2.2 Método de Gupta

Gupta, em 2003 [20] sugeriu uma nova abordagem baseada na árvore DOM de uma página Web. O algoritmo começa por transformar o código HTML da página numa árvore DOM sendo posteriormente aplicados 2 filtros nos conteúdos da árvore.

O primeiro filtro aplicado remove elementos como imagens, links, scripts e estilos. O segundo filtro consiste em remover publicidade, listas, e tabelas que não contenham nenhuma informação sendo este segundo filtro bastante mais complexo que o primeiro. Estes filtros aplicados regem-se por várias heurísticas diferentes, por exemplo, os atributos de ”href”e ”src”são comparados com uma lista de servidores comuns de publicidade e caso algum desses atributos esteja contido na lista então o nó respetivo é removido da árvore DOM.

Por fim, é apresentado o conteúdo final da árvore, sendo que, são removidas todas as etiquetas

ficando apenas com o texto dos nós, sendo este, dado como o conteúdo principal da página Web.

3.3 Extração de Conteúdo através da visão

Nesta secção é apresentado uma forma de extração de conteúdo das páginas onde é pretendido simular uma visão humana da estrutura da página Web. Um humano não consegue ver o código HTML de uma página nem tão pouco uma árvore DOM da mesma, apenas vê o visual da página e, como tal, nesta secção é especificado um algoritmo introduzido em 2003 chamado Vision-Based Page Segmentation Algorithm (VIPS) tendo o visual da página como base.

3.3.1 VIPS

O VIPS foi proposto por Cai, Yu, Wen, e Ma em 2003 [21] e pretende extrair a estrutura do conteúdo principal da página. Este algoritmo é aplicado recursivamente na árvore DOM de uma página Web.

Primeiramente neste algoritmo é aplicada a fase da extração de blocos, sendo que todos os nós da árvore são avaliados para verificar se representa ou não um bloco visual. Quando é feita esta avaliação, se um certo nó tiver vários filhos então esses filhos serão avaliados individualmente sendo por isso uma avaliação recursiva. No final desta tarefa, os blocos selecionados como blocos visuais irão ser adicionados a uma lista de blocos sendo todos os outros descartados.

A cada bloco da lista é atribuído um grau de coerência, este grau corresponde ao nível de conteúdo consistente que o bloco respetivo tem. Posto isto, é inicialmente pré-definido um treshold de grau de coerência, desta forma consegue-se alcançar uma certa granularidade na estrutura do conteúdo. Uma página com maior granularidade precisa de ser dividida em mais blocos e sub-blocos, ao contrário de uma página com baixa granularidade que fica composta por menos blocos, na Figura 3.4 tem-se um exemplo de uma página com baixa granularidade.

De seguida, entra a fase de deteção dos separadores, ou seja, são identificados todos os separadores entre blocos podendo dessa forma perceber que blocos estão mais visíveis na página tendo por isso mais peso na mesma. Assim sendo, os blocos com mais visibilidade estarão mais acima na hierarquia da página.

Para finalizar, o grau de coerência dos blocos desta estrutura de conteúdo é comparado com o treshold definido inicialmente e, caso seja inferior então esse bloco é considerado como uma sub-página e, como tal, será aplicado todo o algoritmo nesse bloco até ser obtido uma árvore apenas com blocos que satisfaçam o treshold.

Este algoritmo apenas pretende obter a estrutura do conteúdo da Página e não o conteúdo propriamente dito.

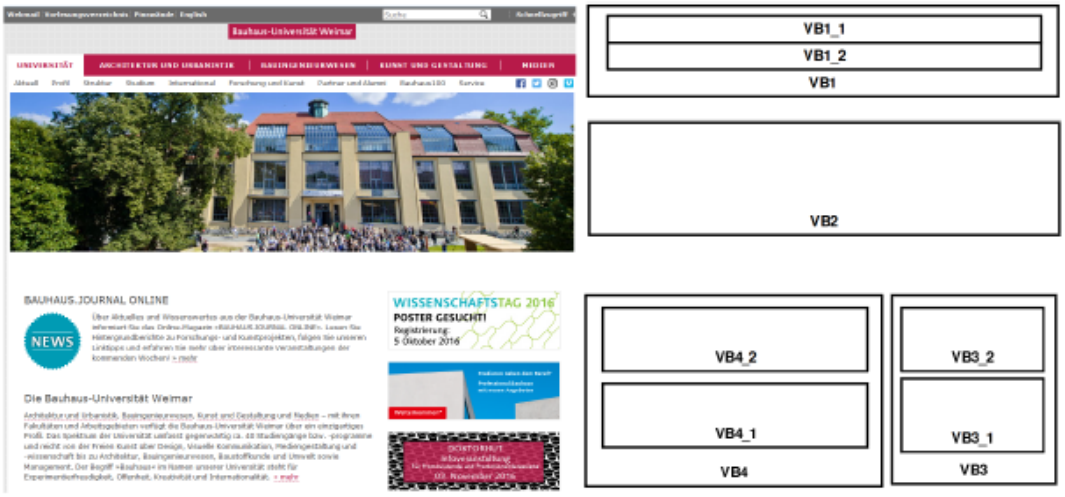

Figura 3.4: Página Web com baixa granularidade

3.3.2 Método de Liu

Wei Liu em conjunto com Xiaofeng Meng propuseram um acrescento ao VIPS em 2006 [22] ao pegar na estrutura final resultante do algoritmo e identificar qual o bloco que corresponde à região que contém a informação.

Para tal é tido em conta duas características bases da região com a informação, primeiro, estas regiões estão sempre centralizadas e segundo, estas regiões são normalmente as mais largas destacando-se no tamanho da página.

Depois de a região com a informação ser encontrada inicia-se o processo de extração da informação propriamente dita, essa extração é efetuada com base em várias características, na sua localização na região, normalmente alinhado à esquerda da mesma, toda a informação está junta, todas as partes de informação têm uma aparência similar e apresentação desses conteúdos segue uma ordem fixa.

3.4 Wrappers

Uma das maneiras para extrair informação de uma página Web é ao programar um *wrapper* que é um programa para extrair informação de uma certa fonte, em particular de uma página Web.

Muitas páginas Web incluem conteúdos dinamicamente gerados, como, por exemplo, características de produtos ou informações sobre dados estatísticos. Os wrappers pretendem restaurar estes conteúdos para a sua forma normal, sendo nestes onde obtêm os melhores resultados.

Aparentemente cada wrapper deve ser programado individualmente para cada coleção de páginas HTML. Este trabalho é o grande e principal problema desta abordagem, pois será bastante tedioso, taxa de erro elevada e é uma abordagem que consome imenso tempo.

Existem três maneiras de construir um wrapper [23], a partir de código manual, a partir de aprendizagem automática supervisionada e aprendizagem automática não supervisionada. Nesta secção é apresentado o AdEater tendo como base aprendizagem automática supervisionada.

3.4.1 AdEater - Método de Kushmerick

Este método proposto por Kushmerick [24] tem como finalidade remover apenas a publicidade das páginas Web. Não se adequa totalmente a este projeto, pois de qualquer forma irá deixar ativos todos os outros blocos sem conteúdo.

Resumidamente este método começa por pegar no dataset de treino para ser usado de forma a gerar uma série de regras para remover publicidade. Por fim acaba por analisar as páginas Web com as regras aprendidas, removendo assim toda a publicidade destas páginas tal como demonstra a Figura 3.5.

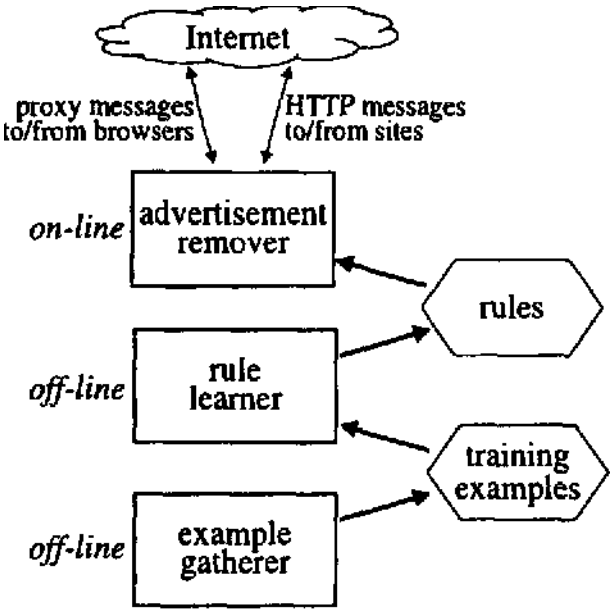

Figura 3.5: Arquitetura AdEater

Este método tem o problema de remover por vezes algumas imagens de navegação do artigo, isso dá-se pelo facto de não ter havido grandes datasets de treino para esta categoria de imagens, algo que seria facilmente contornável, no entanto, este método apresenta grandes resultados de precisão no que toca ao objetivo do algoritmo.

3.5 Deteção de Templates

Um template pode ser definido como um estilo de apresentação de página com vários espaços onde pode ser inserido conteúdo. Os conteúdos repetidos por muitas páginas Web que são baseadas no mesmo template são as fotos de capa, os menus de navegação, cabeçalho e rodapé.

Na deteção de templates pretende-se extrair a estrutura do conteúdo de uma coleção de páginas que são baseadas num template. Uma das tarefas mais complicadas é propor um algoritmo com bons resultados para vários templates. Nesta secção é descrito o InfoDiscoverer [25], Content Extractor, Feature Extractor [2] e o Método de Bar-Yossef e Rajagopalan [26].

3.5.1 InfoDiscoverer

Este algoritmo, desenvolvido por Lin e Ho [25], visa separar o conteúdo relevante do conteúdo ambíguo.

Primeiramente são extraídos os blocos das páginas baseando-se em percorrer a estrutura HTML separando todos os nós internos como blocos a partir da tag <TABLE>. Cada bloco poderá ser um bloco com conteúdo, pois estes terão uma ou mais strings nos seus últimos nós da árvore, strings estas que poderão ser relevantes para as páginas em questão como pode ser visto na Figura 3.6. Após esta primeira fase, é necessário extrair as características de cada um dos blocos. Neste algoritmo uma característica corresponde a uma palavra chave significativa. Estas palavras são obtidas após a remoção das ”Stop Words”sendo de seguida aplicado o Algoritmo Porter Stemming [27].

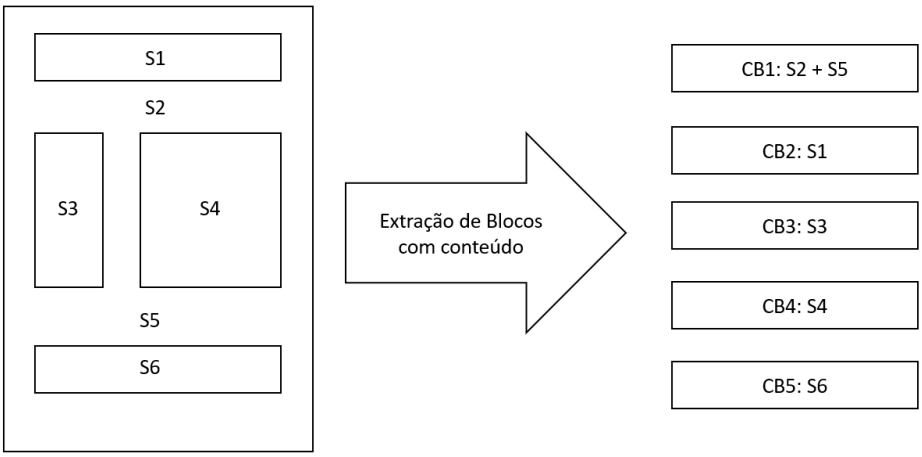

Figura 3.6: Extração de blocos com conteúdo com strings

Após este procedimento é necessário calcular os valores de entropia de cada uma destas características de acordo com a sua distribuição, estas características ficarão agrupadas numa lista com a Frequência da Palavra (TF) e com o seu peso associado. Considerando este grupo de características, pode então ser construída uma matriz de características do documento. O valor de entropia de cada

uma das características pode ser calculado através da probabilidade de estar contida numa linha da matriz.

Seguidamente é efetuado o cálculo do valor de entropia do bloco em questão através da soma dos valores de entropia das características desse bloco como mostra a seguinte equação:

$$H(CBi) = \frac{\sum_{j=1}^{k}(H(Fj))}{k} \tag{3.2}$$

Onde F_j corresponde à característica de um certo bloco com conteúdo (CBi) que contém k características. Assim sendo, o valor de entropia de cada bloco é a média do valor de entropia de todas as suas características. Este valor de entropia de cada bloco serve para identificá-los como informativos ou como redundantes mediante um certo limite. Caso o valor de entropia seja superior ao limite ou perto de 1 então esse bloco é considerado como redundante, pois significa que estas características aparecem em várias páginas, já caso o valor de entropia seja inferior ao limite então o bloco é considerado como informativo.

Este algoritmo apresenta resultados bastante eficientes no que toca à descoberta de conteúdos irrelevantes na página e quanto à extração de conteúdo informativo da página (por exemplo: noticia), no entanto, este sistema tem o problema de assumir antecipadamente como deve separar a página em blocos coerentes assumindo que alguns deles são os mesmos blocos, mas em páginas diferentes pelo que este algoritmo não funciona ao nível do bloco, mas sim ao nível das características.

3.5.2 Content Extractor Algorithm

Este algoritmo [2] pretende identificar os blocos com conteúdo de uma coleção de páginas da mesma classe. Primeiramente, começa por separar todo o documento ou página Web em blocos, para tal é dado como entrada uma lista de páginas Web e ainda uma lista ordenada de etiquetas separando assim estas páginas numa lista de blocos e sub-blocos com base nas etiquetas da lista.

Com todos os blocos identificados precisa-se então de começar a separar estes mesmos blocos em blocos sem conteúdo e com conteúdo. Os que ocorrem com maior frequência nos vários documentos são considerados redundantes por isso classificados como blocos sem conteúdo, ou seja, as secções de pesquisa, de publicidade, de imagens entre outros, pois acabam por ser muito idênticos em várias páginas Web, já os que aparecem de forma isolada são por isso considerados como blocos com conteúdo, pois o texto que aparece será certamente exclusivo da página que é processada. Posto isto, o critério usado para a distinção entre blocos com conteúdo e blocos sem conteúdo é o Inverse Block Document Frequency (IBDF) que vê a frequência de uso de cada bloco entre documentos/páginas Web e faz o inverso da mesma, ou seja, quanto mais frequente for um bloco menos conteúdo importante terá e quanto menos frequente for mais importante será.

Para medir esta similaridade são usados vetores de características, ou seja, cada bloco terá um vetor associado em que cada entrada do vetor corresponde a uma característica, como, por exemplo, o número de imagens, o número de termos, o número de scripts, entre outras. Para o conteúdo do bloco é adicionado um vetor binário em que cada entrada do vetor é associado a uma palavra da coleção de páginas. Caso a palavra correspondente exista no bloco então o valor dessa entrada no vetor será 1, caso contrário será 0. [28]

Posteriormente os blocos são comparados entre si com base nos seus vetores de características a partir da similaridade do cosseno e é definido um limite, caso o limite de similaridade seja ultrapassado então o valor de IBDF do bloco diminui.

À medida que se for a percorrer os blocos, compara-se o seu valor de IBDF com o limite pré-definido, caso o valor esteja acima do limite então o bloco é considerado um bloco com conteúdo, caso contrário será considerado como um bloco sem conteúdo sendo por isso descartado.

Assim desta forma chega-se ao fim do algoritmo "Content-Extractor"com os blocos devidamente separados entre blocos relevantes e blocos irrelevantes para o corpo da página Web. O pseudo código deste algoritmo pode ser visto na Figura 3.7.

Este algoritmo produz excelentes valores de recall, precision e ainda uma grande eficiência em tempo de execução detetando assim os blocos redundantes baseando-se na ocorrência destes blocos em várias páginas Web, no entanto, se houver páginas Web com o mesmo estilo, mas com conteúdos diferentes então este algoritmo não será capaz de os detetar.

3.5.3 Feature Extractor Algorithm

Inicialmente o algoritmo Feature Extractor [2] recebe como input uma coleção de páginas Web em formato de código HTML, uma lista de etiquetas ordenada e ainda a característica desejada para a extração da informação. Primeiramente é extraído das páginas todos os blocos existentes nas mesmas formando assim uma lista de blocos.

Seguidamente percorre-se a lista dos blocos extraídos e é calculada a probabilidade da característica desejada estar em cada um dos blocos, se essa probabilidade for superior a um certo limite então esse bloco será um bloco com conteúdo, caso contrário será um bloco sem conteúdo.

Por fim é calculada a probabilidade de cada bloco conter a característica desejada com base na lista de etiquetas, onde o bloco com a maior probabilidade é considerado o bloco vencedor e como tal terá o seu conteúdo exibido. O pseudo código deste algoritmo pode ser consultado na Fig. 3.8

O algoritmo Feature Extractor obtém melhores resultados em comparação com muitos outros algoritmos (alguns deles aqui mencionados), sendo que, a grande vantagem deste algoritmo é o facto de incorporar processos pouco complexos de machine learning. Por outro lado, este método apresenta pior desempenho nas métricas calculadas quando é usado em páginas de notícias.

Algorithm 1: *ContentExtractor algorithm and GetBlockSet function. GetBlockSet function is also used by FeatureExtractor algorithm*

Input : Set $\mathcal{S}$ of HTML pages, Sorted tag-set $\mathcal{T}$
Output: Primary Content Blocks and their associated pages in $\mathcal{S}$

begin
- $\mathcal{M}_{BD} \longleftarrow \emptyset$
- { Here the $\mathcal{M}_{BD}$ matrix is the block-document matrix where rows represent document and columns represent block identifier.}
- **for** *each* $H^k \in \mathcal{S}$ **do**
  - { Here B^k represents the kth row of the $\mathcal{M}_{BD}$ matrix. }
  - $B^k \longleftarrow$ **GetBlockSet**$(H^k, \mathcal{T})$
  - $\mathcal{M}^k_{BD} \longleftarrow B^k$
- **for** *each* $b_{ij} \in \mathcal{M}_{BD}$ **do**
  - $IBDF_{ij} \longleftarrow 1$
  - **for** *each* $b_{kl} \in \mathcal{M}_{BD}$ **do**
    - { Here $i \neq k$.}
    - $sim_{ijkl} \longleftarrow sim(b_{ij}, b_{kl})$
    - **if** $sim_{ijkl} > \epsilon$ **then**
      - $IBDF_{ij} \longleftarrow Update(IBDF_{ij})$
      - {Update Recalculates IBDF}
- { If IBDF value above threshold we will produce the output }
- **for** *each* $b_{ij} \in \mathcal{M}_{BD}$ **do**
  - **if** $IBDF_i > \theta$ **then**
    - Output the content of the block

end

Function: GetBlockSet
Input : HTML page H, Sorted tag-set $\mathcal{T}$
Output: Set of Blocks in H

begin
- $B \longleftarrow H$; // set of blocks, initially set to H.
- $f \longleftarrow Next(\mathcal{T})$
- **while** $f \neq \emptyset$ **do**
  - $b \longleftarrow First(B)$
  - **while** $b \neq \emptyset$ **do**
    - **if** b *contains* f **then**
      - $B^N \longleftarrow GetBlocks(B, f)$
      - $B \longleftarrow (B - b) \cup B^N$
    - $b \longleftarrow Next(B)$
  - $f \longleftarrow Next(\mathcal{T})$

end

Figura 3.7: Pseudo Código do algoritmo ”Content Extractor”

Algorithm 2: FeatureExtractor

Input : HTML pages H, Sorted Tag Set $\mathcal{F}$, Desired Feature $\mathcal{F}_I$

Output: Content Blocks of H

Feature: Feature set $\mathcal{F}_S$ used for block separation sorted according to importance taken from $\mathcal{F}$

begin

$B \longleftarrow GetBlockSet(H, \mathcal{F})$

for *each* $b \in B$ **do**

$P_1 \longleftarrow Pr(\mathcal{F}_I|\mathcal{F})$

if $P_1 > 0.5$ **then**

$\mathcal{W} \longleftarrow W \cup b$

for *each* $b \in \mathcal{W}$ **do**

$P_b \longleftarrow Pr(\mathcal{F}_I|\mathcal{F}, \mathcal{W})$

{ Output: Sort $\mathcal{W}$ according to the Probability value P_b and (1) Produce the content of the Winner block }

end

Figura 3.8: Pseudo Código do algoritmo ”Feature Extractor”

3.5.4 Método de Bar-Yossef e Rajagopalan

De seguida foi visto o método de Bar-Yossef e Rajagopalan [26], que visa detetar e identificar templates de sites Web através de técnicas de Data Mining. Antes de mais, é necessário referir que um template é uma coleção de páginas que têm o mesmo visual.

Primeiramente é efetuada uma separação em áreas das páginas através do algoritmo Page Partitioning que separa as várias *pagelets*, que são áreas com uma única funcionalidade bem definida não sendo localizada mais nenhuma área da página com a mesma funcionalidade.

Posto isto, após a separação das áreas é feita a deteção dos templates através de dois algoritmos, o ”local template detection algorithm”e o ”global template detection algorithm”, o primeiro para grupos de páginas menores e o segundo para grupos maiores.

Este método não tem resultados tão bons quando é testado com outras técnicas de IR como o Clustering (Agregação) [29] e classificação [29]. Como desvantagens tem o facto de ser limitado quanto aos métodos que usa para analisar e ”limpar”as páginas usando heurísticas muito simples. Por outro lado, os algoritmos apresentados neste método são bastante consistentes em espaço e em tempo pelo que consegue correr grandes agregados de páginas.

3.6 Sumário

Este capítulo apresentou uma lista de hipóteses e abordagens diferentes à resolução do problema deste trabalho. Foi introduzido as abordagens mais comuns como a Extração de Texto, abordagens através das árvores DOM das páginas Web, através do visual da página, através de Wrappers e por fim através da Deteção de Templates sendo esta a abordagem mais aproximada da Solução proposta no capítulo seguinte.

4

Proposta de Solução

Conteúdo

Neste capítulo é apresentada a minha proposta para a resolução do problema enunciado no capítulo 1 relativamente à extração do conteúdo informativo de páginas Web de notícias. Primeiramente dou uma visão geral sobre o problema e apresento os critérios de decisão por trás da escolha tomada para a solução do problema. De seguida apresento o algoritmo proposto e todas as variantes implementadas no mesmo. Por fim apresento alguns processos relativos à implementação da solução proposta e faço ainda um resumo do capítulo.

4.1 Visão geral

No capítulo 3 foi visto que a área da extração de informação de Páginas Web está bem explorada, e, ao longo dos últimos anos tem sido cada vez mais. Desde os anos 2000 que é uma realidade muito presente e o aumento da importância da tecnologia tem sido exponencial. Como tal, o uso das páginas Web para consulta de notícias é enorme, e, como pode ser visto pelo capítulo 3 existem imensos trabalhos de pesquisa tendo já sido propostos vários algoritmos.

A solução proposta por mim, seria sempre por um algoritmo que tivesse como base uma abordagem através da deteção de Templates. Refiro isto, pois o volume de templates atualmente em páginas Web varia de 40 a 50% e tende inclusivamente a aumentar [30]. Em páginas de notícias este número é consideravelmente mais elevado o que leva a crer que os templates fazem parte da atualidade e do futuro dentro deste âmbito.

O meu maior desafio deste projeto é tentar implementar uma solução que se possa adaptar a uma grande variedade de templates sendo esta a grande dificuldade do trabalho. A solução que proponho terá como base o algoritmo Content Extractor mencionado e descrito na secção 3.5.2.

4.1.1 Algoritmo Content Extractor

Decidi escolher o algoritmo Content Extractor como base do meu trabalho porque após uma análise cuidada de todos os algoritmos na abordagem com Deteção de Templates mencionados, este era o que apresentava um maior equilíbrio tanto ao nível de resultados das métricas como também ao nível de complexidade de implementação. Um dos fatores mais importantes para a tomada de decisão foi também o facto de ser um algoritmo pouco explorado [2] em nível de métricas e heurísticas no mesmo.

Em primeiro lugar, relativamente aos resultados apresentados pelos criadores [2], foi possível verificar que este algoritmo apresenta uma consistência bastante grande de Precisão, Recall e F-Measure para vários datasets o que me deu bastantes garantias para a eficácia do mesmo.

Apesar de o algoritmo ser relativamente antigo, no mesmo artigo foi proposto mais um algoritmo com a mesma lógica de procedimento que este, o Feature Extractor, tendo inclusivamente este último apresentado melhores resultados ao nível de Precisão e só precisava de ter como entrada uma página

e não uma coleção de páginas como acontece com o Content Extractor, no entanto, o algoritmo Feature Extractor apresenta um nível de complexidade de implementação bastante elevado o que não era de todo uma possibilidade para o tempo disponível e para os testes que queria realizar.

Por estas razões, achei que o algortimo Content Extractor era o que apresentava um maior equilíbrio no seu todo, sendo por isso, o algoritmo que me dava mais garantias de um trabalho de sucesso. Posto isto, o algoritmo que proponho para a solução do trabalho é o X-CEX que será explicado na secção 4.2.

4.2 Algoritmo X-CEX

Como referi na secção 4.1, o algoritmo X-CEX proposto por mim, é baseado no Content Extractor, o meu objetivo principal com o desenvolvimento deste algoritmo é fazer uma análise mais aprofundada sobre o Content Extractor e poder assim dar mais conhecimento acerca do mesmo apresentando várias melhorias até chegar a um resultado satisfatório.

O nome X-CEX foi criado com base na raiz do mesmo, ou seja, visto que seria uma melhoria ao Content Extractor então surgiu a ideia de criar uma sigla para o nome Extended Content Extractor transformando Extended em X e Content Extractor em CEX.

O algoritmo X-CEX terá como argumentos (input) uma lista de etiquetas HTML ordenada e uma coleção de páginas Web de notícias da mesma classe. Uma classe é definida como sendo um Web Site, ou seja, este algoritmo recebe uma lista de notícias vindas do mesmo Web Site garantindo assim que tem o mesmo template ou estilo de apresentação daí um dos grandes desafios para este algoritmo é ser capaz de extrair os blocos com conteúdo para vários templates.

O resultado esperado (output) do X-CEX é uma lista de blocos considerados pelo algoritmo como blocos com conteúdo das páginas Web da classe fornecidas. É importante referir que pretendo testar o algoritmo em classes que se distingam completamente das classes testadas na proposta do algoritmo Content Extractor, assim desta forma, obtenho um método mais capaz e ainda com mais informação não ficando limitado a uma classe.

Posto isto, o X-CEX será por isso composto por 3 fases, a fase de Extração de blocos, a fase da comparação entre blocos e por fim a fase do IBDF.

4.2.1 Arquitetura

Nesta secção é apresentada a arquitetura do sistema de extração proposto por mim como pode ser visto na Figura 4.1.

Nesta arquitetura podemos observar rapidamente dois argumentos de entrada, a coleção de páginas Web e a lista de etiquetas HTML. Na execução do algoritmo X-CEX podemos observar as três fases referenciadas na secção 4.2, a extração de blocos que neste caso fazem o papel de candidatos, a

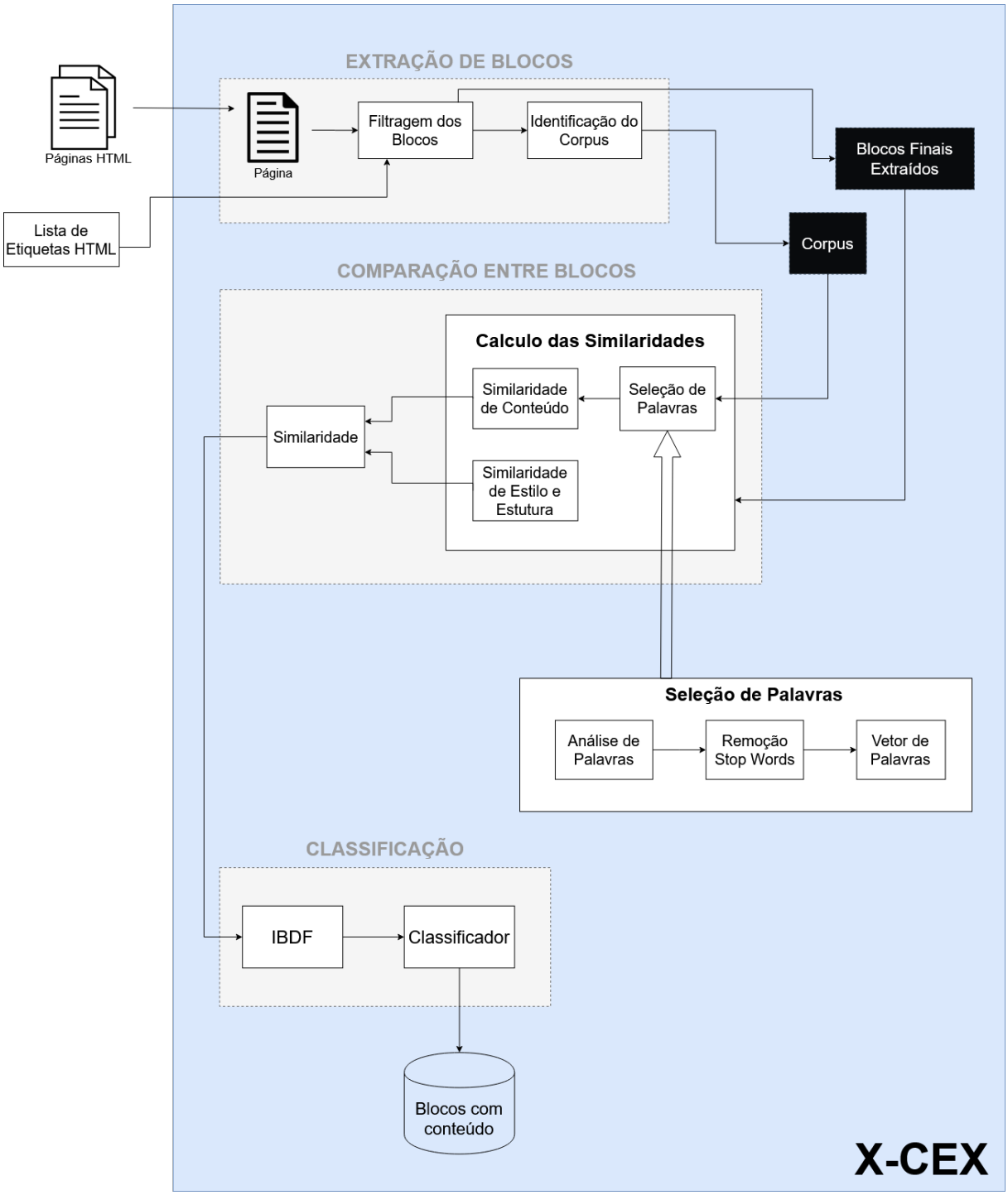

Figura 4.1: Aquitetura da solução proposta

comparação entre blocos e ainda a classificação dos blocos como blocos com conteúdo, as descrições destas fases podem ser vistas na secção 4.2.2, 4.2.3 e 4.2.4 respetivamente.

De realçar que da fase da extração de blocos obtemos os blocos extraídos pela lista de etiquetas

HTML e o corpus dos blocos, ou seja, todo o texto efetivo dos blocos extraídos. Na fase de comparação entre blocos obtemos a similaridade entre dois blocos que será usada para o IBDF do bloco comparado já na fase da classificação dos blocos.

A seleção de palavras usada no cálculo das similaridades é apenas usada para o cálculo da similaridade de conteúdo, ou seja, primeiro filtram-se as palavras do 'corpus' e então depois analisam-se as palavras filtradas nos blocos.

4.2.2 Extração de Blocos

O Algoritmo X-CEX começa primeiramente pela fase da extração de blocos (um bloco é uma secção da página começada por uma etiqueta HTML e fechada por essa mesma etiqueta HTML). Esta fase consiste em extrair uma coleção de blocos de todas as páginas fornecidas ao algoritmo.

Primeiramente o algoritmo percorre todas as páginas Web da mesma classe e, para cada uma delas é usada a lista ordenada de etiquetas HTML também recebida como argumento. Esta lista é percorrida etiqueta a etiqueta, uma por uma extraindo todos os blocos existentes na página Web com essas mesmas etiquetas obtendo assim uma coleção de blocos extraídos, sendo estes os blocos usados posteriormente para a comparação entre si.

A lista de etiquetas HTML é uma lista ordenada criada por mim com base na análise visual que fiz sobre vários templates diferentes que serve para extrair e filtrar os blocos da página a ser analisados. O algoritmo Content Extractor usava a seguinte lista de etiquetas: `<table>`, `<tr>`, `<p>`, `<hr>`, `<ul>`, `<div>`, `<span>`, no entanto, e após ponderação, decidi tomar a decisão de trocar esta lista de etiquetas para `<body>`, `<main>`, `<article>`, `<section>`, `<div>`, `<p>`.

Decidi mudar a lista porque a lista fornecida originalmente foi sugerida em 2005 e desde essa altura as estruturas HTML das páginas Web têm mudado bastante, como tal, aquela lista estava demasiado desatualizada para a realidade que encontrei.

4.2.3 Comparação entre blocos

A fase da comparação entre blocos é a fase com mais mudanças face ao algoritmo base tendo em conta que é aqui que são efetuados todos os cálculos para atribuir uma importância a cada um dos blocos.

Nesta fase, o algoritmo X-CEX, começa por percorrer todos os blocos extraídos da fase anterior descrita na secção 4.2.2 um por um comparando-os com todos os outros blocos, caso o bloco que é comparado tiver um índice de similaridade muito alto com outro, então o bloco em questão será prejudicado para a classificação final como bloco com conteúdo ou sem conteúdo.

Um dos objetivos principais da minha proposta de algoritmo era poder reformular esta fase de

comparação, pois considerei haver muitas debilidades demonstradas nesta secção, o algoritmo base misturava os dois conceitos de similaridade de estrutura/estilo e similaridade de conteúdo o que, na minha opinião, é errado visto serem duas componentes completamente diferentes uma da outra e com funcionalidades completamente distintas. O algoritmo Content Extractor ao não separar adequadamente estas duas componentes fazia com que ambas tivessem importâncias distintas de bloco para bloco não mantendo por isso uma consistência na altura da comparação. Como foi explicado na secção 3.5.2, o algoritmo Content Extractor baseava a comparação entre blocos a partir de vetores de características de cada bloco, sendo estas características o número de imagens, o número de termos, o número de scripts, entre outras, e era adicionado a este vetor um vetor binário em que cada posição correspondia a uma palavra do 'corpus' de documentos, se a palavra existisse no bloco o valor dessa entrada no vetor seria 1, caso contrário seria 0. Depois era calculada a similaridade do cosseno entre os dois vetores de características.

Este método falha, a meu ver, por duas razões, em primeiro lugar porque quantas mais características de estrutura forem usadas menos relevância terá o texto dos blocos, e como Debnath afirmou [2] na sua proposta do Content Extractor, no algoritmo base são usadas como características todas as etiquetas existentes no W3C [1]. O facto de ter imensas características relativas à estrutura do bloco faz com que a predominância no vetor de características seja de estrutura dando por isso mais relevância a essa parte o que está errado porque o objetivo é extrair os blocos com conteúdo relevante, logo, deve ser as características de texto a ter mais importância. Ainda para mais, os blocos que estão em fase de comparação já são blocos de baixo nível na árvore da página pelo que supostamente terão maioritariamente texto e não terão muitas destas características mencionadas, o que faz com que as características da estrutura do bloco sejam muito idênticas na sua maioria e assim estando esse fator a prejudicar a similaridade de ambos os blocos. Em segundo lugar, o algoritmo original falha também, pois quanto às características de estrutura, faz uma contagem do número dessas etiquetas no bloco o que provoca números diferentes de 0 e 1, algo que não acontece na parte do vetor referente ao texto, isto faz com que as discrepâncias na parte da estrutura sejam muito maiores por isso mais relevantes na comparação entre os dois vetores.

De forma a resolver estes problemas, decidi propor um sistema de importância no algoritmo X-CEX, ou seja, primeiramente separei as comparações entre estrutura/estilo do bloco e conteúdo do bloco usando assim uma métrica de similaridade para a primeira componente de estrutura do bloco e outra métrica de similaridade para comparar o conteúdo (texto) do bloco. Decidi atribuir uma importância de 30% à similaridade da estrutura e uma importância de 70% à similaridade do texto, desta forma não desvalorizo completamente a estrutura e o estilo de apresentação do bloco, mas por outro lado, dou muito mais relevância ao conteúdo escrito no mesmo. Estas percentagens foram escolhidas, por agora,

[1] http://w3c.org

de forma arbitrária, terão de ser testadas outras percentagens neste algoritmo de forma a perceber se conseguem alcançar resultados melhores.

O cálculo desta similaridade é então efetuado através da seguinte equação:

$$Sim(Bi) = (SImp * SSim) + (CImp * CSim), \tag{4.1}$$

sendo SImp a importância da estrutura, SSim a similaridade da estrutura, CImp a importância do conteúdo e CSim a similaridade do conteúdo, obtendo assim um resultado entre 0 e 1, sendo 0 o mais diferente possível e 1 o mais similar possível.

Tendo o resultado desta similaridade, é então importante definir um limite (treshold) no início do algoritmo para o resultado da mesma, caso o valor seja superior ao limite então os dois blocos são considerados como similares, caso contrário, os dois blocos são considerados como diferentes.

4.2.3.A Estrutura e Estilo

Para calcular a similaridade de estrutura e estilo entre dois blocos decidi usar a já implementada HTML-Similarity[2], esta usa a comparação de sequência de etiquetas HTML para o cálculo da similaridade de estrutura, para o cálculo da similaridade de estilo, este método extrai todas as classes CSS dos blocos aplicando depois a similaridade de Jaccard nos nomes dessas classes. O resultado da similaridade de estrutura e estilo é uma combinação destas duas similaridades tendo ambas a mesma importância.

O uso da comparação de sequência de etiquetas HTML dos blocos foi baseada na Page Compare[3], já as outras ideias deste método foram baseadas no artigo apresentado por Gowda em 2016 [31].

Decidi usar esta medida de similaridade por ser bastante atual baseando-se em artigos e propostas dos últimos anos e também por mostrar ser bastante eficaz no que toca à comparação entre dois blocos HTML. Tendo em conta que falamos de blocos HTML simples por serem blocos maioritariamente apenas de texto e de poucas etiquetas, creio fazer ainda mais sentido usar esta medida sendo ela baseada maioritariamente nas etiquetas e classes presentes dentro destes blocos.

Esta forma de calcular a similaridade é potencialmente melhor que a proposta pelo algoritmo Content Extractor visto que acaba por pegar apenas nos pontos que para mim são relevantes num bloco pequeno e simples como estes não sendo necessário ter em conta todas as etiquetas existentes como características de um bloco como foi explicado na secção 4.2.3. Desta forma o método fica muito mais simples e mais rápido na execução e com muito menos características desnecessárias.

[2] https://pypi.org/project/html-similarity/
[3] https://github.com/TeamHG-Memex/page-compare

4.2.3.B Conteúdo

Para calcular a similaridade entre dois blocos quanto ao conteúdo dos mesmos é usada a medida de similaridade do cosseno com o mesmo raciocínio do algoritmo Content Extractor, ou seja, é atribuído a cada bloco um vetor binário em que cada uma das entradas desse vetor corresponde a uma palavra, se essa palavra existir no texto do bloco então essa entrada do vetor é 1, caso contrário a entrada do vetor será 0 calculando por fim o valor do cosseno entre esses dois vetores binários.

É importante referir que as palavras tidas em conta para a criação deste vetor binário são todas as palavras existentes na coleção de blocos extraídos, como tal, todos os vetores de todos os blocos terão a mesma dimensão para dessa forma puderem ser comparados.

A similaridade do cosseno [4] é aplicada a dois vetores numa certa dimensão, o resultado da similaridade não é nada mais que o cosseno do ângulo feito pelos dois vetores, no entanto, este ângulo com vetores muito grandes e diferentes não é fácil de saber por isso pode-se usar seguinte fórmula para determinar esta similaridade:

$$Sim(A, B) = \cos(\Theta) = \frac{A \cdot B}{\|A\| \, \|B\|}, \tag{4.2}$$

sendo A e B os dois vetores a ser comparados, o numerador da fração é o produto escalar[5] entre os dois vetores e o denominador da fração é a multiplicação da distância euclididana[6] do vetor A com a do vetor B.

Posto isto, acredito que esta medida de similaridade apresenta bastante consistência já desde o algoritmo Content Extractor e como tal, não pensei que devesse ser trocado, ainda assim é da minha intenção testar outras medidas para poder comprovar a eficácia desta. Com esta medida de similaridade, se o conteúdo escrito for muito similar a outros blocos, ou seja, se grande parte do texto se repetir então significa que provavelmente esse bloco será um bloco irrelevante, pois o conteúdo do mesmo não é totalmente exclusivo a esse bloco, este facto dá assim ainda mais razão para considerar uma importância mais elevada para a similaridade de conteúdo.

4.2.4 Classificação dos blocos

Quando dois blocos são considerados como similares, ou seja, quando a similaridade é superior ao limite estabelecido será acrescentada uma página ao número de páginas com blocos similares ao do bloco que é analisado.

No fim da análise de cada um dos blocos entra em consideração o IBDF [2]. O IBDF é o inverso da frequência com que um bloco aparece nas várias páginas Web, ou seja, em termos práticos, quantas

[4] https://en.wikipedia.org/wiki/Cosine_similarity
[5] https://en.wikipedia.org/wiki/Dot_product
[6] https://en.wikipedia.org/wiki/Euclidean_distance

mais páginas tenham um bloco similar ao bloco que é comparado menos relevante será este bloco. Isto significa que se um bloco se repete/tem blocos similares em várias páginas Web então esse bloco não será um bloco com conteúdo.

Cada bloco tem um valor IBDF associado que começa a 1 no início da fase da comparação e após este ser analisado em todas as páginas 'web' e verificado em quantas delas tem um bloco similar presente então o IBDF do bloco será recalculado, sendo que, quantas mais páginas tiverem pelo menos um bloco similar a si mesmo pior será.

Posto isto, o IBDF de um bloco não é nada mais nada menos que o inverso da soma do número de páginas que têm pelo menos um bloco similar a si mesmo tal como demonstra a equação seguinte:

$$IBDF_i = f\left(\frac{1}{|S^i|}\right), \tag{4.3}$$

em que S_i é a soma do número de páginas tal como se pode ver de seguida,

$$S^i = \cup\{P_l : Sim(B_i, B_k) > \varepsilon, \forall B_k \in P_l, \forall P_l \in S\}, \tag{4.4}$$

sendo S a coleção de páginas da mesma fonte, B_i e B_k os blocos comparados e P_l a página adicionada.

Reparei serem realizadas demasiadas comparações para blocos que se repetiam, como tal, uma melhoria que propus é que, caso encontrássemos um bloco exatamente igual noutra página então este bloco ficaria automaticamente eliminado do teste colocando o seu IBDF igual a -1, desta forma evitava fazer qualquer comparação adicional para este bloco tendo já a certeza que não poderia ser um bloco com conteúdo.

É importante realçar que quanto maior for o IBDF mais relevante é o bloco, como tal, um bloco com conteúdo terá certamente muitas páginas Web fornecidas que não terão nenhum bloco similar a si mesmo, daí que no algoritmo X-CEX, se encontrarmos um bloco similar numa página essa página deixa de ser analisada e passa-se automaticamente para a próxima visto que aquela anterior já se verificou com pelo menos um bloco similar. Esta foi uma das mudanças propostas por mim neste algoritmo visto que o algoritmo Content Extractor fazia esta verificação com todos os blocos na mesma quando já não é necessário.

Outra das alterações propostas por mim neste algoritmo é efetuar esta verificação também com os blocos da página do bloco em questão, ignorando obviamente o caso em que o bloco seria comparado consigo mesmo. Tomei esta decisão, pois reparei haver alguns blocos não relevantes que só apareciam em certas páginas de forma repetida, como tal, achei importante fazer esta verificação também com os blocos da própria página.

4.2.4.A Classificador

Para poder classificar um bloco como um bloco com conteúdo é necessário definir um limite (treshold) para este valor de IBDF do bloco. Como tal, defini para este algoritmo que se um bloco tiver blocos similares em 20% das páginas então este bloco deixa de ser um bloco relevante para o contexto do problema e é classificado como bloco sem conteúdo. Por exemplo, numa coleção com 100 páginas Web, se um bloco tiver blocos similares em menos de 20 páginas então esse bloco será um bloco com conteúdo extraído pelo algoritmo.

Como o valor do IBDF não é exatamente o número de páginas, mas sim o inverso do número de páginas com blocos similares então o limite do IBDF será também esse inverso, mas aplicado a precisamente 20% do número total de páginas tal como indica a expressão seguinte:

$$Treshold - IBDF = \frac{1}{|0,2 * N|}, \tag{4.5}$$

sendo N o número de páginas totais da coleção.

O algoritmo chega assim ao fim com todos os blocos com conteúdo extraídos da coleção de páginas Web fornecidas da mesma classe.

4.3 Implementação

Para a implementação do algoritmo X-CEX, o objetivo era poder implementar o algoritmo Content Extractor e adaptá-lo e melhorá-lo com as alterações já mencionadas, no entanto, para a implementação do Content Extractor, a intenção era encontrar na Internet código aberto (open source) deste algoritmo para assim poder otimizar os processos de implementação em tempo focando-me nas melhorias. O problema é que este algoritmo não se encontra disponível em código aberto pelo que, tive de implementar todo o algoritmo de raiz a partir apenas do seu pseudo código mostrado na Figura 3.7. Este foi um dos maiores entraves se não mesmo o maior, pois provocou um retrocesso bastante grande em todo o processo de implementação.

O facto de implementar o algoritmo apenas com base no seu pseudo código é também um risco muito alto, pois podem haver várias decisões e opções tomadas por mim erradamente. Quando não se tem acesso a todos os detalhes do algoritmo base existe sempre margem para erro, o que pode provocar resultados não tão verdadeiros como os pretendidos.

4.3.1 Extração de Blocos

O objetivo desta fase de extração é poder remover o máximo de blocos irrelevantes possíveis deixando apenas os blocos mais perto de ser considerados como blocos com conteúdo e como tal, chegamos à

fase seguinte com uma lista/coleção de blocos em que cada um deles é aberto e fechado pela etiqueta <p> visto que maior parte do texto das notícias na estrutura HTML se encontra nesta. É considerado um nó folha por normalmente não conter nenhum bloco dentro de si. As duas primeiras etiquetas foram escolhidas, pois, em quase todos os templates vistos, o conteúdo informativo encontra-se na secção principal do corpo da página (<body> e <main>), a etiqueta <article> é usada de seguida por ser um filtro bastante eficaz em páginas de notícias, o conteúdo das mesmas está normalmente localizado dentro dessa etiqueta. Por fim, foi usada a etiqueta por ser a secção que normalmente aglomera todas as partes com texto da página. Posto isto, é importante realçar que, caso este trabalho seja usado no futuro para algum objetivo semelhante, esta lista poderá ser modificada em qualquer altura com base nas necessidades do criador.

4.3.2 Tokenização / Tratamento de texto

Os blocos finais propostos para avaliação são obtidos na fase de Extração, esses blocos são usados para se obter o Corpus da coleção de documentos. Este 'corpus' é o que será usado para a similaridade de conteúdo e como tal este texto precisa de ser pré-processado e passar por uma fase de tratamento de forma a remover todo o conteúdo que não é necessário.

Como tal, o algoritmo nesta fase percorre todo o 'corpus' e primeiramente são removidos todos os sinais de pontuação como pontos finais, vírgulas, pontos de interrogação e exclamação entre outros e ainda espaços vazios sem qualquer relevância, esta remoção acontece através da seguinte expressão regular: "\b[A-Za-z]+\b". De seguida e usando a biblioteca nltk do Python são removidas todas as palavras chamadas de *Stop Words* que, resumidamente, são as palavras tipicamente mais usadas apenas como ligação frásica não tendo importância real no contexto no tema.

4.3.3 Similaridades

Ainda relativamente ao cálculo da Similaridade entre dois blocos estabeleci um limite (treshold) para definir uma linha entre os blocos que eram similares um ao outro e os que não eram. Este limite foi colocado nos 50%, visto que, os 90% usados no Content Extractor faziam total sentido num sistema que não envolvia nenhuma importância às similaridades juntando inclusivamente as duas (a de Conteúdo e a Estilo/Estrutura), neste caso as importâncias fazem baixar esta similaridade. É por isso importante realçar, que este valor de limite é apenas um valor exemplo para o algoritmo, cada utilizador e cada criador usa o limite que mais lhe der jeito e que mais satisfaz as suas necessidades.

Foi implementada uma verificação adicional para caso o bloco que é analisado for comparado consigo mesmo então o algoritmo ignora e passa automaticamente à frente e evita esta comparação, assim reduzi mais uma comparação totalmente desnecessária apenas verificando o identificador do bloco.

4.3.3.A Similaridade de Estilo e Estrutura

Para medir a eficácia do HTML-Similarity decidi compará-lo com outros e, como tal, implementei mais dois métodos diferentes, um baseado no método de Griazev proposto em 2018 [32], no entanto, com algumas ligeiras mudanças, e um segundo método implementado para comparar com o proposto baseado na Similaridade do Cosseno, este método foi usado no algoritmo Content Extractor e decidi implementá-lo aqui também para poder comparar com as bases do algoritmo original. Estes dois métodos são descritos mais aprofundadamente de seguida:

- **Método de Griazev**

  Este método de Griazev foi proposto em 2018 [32] e pretende dar mais peso aos nós com um nível mais elevado na árvore por influenciarem mais o estilo e a estrutura dos blocos. Decidi usar este método por ser um método muito recente tendo apenas dois anos e, como tal, prova ser um método atual das páginas Web. Decidi fazer algumas adaptações ao método, visto que os blocos que aqui eram comparados já eram todos blocos de baixo nível na árvore e, como tal, retirei o fator do nível. Assim sendo, a equação para atribuição de uma pontuação à ação (inserção, remoção ou modificação) realizada é a seguinte:

$$L_{score} = \begin{cases} 1, & \text{se etiquetas iguais} \\ (O_c)^2 * K_L, & \text{se etiquetas diferentes} \end{cases} \tag{4.6}$$

  Posto isto, o método implementado começa por verificar se as etiquetas que estão ao mesmo nível são ou não iguais, se não, é adicionado um custo de modificação onde Oc é o custo da operação, KL é o coeficiente do nível que neste caso é apenas uma constante igual a 0,9. Quando as etiquetas são iguais, a pontuação é 1, pois ambas têm 100% de similaridade, sendo este processo realizado para todas as etiquetas presentes ao mesmo nível dos dois blocos. Na última fase do método são somadas todas as pontuações e dividido pelo número maior de etiquetas existentes apenas num dos blocos.

$$B_{sim} = \frac{\sum_N^1 L_{score}}{N}, \tag{4.7}$$

  onde N é o maior número de etiquetas presentes entre os dois blocos e Lscore é a pontuação calculada anteriormente.

- **Cosseno**

  Neste método são extraídas várias características dos blocos de acordo com as etiquetas presentes na W3C[7], ou seja, são extraídos o número de imagens através da etiqueta `<img>`, o número

[7] https://www.w3.org/

de links pela etiqueta <a>, entre outras como número de tabelas, de scripts, de parágrafos, de palavras, de artigos, figuras, botões, etc. Estas características são todas inseridas num vetor por ordem chegando assim ao vetor de características do bloco.

Por fim é usada a similaridade do cosseno entre os vetores de características dos dois blocos em comparação, esta similaridade é calculada através da biblioteca sklearn do Python. Este método peca por retirar demasiadas características de um bloco que por si só já é considerado como um bloco de baixo nível e como tal com poucas características dentro de si.

4.3.3.B Similaridades de Conteúdo

Neste caso para teste foram usadas a distância de Levenshtein, a distância de Jaro-Winkler, a distância de Damerau-Levenshtein e ainda a distância de Hamming. Estas métricas são calculadas com a biblioteca jellyfish do Python e têm a particularidade de usar o raciocínio de transformar o texto de um bloco no texto do outro bloco sendo a similaridade calculada com base nessa transformação.

A percentagem de similaridade é então calculada pela seguinte equação:

$$Sim = 1 - \left(\frac{d}{\max\left(textB1, textB2\right)} \right), \tag{4.8}$$

onde d é a distância calculada, B1 e B2 são os blocos que são comparados e, como tal, o denominador é o máximo de caracteres entre o texto dos dois blocos.

- **Distância de Levenshtein**

  A distância de Levenshtein[8] é resumidamente o número operações efetuadas de forma a transformar uma 'string' na outra, ou seja, é o número de remoções, inserções e substituições de letras necessárias para a transformação.

- **Distância de Jaro-Winkler**

  A distância de Jaro-Winkler[9] é uma medida de similaridade entre duas 'strings', baseada no prefixo das mesmas, ou seja, esta medida dá mais importância a 'strings' que contenham o seu início parecido. É mais adequada a 'strings' mais pequenas como uma ou duas palavras, não se relacionando tanto com o contexto deste trabalho.

- **Distância de Damerau-Levenshtein**

  A distância de Damerau-Levenshtein[10] é muito idêntica à distância de Levenshtein, no entanto, esta medida tem em conta a transposição de duas letras adjacentes, ou seja, caso a troca de duas

[8] https://en.wikipedia.org/wiki/Levenshtein-distance
[9] https://en.wikipedia.org/wiki/Jaro-Winkler-distance
[10] https://en.wikipedia.org/wiki/Damerau-Levenshtein-distance

letras seguidas faça corresponder à 'string' final nessas posições então é acrescentado apenas um movimento em vez de dois como seria na distância de Levenshtein.

- **Distância de Hamming**

  A distância de Hamming[11], apesar de diferente das outras, acaba por resultar no mesmo conceito. Esta corresponde ao número de posições em que o símbolo/letra diferem nas duas 'strings', ou seja, esta medida calcula o número mínimo de substituições para transformar uma 'string' na outra.

4.4 Sumário

Neste capítulo apresentei a estrutura da minha proposta para a resolução do problema do trabalho. Comecei por revelar que a base para a minha proposta é o algoritmo Content Extractor e o porquê de ter-me baseado nele.

Seguidamente, apresentei a arquitetura da minha proposta e os detalhes do algoritmo proposto, os objetivos principais do mesmo, descrevi as alterações efetuadas ao algoritmo Content Extractor e as razões das mesmas. Mostrei que o algoritmo X-CEX está constituído por três fases diferentes, a extração de blocos, a comparação entre blocos e ainda a classificação sendo a segunda fase a que mais melhorias e inovações teve.

Por fim, apresentei alguns detalhes da implementação da solução, como alguns obstáculos ultrapassados e ainda os métodos usados para os testes.

[11] https://en.wikipedia.org/wiki/Hamming-distance

5

Testes e Resultados

Conteúdo

Neste capítulo apresento os testes realizados e os resultados obtidos com o algoritmo X-CEX. Começo por dar visão geral pelos datasets recolhidos e quais as divisões realizadas, de seguida falo sobre os vários testes realizados e a razão dos mesmos sendo seguidos dos respetivos resultados. Por fim estes resultados são analisados, retirando assim algumas conclusões.

5.1 Datasets

Para testar o algoritmo proposto foi necessário extrair os datasets implementando dois scripts diferentes em Python visto que os datasets pretendidos não existiam online para poder descarregar o que dificultou imenso o processo de extração. Estes scripts usam maioritariamente a biblioteca Beautiful Soup do Python: (a) Script de Extração de Notícias e (b) Script de Extração do Conteúdo;

O primeiro script pretende extrair todas as notícias em ficheiros HTML dos Web Sites (classes) escolhidos através dos links fornecidos ao script, links estes com várias notícias incorporadas. O segundo script pretende extrair todos os blocos com conteúdo relevantes das notícias extraídas, ou seja, estes blocos não são mais que os resultados que o algoritmo proposto deveria extrair a servir por isso como verdade absoluta, como tal, estes servem para no fim poder comparar com os blocos gerados pelo algoritmo calculando nessa altura os resultados para as métricas de avaliação descritas na secção 5.2.

No primeiro script os principais obstáculos foram o facto de as notícias estarem em secções diferentes da mesma página e a remoção de notícias pagas, de notícias só com vídeos e de notícias só com galeria de fotos. Por outro lado, no segundo script, as maiores dificuldades encontradas, foram os ficheiros que não podiam ser considerados para o dataset como as páginas de acompanhamento de eventos em direto por estas não terem a mesma estrutura que as notícias normais.

É importante frisar que, os blocos com conteúdo extraídos do segundo script, podem não corresponder completamente à verdade absoluta estando assim suscetível a resultados não totalmente verdadeiros. Podem existir alguns blocos com conteúdo que este script não esteja a extrair e, como tal, pode estar a prejudicar os resultados obtidos pelo algoritmo erradamente.

Para a avaliação do algoritmo X-CEX foram recolhidos os Datasets de forma a perceber a fiabilidade do mesmo nos mais variados templates. Cada dataset é constituído por várias páginas da mesma fonte, ou neste caso, do mesmo web site. Para a avaliação deste trabalho foram recolhidas 160 páginas web de notícias para cada dataset escolhido, idealmente seriam necessárias mais páginas para testar o algoritmo, no entanto, tal como foi explicado a extração das mesmas foi realizada por mim através de scripts o que dificultou em parte a extração, o facto de haver muitas notícias que tinham de ser excluídas também dificultou o processo por isso apenas foram recolhidas estas 160 páginas que eu acredito, apesar de tudo, já serem suficientes para obter resultados fidedignos. Posto isto, o dataset é constituído por várias páginas HTML sendo este o formato dos dados tratados.

Estes datasets foram divididos em datasets de treino e datasets de teste, sendo que, a divisão foi feita com 75% das páginas para o dataset de treino e 25% das páginas para o dataset de teste, ou seja, as 160 páginas de um template foram divididas com 120 delas para dataset de treino e 40 delas para dataset de teste. Esta divisão foi importante, pois permitiu-me implementar o algoritmo sem problemas usando sempre o dataset de treino para aprimorá-lo ao máximo, com isso, mexi em certos parâmetros para obter os melhores resultados possíveis com o dataset de Teste. É importante frisar que todos os testes efetuados descritos nas secções seguintes foram realizados com o dataset de teste.

De forma a testar o algoritmo para vários templates e analisar a flexibilidade do algoritmo tive de extrair um dataset de vários web sites, ou seja, vários templates diferentes.

5.1.1 Web Sites

O algoritmo Content Extractor extraiu apenas páginas de notícias de Web Sites ingleses, no entanto, neste trabalho pretendo avaliar o algoritmo X-CEX com outras línguas como o Português, o Espanhol, o Italiano e o Francês, de forma a testar a eficácia do mesmo com línguas distintas. Assim desta forma não só são testados vários templates diferentes como também várias línguas, o que, transmite mais segurança no trabalho realizado.

Foram escolhidos 7 Web Sites para serem analisados com o algoritmo X-CEX, estes são, os jornais portugueses A Bola, Record e Diário de Notícias, os espanhóis Marca e AS, o L'Équipe que é o único francês e por fim como único italiano escolhi o Corriere dello Sport. Foram extraídas 160 páginas HTML de cada um destes Web Sites, portanto, no total, tenho 1120 páginas divididas em 7 templates diferentes.

5.2 Métricas de Avaliação

As métricas usadas para avaliar e comparar este trabalho foram a Precisão, o Recall, a F-measure e a Accuracy. Em primeiro lugar, convém explicar que neste contexto os valores de precisão e recall são baseados nos 'items' considerados como Positivo Verdadeiro (tp), Positivo Falso (fp), Negativo Verdadeiro (tn) e Negativo Falso (fn), sendo que, na prática, estes grupos pretendem dividir e mostrar os 'items' que foram avaliados corretamente e os 'items' que não foram avaliados corretamente. O grupo dos positivos representa os blocos extraídos pelo nosso trabalho como relevantes, podendo ter sido bem ou mal extraídos (Verdadeiro ou Falso), o mesmo raciocínio é aplicado aos negativos sendo estes os blocos que foram considerados como irrelevantes pelo trabalho desenvolvido, também estes divididos em bem ou mal (Verdadeiro ou Falso).

Tal como mostra na Figura 5.1, a Precisão é definida como sendo o rácio dos 'items' tp pelos tp + fp, ou seja, a Precisão pretende mostrar a percentagem dos 'items' selecionados que são realmente rele-

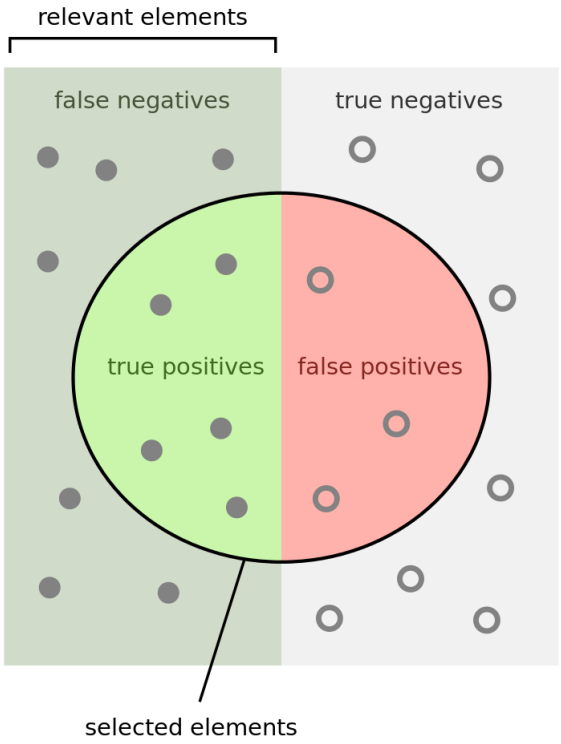

Figura 5.1: Definição de Precisão e Recall

vantes. Já o Recall que é definido pelo rácio dos 'items' tp pelos tp + fn pretende calcular a percentagem dos 'items' relevantes que foram considerados como tal, ou seja, a taxa de acerto relativa aos relevantes. No contexto deste problema os 'items' são considerados como blocos, sendo que, as métricas de avaliação pretendem avaliar os blocos com conteúdo e os blocos sem conteúdo considerados como tal correta ou incorretamente.

Assim sendo, a Precisão calcula o rácio do número de blocos com conteúdo, corretamente considerados (r) pelo número total dos blocos com conteúdo sugeridos pelo algoritmo (t):

$$Precisão = \frac{r}{t} \tag{5.1}$$

O Recall calcula o rácio do número de blocos com conteúdo, corretamente considerados (r) pelo número total dos blocos com conteúdo desejados, estes últimos incluem os blocos com conteúdo, corretamente considerados (r) e os blocos com conteúdo que o algoritmo não considerou (m):

$$Recall = \frac{r}{r+m} \tag{5.2}$$

A F-measure, de forma similar à proposta por Van Rijsbergen [33], serve para calcular a média harmónica da Precision e do Recall indo desde 0 (pior caso) até 1 (melhor caso), para tal, apenas é

necessário ter em conta os valores calculados pela Precisão e pelo Recall. Esta calcula-se da seguinte maneira:

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.3}$$

Por último, a Accuracy [34] pretende calcular a exatidão com que se chega ao resultado sem detetar erros, ou seja, é a percentagem de eficácia com que se chega aos valores reais. Isto é, a accuracy nada mais é que o rácio entre os blocos com conteúdo e sem conteúdo, obtidos corretamente (r + p) por todos os blocos considerados para avaliação (nBlocos).

$$Accuracy = \frac{r + p}{nBlocos} \tag{5.4}$$

Para além das métricas de Precision, Recall, F-measure e Accuracy, são também alvo de avaliação o algoritmo em tempo de execução. Para este, basta ter um contador de tempo desde o início da execução do algoritmo até ao fim, de forma a obter o tempo necessário para a execução completa do algoritmo.

5.3 Testes Realizados e Resultados

Para avaliar a solução proposta foram realizados vários testes de estrutura do algoritmo, ou seja, o objetivo com os testes realizados é poder avaliar as inovações, as métricas escolhidas e os métodos propostos para a solução do problema. Dessa forma consigo perceber se tomei as decisões certas ou não no decorrer do trabalho. Todos os resultados obtidos para a Precision, Recall, F-Measure e Accuracy são arredondados a 2 casas decimais. O tempo é apresentado em horas:minutos:segundos, sendo que, quando o tempo fica abaixo de 1 hora, o tempo é apresentado em "minutos:segundos".

Posto isto, primeiramente testei o algoritmo X-CEX comparando-o com o algoritmo Content Extractor que serviu de base para o algoritmo proposto. Estes resultados podem ser vistos nas tabelas 5.1 e 5.2 respetivamente.

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	**0,99**	0,62	0,46	00:37
AS	523	0,92	**0,95**	0,93	**0,98**	01:40
Corriere dello Sport	81	**1,00**	0,67	0,80	**1,00**	00:06
Diário de Notícias	811	0,92	**0,99**	**0,96**	0,93	11:32
L'Équipe	246	**0,98**	0,73	0,83	**0,98**	00:47
Marca	387	0,71	**1,00**	0,83	0,71	02:53
Record	431	0,74	0,93	0,83	0,86	01:58

Tabela 5.1: Resultados Algoritmo X-CEX

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,92	0,61	0,50	00:25
AS	616	0,43	0,53	0,48	0,83	02:08
Corriere dello Sport	211	0,90	0,84	0,87	0,95	00:19
Diário de Notícias	811	0,93	0,87	0,90	0,94	11:09
L'Équipe	246	0,98	0,81	0,89	0,98	00:35
Marca	503	0,66	0,90	0,76	0,75	02:51
Record	509	0,58	0,64	0,61	0,83	01:44

Tabela 5.2: Resultados Algoritmo Content Extractor

De seguida, testei o algoritmo X-CEX com duas listas de etiquetas diferentes da proposta, estas duas listas são usadas para efeitos comparativos, sendo a Lista 1 uma variante da lista de etiquetas proposta adicionando apenas a etiqueta `<section>` e a Lista 2 é a escolhida pelo algoritmo Content Extractor:

- Lista 1: `<body>`, `<main>`, `<article>`, `<section>`, `<div>`, `<p>`
- Lista 2: `<table>`, `<tr>`, `<p>`, `<hr>`, `<ul>`, `<div>`, `<span>`

Estas listas foram testadas para todos os datasets escolhidos, os resultados para a Lista 1 são apresentados na Tabela 5.3 e os resultados para a Lista 2 são apresentados na Tabela 5.4. É importante frisar que estes testes foram realizados com todos os métodos propostos no algoritmo X-CEX mudando apenas as listas de etiquetas HTML, desta forma consigo perceber se a lista escolhida é a que obtém melhores resultados com os métodos usados no algoritmo.

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:34
AS	523	0,92	0,95	0,93	0,98	01:36
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:06
Diário de Notícias	811	0,92	0,99	0,96	0,93	11:29
L'Equipe	5	1,00	0,02	0,04	1,00	00:04
Marca	376	0,71	0,97	0,82	0,71	02:35
Record	251	0,16	0,05	0,07	0,80	00:18

Tabela 5.3: Resultados Lista 1

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:30
AS	616	0,47	0,78	0,59	0,78	02:58
Corriere dello Sport	211	0,92	0,99	0,95	0,95	00:25
Diário de Notícias	811	0,92	0,99	0,96	0,93	11:39
L'Equipe	246	0,98	0,73	0,83	0,98	00:45
Marca	503	0,66	0,90	0,77	0,76	03:36
Record	509	0,61	0,69	0,64	0,83	01:52

Tabela 5.4: Resultados Lista 2

Outra das inovações implementadas, foi colocar o valor de IBDF dos blocos a -1 quando estes têm blocos exatamente iguais no dataset fornecido, ficando automaticamente como bloco sem conteúdo relevante, como tal, decidi testar esta atribuição executando o algoritmo sem esta condicionante. Os resultados obtidos podem ser vistos na Tabela 5.5.

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:34
AS	523	0,43	1,00	0,60	0,61	04:02
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:06
Diário de Notícias	811	0,86	1,00	0,93	0,86	14:50
L'Equipe	246	0,98	0,96	0,97	0,98	00:59
Marca	387	0,70	1,00	0,82	0,70	02:55
Record	431	0,41	0,93	0,57	0,41	03:06

Tabela 5.5: Resultados sem IBDF=-1

Para o limite do IBDF escolhido, foi necessário realizar vários testes de forma a perceber qual dos limites iria obter melhores resultados. É importante frisar que este limite tem como base o número de páginas com blocos similares ao bloco analisado, por isso, os vários limites testados têm única e exclusivamente a ver, com a percentagem de páginas que têm um bloco similar a si.

Os vários limites usados para os testes correspondem a 15%, 25%, 30% e 35% das páginas terem um bloco similar ao bloco que é analisado. De realçar que a percentagem implementada no algoritmo X-CEX foi a de 20%. Estes resultados podem ser visto nas Tabelas 5.6 e 5.7.

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:41
AS	523	0,92	0,95	0,93	0,98	01:39
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:08
Diário de Notícias	811	0,93	0,99	0,96	0,94	13:47
L'Equipe	246	0,98	0,66	0,79	0,98	00:52
Marca	387	0,71	1,00	0,83	0,71	02:52
Record	431	0,74	0,93	0,83	0,86	01:58

Tabela 5.6: Resultados para 15% das páginas como limite

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:40
AS	523	0,92	0,95	0,93	0,98	01:41
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:06
Diário de Notícias	811	0,92	0,99	0,96	0,93	13:46
L'Equipe	246	0,98	0,75	0,85	0,98	00:46
Marca	387	0,71	1,00	0,83	0,71	02:49
Record	431	0,74	0,93	0,83	0,86	01:57

Tabela 5.7: Resultados para 25%, 30% e 35% das páginas como limite

Como os resultados de tempo de execução entre os testes com 25%, 30% e 35% apenas alteravam

pequenos segundos irrelevantes e mantinham todos os outros valores iguais, então, decidi incluir na tabela 5.7 os 3 testes juntos, sendo que, na coluna do tempo, os resultados apresentados são os do teste com 25% das páginas.

A maior inovação e melhoria implementada foi, efetivamente, a alteração do cálculo das similaridades para um sistema de importância, em que, é dada uma maior relevância à comparação do conteúdo do bloco em vez da comparação da estrutura e estilo do mesmo, o que, na minha opinião, prejudicava o anterior algoritmo. Foram efetuados vários testes com diferentes importâncias, de forma a perceber se a importância escolhida foi ou não a decisão mais acertada.

Assim sendo, foram efetuados testes ao algoritmo X-CEX com 50%, 55%, 60%, 75% e 80% de importância para o conteúdo dos blocos, estes resultados podem ser vistos nas Tabelas 5.8, 5.9, 5.10, 5.11 e 5.12.

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,51	0,29	0,37	0,87	00:28
AS	523	0,77	0,28	0,41	0,98	01:36
Corriere dello Sport	81	1,00	0,23	0,38	1,00	00:06
Diário de Notícias	811	0,83	0,13	0,22	0,98	07:48
L'Equipe	246	0,95	0,30	0,45	0,98	00:44
Marca	387	0,46	0,34	0,39	0,73	02:40
Record	431	0,50	0,27	0,35	0,88	02:08

Tabela 5.8: Resultados para 50% de importância para o conteúdo

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,48	0,88	0,63	0,57	00:35
AS	523	0,92	0,91	0,91	0,98	01:45
Corriere dello Sport	81	1,00	0,66	0,80	1,00	00:07
Diário de Notícias	811	0,93	0,75	0,83	0,95	12:02
L'Equipe	246	0,98	0,82	0,89	0,98	00:52
Marca	387	0,70	1,00	0,83	0,71	03:06
Record	431	0,76	0,87	0,81	0,88	02:23

Tabela 5.9: Resultados para 55% de importância para o conteúdo

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,63	0,46	00:35
AS	523	0,92	0,95	0,93	0,98	01:43
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:06
Diário de Notícias	811	0,93	1,00	0,96	0,93	11:52
L'Equipe	246	0,98	0,77	0,86	0,98	00:49
Marca	387	0,70	1,00	0,83	0,71	03:02
Record	431	0,73	0,93	0,82	0,85	02:14

Tabela 5.10: Resultados para 60% de importância para o conteúdo

Para finalizar os testes realizados, foram ainda testadas todas as métricas implementadas descritas

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:38
AS	523	0,92	0,95	0,93	0,98	01:49
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:08
Diário de Notícias	811	0,92	0,99	0,96	0,93	12:23
L'Equipe	246	0,98	0,73	0,83	0,98	00:53
Marca	387	0,71	1,00	0,83	0,71	03:13
Record	431	0,74	0,93	0,83	0,86	02:04

Tabela 5.11: Resultados para 75% de importância para o conteúdo

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:38
AS	523	0,92	0,95	0,93	0,98	01:49
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:06
Diário de Notícias	811	0,92	0,99	0,96	0,93	12:09
L'Equipe	246	0,98	0,71	0,83	0,98	00:49
Marca	387	0,71	1,00	0,83	0,71	03:08
Record	431	0,74	0,93	0,83	0,86	02:10

Tabela 5.12: Resultados para 80% de importância para o conteúdo

na secção 4.3.3, de forma, a obter os resultados de Precision, Recall, F-Measure e Accuracy juntando ainda o tempo de execução. Estes testes servem para perceber, se as métricas escolhidas para o cálculo das similaridades no algoritmo X-CEX, foram as métricas mais acertadas.

Primeiro foram testadas as métricas do Cosseno e do Método de Griazev para a similaridade de estilo e estrutura. Estes resultados podem ser consultados já de seguida nas Tabelas 5.13 e 5.14.

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:31
AS	523	0,92	0,96	0,94	0,98	01:30
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:05
Diário de Notícias	811	0,92	0,99	0,96	0,93	10:41
L'Equipe	246	0,98	0,74	0,84	0,98	00:37
Marca	387	0,71	1,00	0,83	0,71	02:24
Record	431	0,74	0,92	0,82	0,86	01:42

Tabela 5.13: Resultados para testes com método Cosseno

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:25
AS	523	0,92	0,95	0,94	0,98	01:02
Corriere dello Sport	81	1,00	0,67	0,80	1,00	00:04
Diário de Notícias	811	0,92	0,99	0,96	0,93	10:03
L'Equipe	246	0,98	0,74	0,84	0,98	00:31
Marca	387	0,71	1,00	0,83	0,71	01:59
Record	431	0,74	0,92	0,82	0,86	01:24

Tabela 5.14: Resultados para testes com o Método de Griazev

Os resultados dos testes com as distâncias de Levenshtein, Damerau-Levenshtein e Hamming relativas à similaridade de conteúdo são apresentados nas Tabelas 5.15, 5.16 e 5.17 respetivamente. A similaridade de conteúdo através da distância de Jaro-Winkler não obteve nenhum resultado para nenhum dos Web Sites, pelo que, não são apresentados os resultados com essa métrica.

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,63	0,47	19:04
AS	523	0,93	0,95	0,94	0,98	52:17
Corriere	81	1,00	0,67	0,80	1,00	03:05
Diário de Notícias	811	0,90	0,69	0,78	0,94	05:57:22
L'Équipe	246	0,98	0,79	0,87	0,98	23:35
Marca	387	0,69	0,94	0,80	0,71	01:30:57
Record	431	0,73	0,76	0,75	0,88	01:02:48

Tabela 5.15: Resultados para testes com a distância de Levenshtein

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,63	0,47	36:39
AS	523	0,92	0,94	0,93	0,98	01:39:23
Corriere	81	1,00	0,67	0,80	1,00	05:43
Diário de Notícias	811	0,90	0,65	0,75	0,94	11:27:14
L'Équipe	246	0,98	0,78	0,87	0,98	48:26
Marca	387	0,69	0,93	0,79	0,71	02:53:47
Record	431	0,73	0,73	0,73	0,88	01:40:14

Tabela 5.16: Resultados para testes com a distância de Damerau-Levenshtein

Sites	Blocos	Precision	Recall	F-Measure	Accuracy	Tempo
A Bola	184	0,46	0,99	0,62	0,46	00:17
AS	523	0,93	0,95	0,94	0,98	00:41
Corriere	81	1,00	0,67	0,80	1,00	00:03
Diário de Notícias	811	0,92	0,99	0,96	0,93	03:28
L'Équipe	246	0,98	0,74	0,85	0,98	00:23
Marca	387	0,71	1,00	0,83	0,71	01:01
Record	431	0,74	0,93	0,83	0,86	00:46

Tabela 5.17: Resultados para testes com a distância de Hamming

Por fim, gostava também de ter testado o algoritmo para texto extraído, e não só para blocos extraídos, no entanto, o tempo necessário para implementar esta alternativa era demasiado e não cabia no espaço de tempo que tinha para a resolução do trabalho ficando como sugestão para trabalho futuro.

5.4 Análise de Resultados

Nesta secção, são analisados ao pormenor todos os resultados dos testes efetuados e descritos na secção 5.3.

5.4.1 Algoritmo X-CEX

Os resultados obtidos através da execução do algoritmo X-CEX são apresentados na tabela 5.1. Através dos mesmos consigo concluir que, o site A Bola é o que apresenta piores resultados nesta execução em Precisão, F-Measure e Accuracy. Estes resultados podem ser justificados pelos comentários das notícias em questão, visto que, após análise, reparei serem considerados como blocos com conteúdo a maior parte dos comentários realizados pelos leitores destas notícias.

Por outro lado, podemos observar que, o algoritmo funciona muito bem para os templates do AS e do Diário de Notícias obtendo valores de Precisão, Recall, F-Measure e Accuracy acima dos 90% o que mostra ser bastante fiável nestes mesmos Web Sites. A Marca e o Record acabam por representar um excelente Recall, o que significa, obter quase todos os blocos com conteúdo das notícias em questão, no entanto, acabam por incluir muitos blocos irrelevantes baixando assim o valor de Precisão. Já o L'Équipe e o Corriere dello Sport, mostram-se em sentido inverso obtendo uma excelente Precisão e baixo Recall, significando que, praticamente todos os blocos extraídos são relevantes, no entanto, ficam a faltar identificar alguns outros. Em ambos os sites, estes resultados podem ser justificados pela sua fase de extração de blocos que também foi testada e analisada na secção seguinte.

É também importante referir o excelente tempo de execução do algoritmo, onde apenas se destaca negativamente neste campo o site do Diário de Notícias. Pode-se justificar este aumento de tempo pelo facto de os blocos de texto deste template conterem na sua maioria muitos sub-blocos (como, por exemplo, negrito, citações, itálicos, etc.), o que os torna mais complexos para as comparações efetuadas no algoritmo.

Podemos ainda observar que, a Accuracy mostra estar em bastante sintonia com a Precisão, isto justifica-se pelo facto da Accuracy basear-se no número de blocos extraídos corretamente (tanto com conteúdo como sem conteúdo), já a Precisão, apenas se baseia nos blocos com conteúdo extraídos de forma correta, o que mostra que tanto os com conteúdo como os sem conteúdo são gerados pelo algoritmo de forma bastante proporcional face ao esperado.

No geral, penso que, consegui obter aqui um excelente algoritmo capaz de extrair os conteúdos relevantes para vários templates diferentes, sendo que, excluindo o site A Bola, o pior resultado de F-Measure é de 80% para o site Corriere dello Sport, o que, mostra haver um grande equilíbrio neste algoritmo entre Precisão e Recall. Relativamente às línguas usadas nestes templates, não parece haver grande relação entre elas no que toca aos resultados, pelo que, parece ser uma componente já resolvida anteriormente pelo algoritmo Content Extractor.

Comparando o X-CEX com o algoritmo base, o Content Extractor, podemos observar que existe uma melhoria clara para o AS e para o Record tal como mostra as Figuras 5.2 e 5.3.

Sendo que, quanto mais juntas ficam as quatro métricas apresentadas nas imagens, mais coesos são os resultados, adicionalmente se estas melhorarem os seus resultados, como acontece neste caso

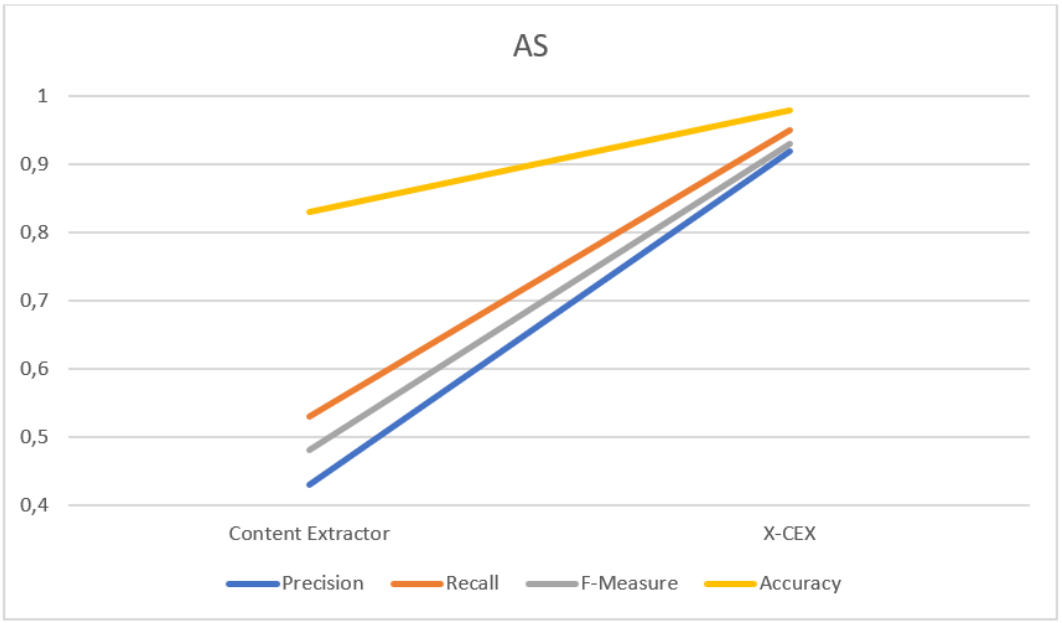

Figura 5.2: Comparação X-CEX e Content Extractor para o AS

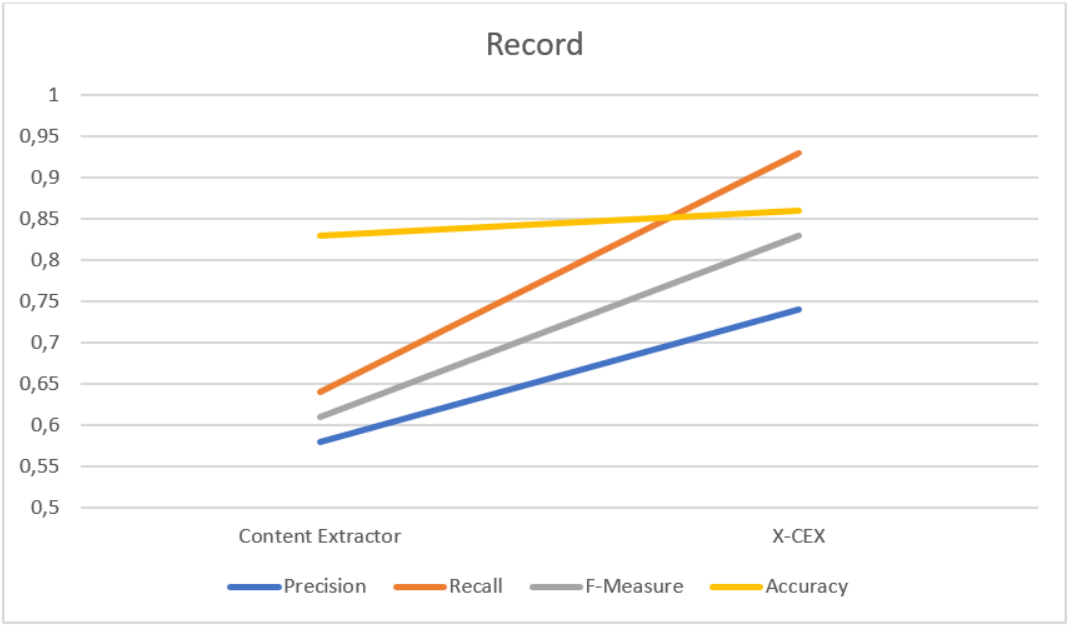

Figura 5.3: Comparação X-CEX e Content Extractor para o Record

melhor ainda. Por outro lado, observamos que para o Corriere dello Sport acontece exatamente o inverso, mostrando que o algoritmo X-CEX acaba por piorar os resultados para este template apesar de, ainda assim melhorar a Precisão e a Accuracy como pode ser visto na Figura 5.4.

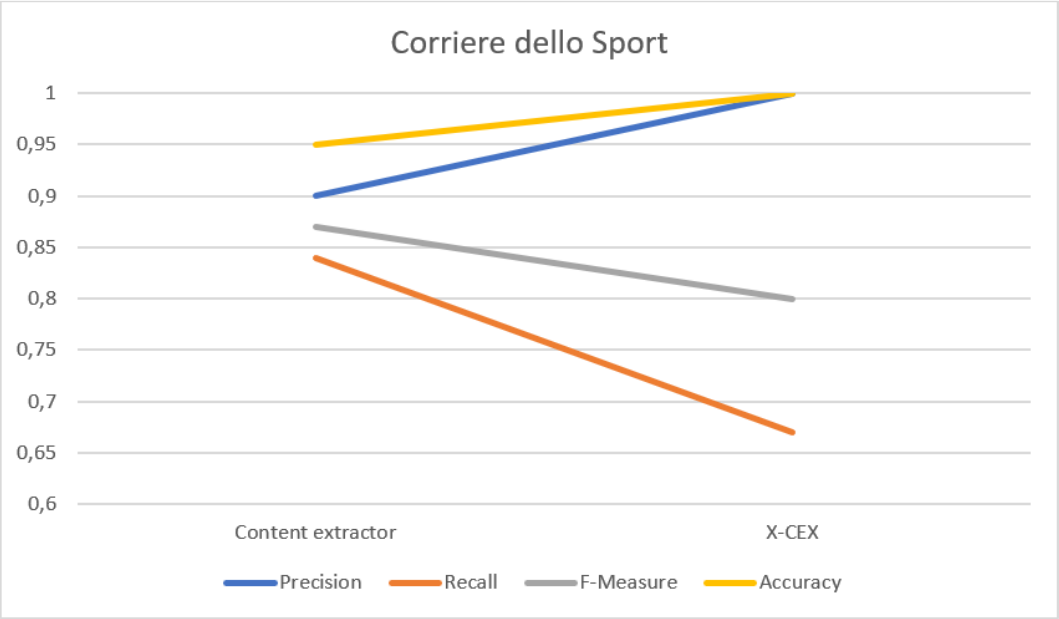

Figura 5.4: Comparação X-CEX e Content Extractor para o Corriere dello Sport

No geral, existe maioritariamente um aumento ao nível de resultados de Precisão e Recall, podemos ainda observar que, existe um decréscimo claro no número de blocos extraídos na maioria dos Web Sites. Este número, é reflexo da fase de extração de blocos, e é explicado na análise aos resultados do próximo teste apresentada na secção 5.4.2.

5.4.2 Testes para Lista de Etiquetas

Tendo em conta os resultados obtidos no teste anterior, posso já antever que, os testes para a lista de etiquetas são dos mais importantes para o algoritmo X-CEX, pois existem lacunas na fase de extração de blocos. Se mudarmos um pouco a lista para extração, conseguimos alterar muito os resultados, sendo por isso um excelente teste para analisar a capacidade de progressão e melhoria que esta fase de extração pode ter.

Podemos concluir que, a partir dos resultados das tabelas 5.1, 5.3 e 5.4, observamos um aumento de blocos entre o algoritmo X-CEX e a versão com a Lista 2, como se vê na Figura 5.5, e como este aumento se reflete também no aumento de tempo, Figura 5.6, o que nos permite concluir que, quanto mais blocos forem extraídos para comparação mais tempo demorará.

Nestas imagens são ignorados os resultados para o Diário de Notícias, L'Équipe e A Bola por não

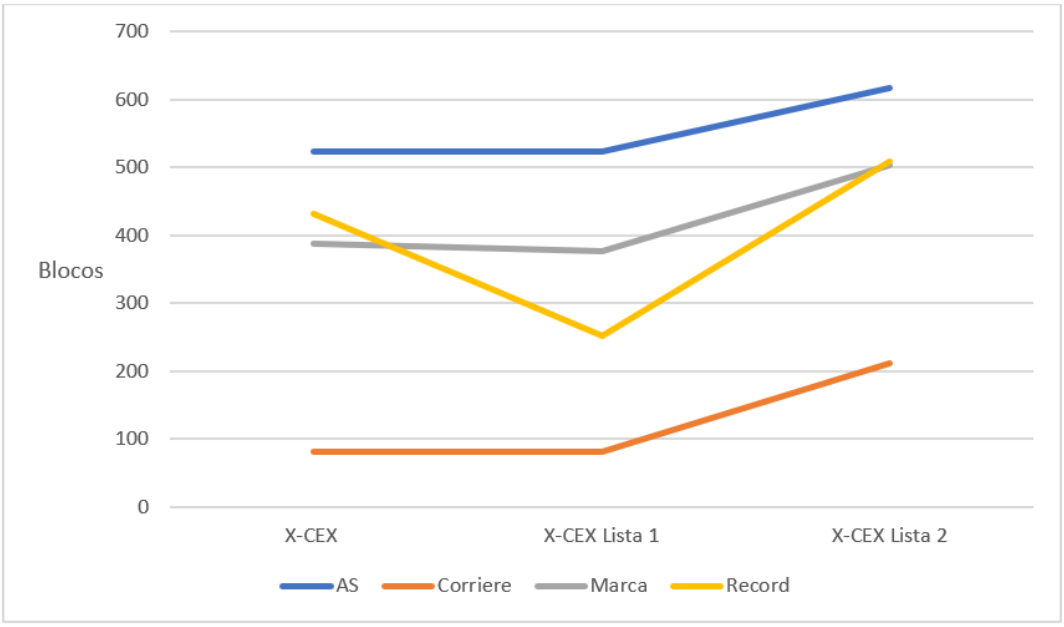

Figura 5.5: Blocos Extraídos para testes de Listas

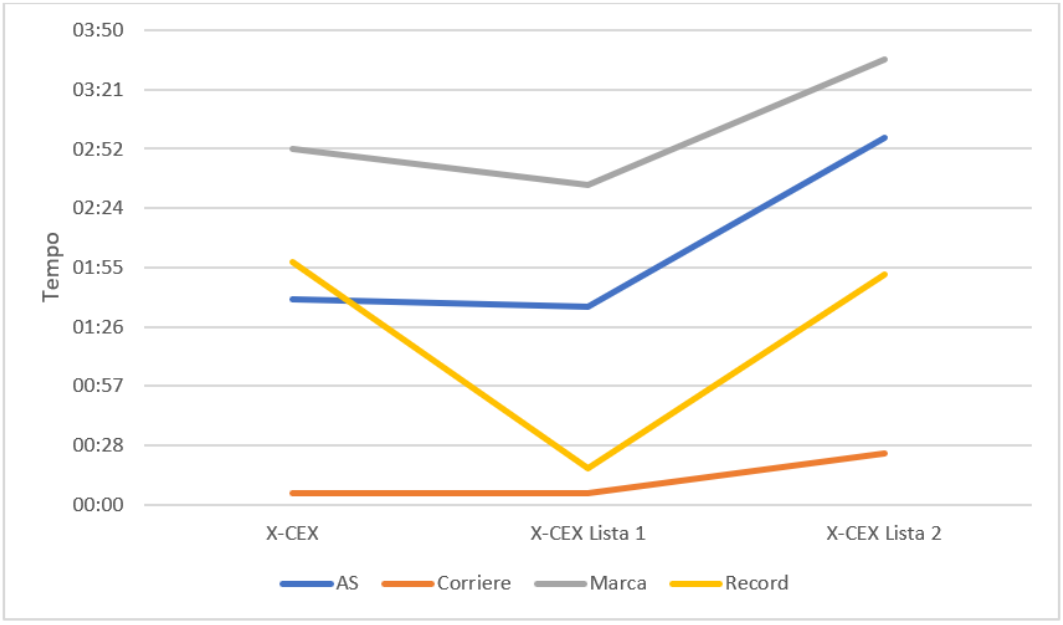

Figura 5.6: Tempo de Execução para testes de Listas

mostrarem nenhuma mudança relevante referente a tempo e blocos. Quanto ao L'Équipe, quero realçar que, apesar de extrair um número interessante de blocos no X-CEX, a sua estrutura tem muitos blocos de texto fora de etiquetas `<div>`, originando assim, uma exclusão de muitos destes blocos logo na fase de extração. Apesar disso, tal como no Record, ocorre ainda uma diminuição drástica de blocos extraídos do X-CEX para a versão com a Lista 1, esta mudança reflete apenas que a Lista 1 não funciona para estes templates, pois contém no artigo principal das notícias uma `<section>` que não tem nenhum texto, excluindo assim quase todos os blocos de texto relevante da página por estarem fora desta `<section>`.

A chave de uma boa lista de etiquetas HTML está em arranjar o equilíbrio certo para o número de blocos extraído não ser muito grande conseguindo, ainda assim, grandes resultados com as métricas de avaliação. Curiosamente podemos observar que da Lista 1 para a Lista 2, os jornais AS e Marca, acabam por aumentar o número de blocos perdendo em Precisão e Recall, isto significa que, quando se extrai demasiados blocos também será maior o "lixo"presente e, como tal, começa a perder-se alguma eficácia. Já para o Corriere dello Sport, podemos observar que, o aumento de blocos beneficia-o, pois tem um aumento enorme de Recall, apesar de perder ligeiramente Precisão, ou seja, com a Lista 2 aproximamo-nos do número de blocos ideal para este Site. Este facto mostra que a lista escolhida no algoritmo X-CEX não é a mais indicada para este template excluindo logo à partida alguns blocos importantes, mostra também que, os métodos usados no algoritmo para comparação de blocos são eficazes, pois apenas com a mudança da lista obtivemos excelentes resultados. Estes dados podem ser vistos nas Figuras 5.7 e 5.8.

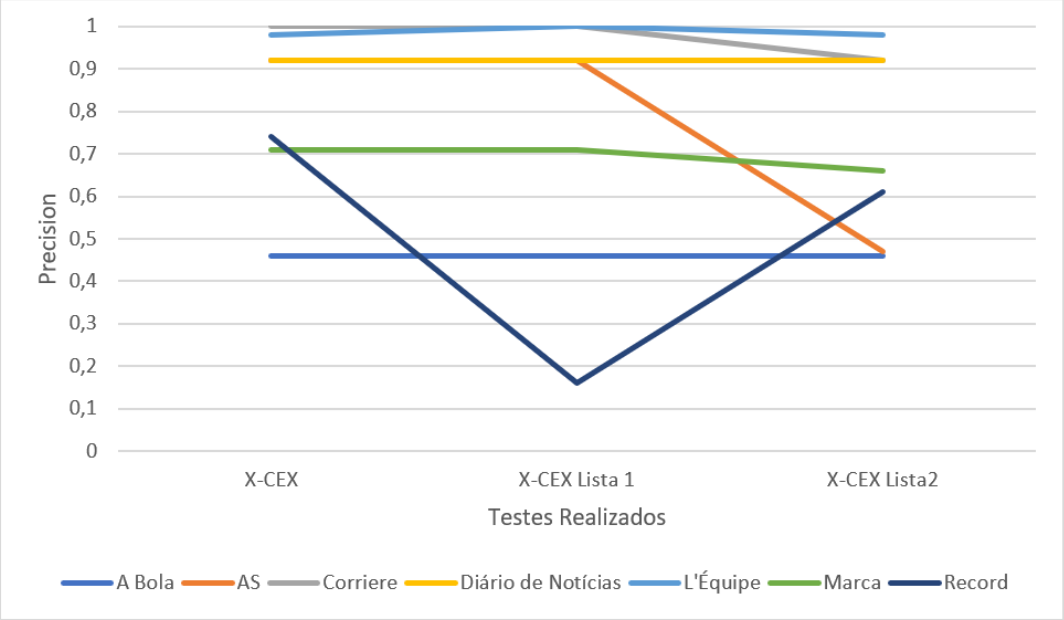

Figura 5.7: Precision para Testes de Listas

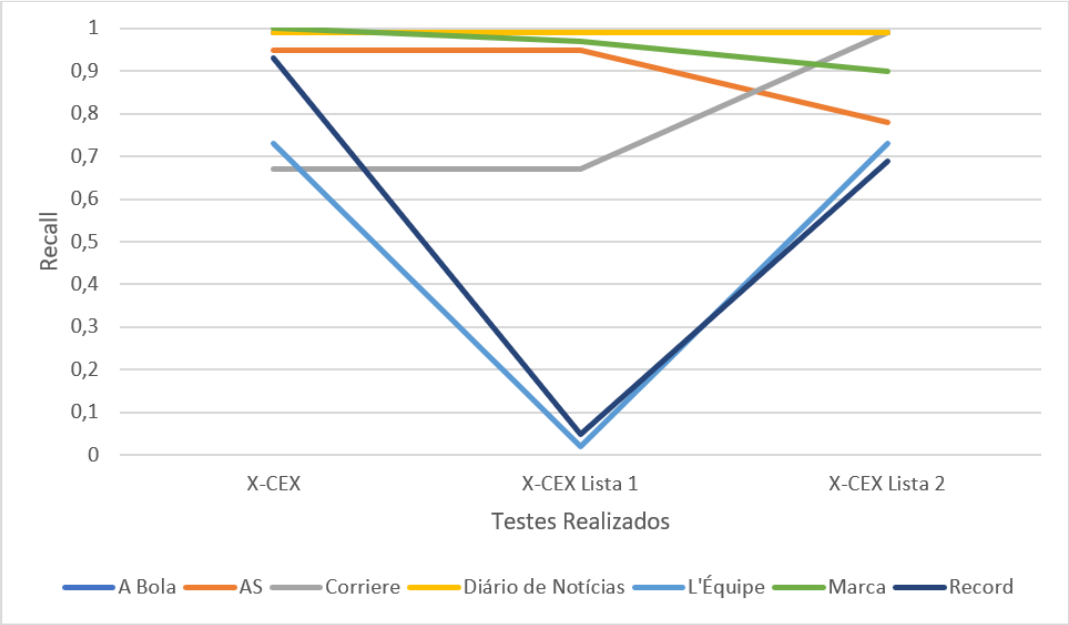

Figura 5.8: Recall para Testes de Listas

5.4.3 Testes para Comparações

O principal objetivo, com a introdução do valor de IBDF do bloco a -1 quando este fica excluído do grupo de blocos gerados pelo algoritmo, é diminuir o tempo de execução do algoritmo sem influenciar os resultados. Conseguimos observar com base nos resultados da Tabela 5.5 que o objetivo é cumprido, o tempo baixa para quase todos os templates testados, exceto para A Bola, Corriere dello Sport e Marca, em que as diferenças são mínimas tal como pode ser visto na Figura 5.9.

Com a introduçao do ibdf = -1, no caso do L'Équipe, os resultados pioram, pois, existem algumas notícias no dataset sobre exercício físico em que muitos dos blocos de conteúdo são exatamente iguais. Por exemplo, nestas páginas são sugeridos vários exercícios que devem ser feitos, e são sugeridos de forma igual em notícias diferentes, o que mostra que neste site, em notícias desta área são dadas as mesmas informações algumas vezes (talvez não tenham tantos leitores e acabam por não se dedicar tanto a esta categoria de notícias).

Nesta inovação, é importante realçar a excelente melhoria para os jornais AS e Record tal como podemos observar pela Figura 5.10, sendo que, com o X-CEX, existe um equilíbrio muito maior a nível das quatro métricas representadas. Este facto justifica-se por ambos os Sites terem blocos irrelevantes/exatamente iguais em poucas páginas. Sem esta inovação, estes blocos seriam considerados relevantes visto que se repetem pouco, no entanto, desta forma, basta repetirem-se uma vez e são automaticamente excluídos.

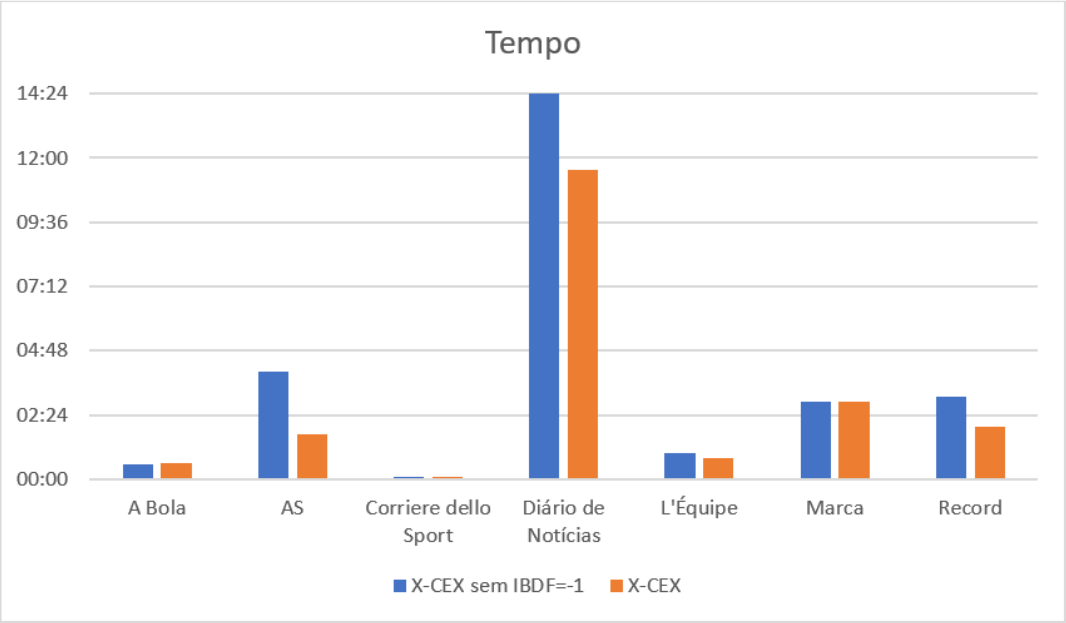

Figura 5.9: Tempo de Execução para Testes de IBDF=-1

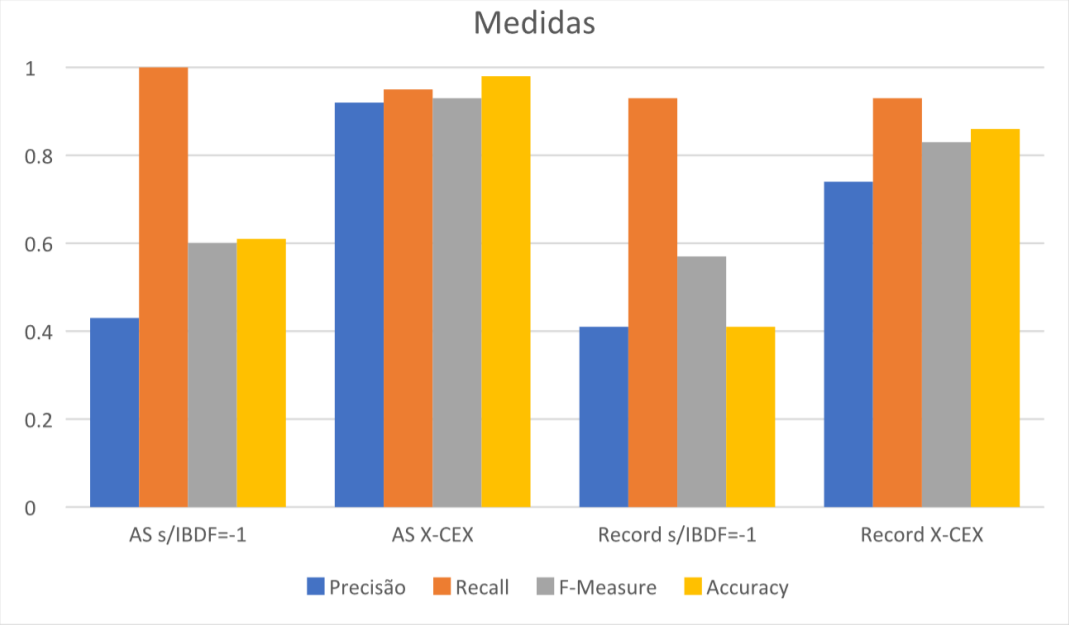

Figura 5.10: Métricas de avaliação para os jornais AS e Record

5.4.4 Testes para IBDF

Relativamente aos testes efetuados para o limite (treshold) do IBDF apresentados nas tabelas 5.6 e 5.7, é possível observar que com o aumento do mesmo, este tende a não produzir nenhum efeito a partir de uma certa percentagem. Como a extração de blocos é bastante restrita na primeira fase, a maior parte dos blocos sem conteúdo acabam por ser logo removidos nessa altura, os que acabam por passar dessa fase são quase todos, blocos repetidos ficando assim com o seu IBDF igual a -1, portanto são descartados numa segunda fase. Isto significa que vão existir poucos blocos afetados pelo limite estabelecido de IBDF, tal como pode ser provado pela Figura 5.11. Da mesma forma, os blocos com conteúdo e pouco similares com todos os outros acabam por ficar na sua maioria com um IBDF igual a 1, assim sendo, este limite apenas vai afetar os blocos ”mais duvidosos”daí a pouca influência que tem nos resultados obtidos.

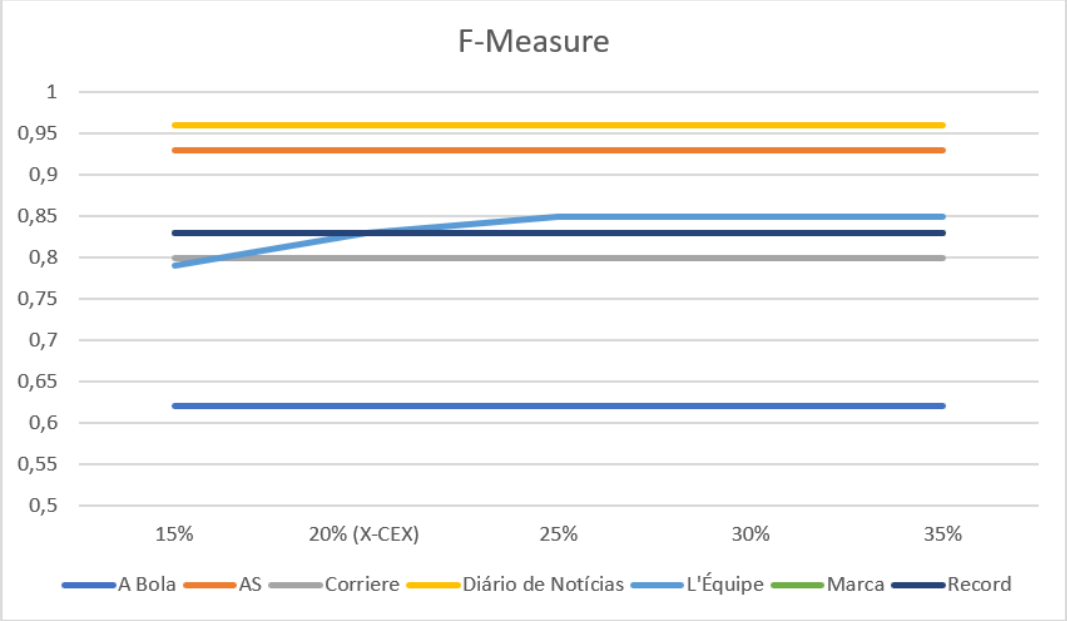

Figura 5.11: F-Measure para os testes do limite de IBDF

Tendo em conta esta observação, podemos então concluir que, existe um intervalo muito grande de valores de IBDF sem qualquer bloco, posicionando grande parte dos blocos presentes na página nos valores extremos de IBDF (-1 e 1).

Quero apenas realçar os resultados obtidos para o jornal L'Équipe, onde acaba por ser mais notório esta mudança de limite, como pode ser visto na Figura 5.12, para os 25% nota-se uma ligeira melhoria face aos 20% usados no X-CEX, isto pode-se justificar pelo facto de o L'Équipe ter mais notícias com temas mais diversificados, colocando uma parte dos blocos no grupo de ”blocos mais duvidosos”. Se só tivermos notícias sobre a mesma área fica mais fácil ao algoritmo perceber o que é ou não conteúdo.

Assim sendo, para este contexto, o limite que faz mais sentido é o de 25%, no entanto, fica a vontade futura de testar o algoritmo para temas diferentes no mesmo template.

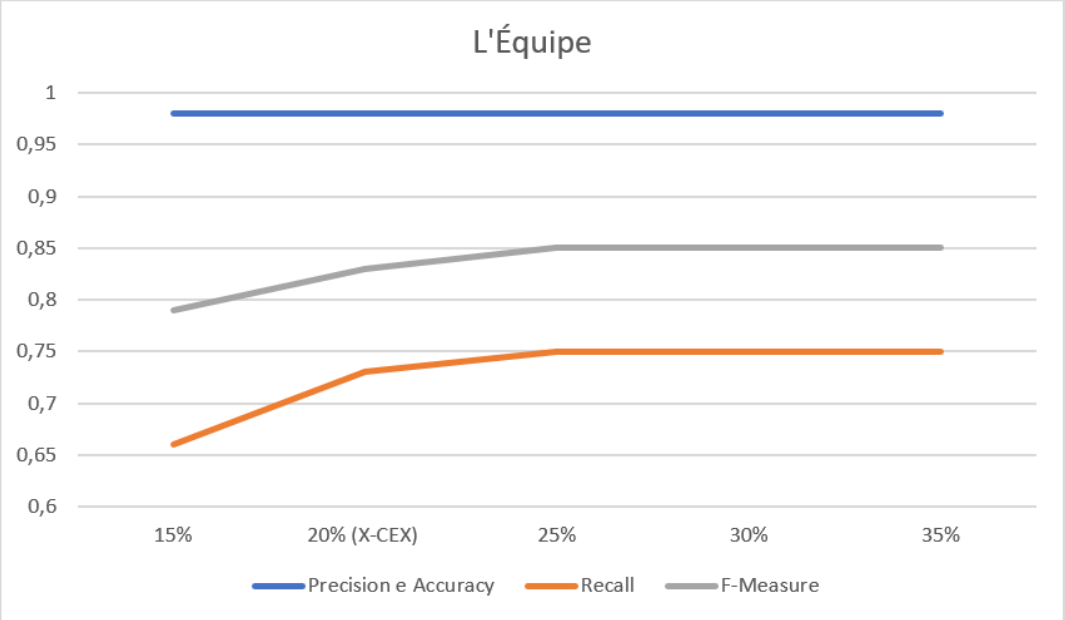

Figura 5.12: Testes de limite de IBDF para o L'Équipe

Num sistema que não tenha uma filtragem tão grande na fase de extração de blocos, creio que se vá notar mais diferenças de Recall com limites de IBDF menores, e mais diferenças de Precisão com limites de IBDF maiores, isto porque, com um limite muito pequeno vai haver muitos blocos que não vão entrar nos requisitos e, como tal, alguns blocos supostamente relevantes serão descartados, prejudicando o Recall. Por outro lado, com um limite de IBDF muito grande, vai haver muitos blocos a cumprir os requisitos e, como tal, existirão muitos blocos irrelevantes a ser considerados como blocos com conteúdo provocando assim uma descida na Precisão.

5.4.5 Testes para Importância de Similaridade

A maior inovação implementada foi a introdução de um sistema de importâncias no algoritmo, este sistema foi testado e os resultados foram apresentados nas tabelas 5.8, 5.9, 5.10, 5.11 e 5.12 em comparação com os resultados obtidos pelo algoritmo X-CEX na tabela 5.1.

Em primeiro lugar, é possível observar que, quando é usado uma importância de 50% ao conteúdo, o algoritmo obtém um excelente tempo de execução tal como pode ser visto na Figura 5.13. Este tempo é justificável pelo facto de encontrar muito mais comparações entre blocos, cuja similaridade é superior ao limite (treshold) definido, assim, sempre que é encontrada uma similaridade superior ao limite, a página do bloco similar deixa de ser analisada e passa à seguinte. Neste caso, com 50%

de importância ao conteúdo, os blocos comparados acabam por encontrar muito mais rapidamente outros blocos similares por todos terem estruturas e estilos idênticos, desta forma, evita-se muitas comparações baixando o tempo drasticamente.

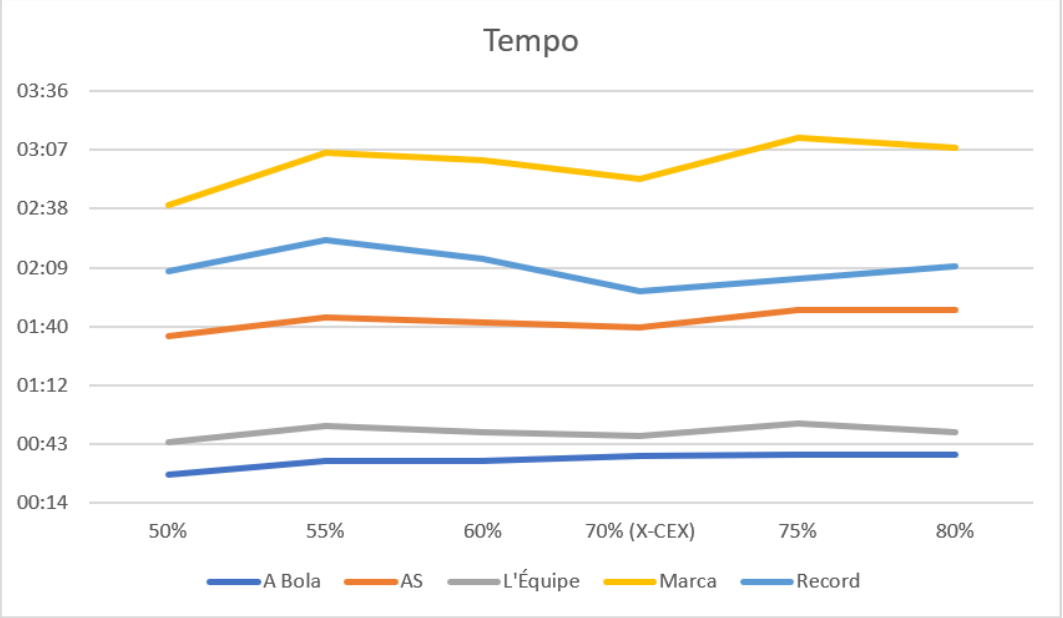

Figura 5.13: Tempo de Execução para diferentes Importâncias

Nesta imagem, não foi inserido o tempo para o Diário de Notícias nem para o Corriere dello Sport por se distanciarem demasiado das outras e dificultarem a visualização do gráfico.

Através dos valores de Recall representados na Figura 5.14, consegue-se perceber perfeitamente o comportamento do algoritmo no que toca ao tempo. Isto porque, como é dada uma importância de 50% ao Estilo e Estrutura, colocando-a no mesmo patamar que o Conteúdo, então acaba por prejudicar imenso os blocos com conteúdo importante e relevante para a notícia só porque têm uma aparência igual à de outros blocos. Desta forma, existem imensos blocos com conteúdo que não considerados como tal pelo algoritmo prejudicando assim os valores de Recall, ao contrário do que acontece no X-CEX.

Assim que o equilíbrio de importância entre a Estrutura/Estilo e o Conteúdo é encontrada, o Recall tende a estabilizar, visto que, os blocos com conteúdo têm, na sua maioria, diferenças no texto apresentado, notando-se logo uma grande diferença dos 50% para os 55%. A partir dos 60% de importância, o valor já não mexe muito, o que nos leva a crer que, uma importância de 60% já seria suficiente para obter bons resultados. É importante descrever separadamente o caso do L'Équipe, que, a partir dos 60% de importância para o conteúdo, começa a descer os valores de Recall, isto justifica-se, pelos blocos importantes existentes com conteúdo igual (como, por exemplo, a recomendação de exercícios físicos nas notícias dessa categoria), tal como já foi mencionado num dos testes anteriores. Mais uma vez,

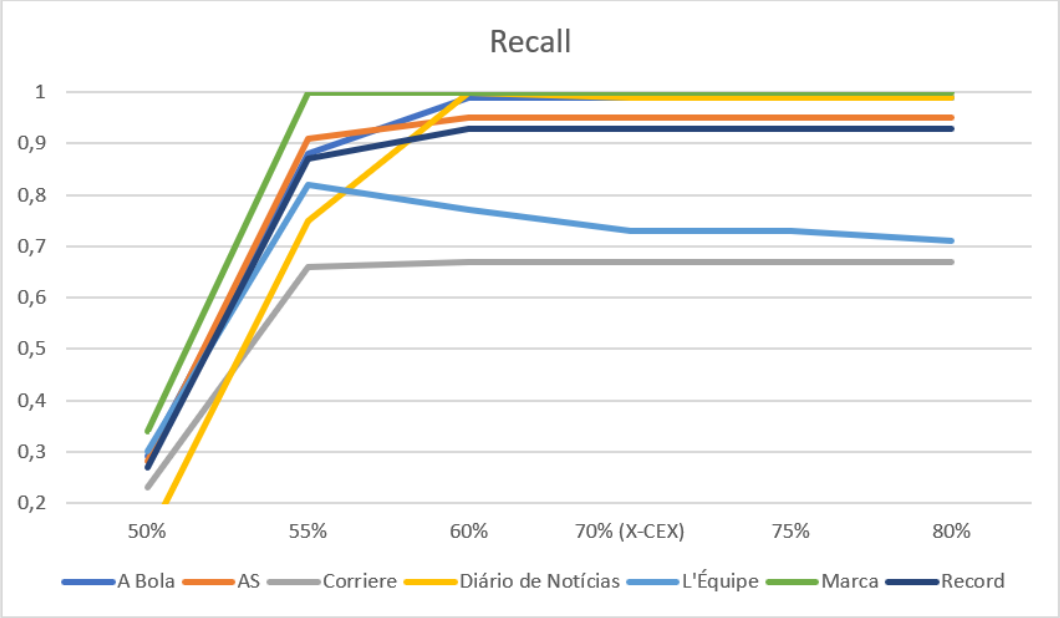

Figura 5.14: Recall para diferentes Importâncias

fica a nota, que se houvesse notícias do mesmo Site, mas sobre mais áreas diversificadas poderíamos assistir aqui a alguns comportamentos como o do L'Équipe.

Relativamente à Precisão, através da Figura 5.15 podemos observar que se nota um ligeiro decréscimo neste campo, no teste dos 50%, isto porque, como se dá igual importância às duas componentes, então, blocos que tenham aparências muito diferentes acabam por ser considerados como blocos com conteúdo independentemente do seu texto, ou seja, acaba por criar mais ”lixo”nos blocos finais obtidos baixando assim a Precisão. Não se nota tanta discrepância na Precisão como no Recall, porque não existem assim tantos blocos com aparências muito diferentes e com conteúdo igual, por isso, os blocos considerados como relevantes no X-CEX também o são no teste com 50%. A Precisão acaba por estabilizar pelas mesmas razões que o Recall, sendo que, os blocos sem conteúdo dificilmente apresentam textos diferentes não passando por isso numa alta importância para o conteúdo.

O caso da A Bola, é um caso específico, tal como já referido, o X-CEX apanha muitos comentários dos utilizadores como blocos relevantes neste template, estes têm a mesma estrutura, mas conteúdos muito diferentes, como tal, estes blocos de comentários acabam por ser prejudicados com métricas que dão mais importância ao estilo e estrutura do que ao conteúdo aumentando assim a precisão e a accuracy para o teste dos 50%. Alguns destes blocos de comentários, deixam assim de ser considerados como relevantes, não afetando o recall, pois este baseia-se apenas nos blocos relevantes deixados de fora.

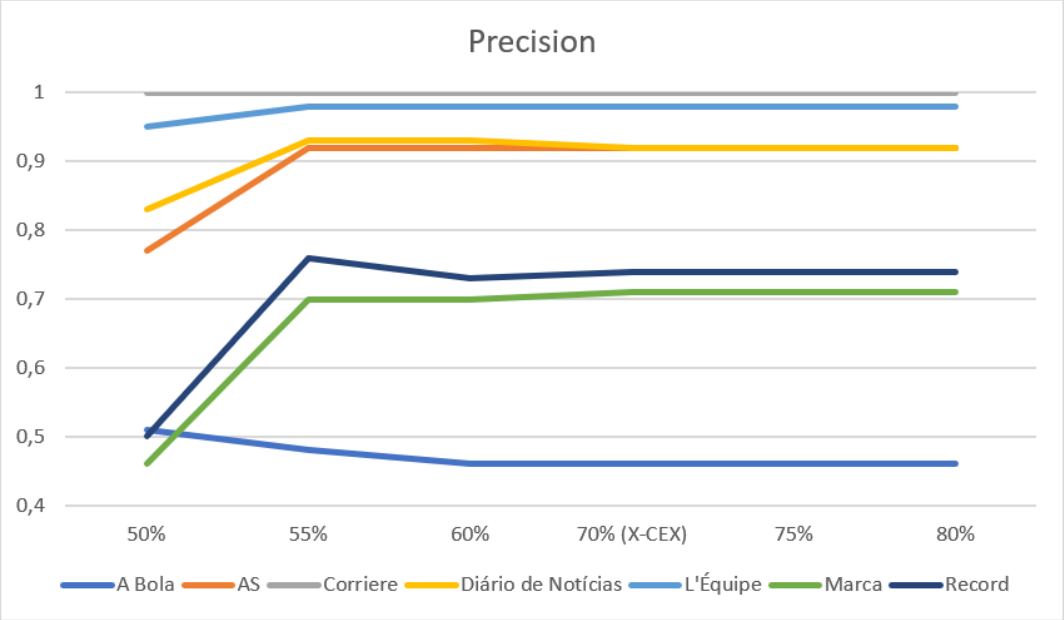

Figura 5.15: Precisão para diferentes Importâncias

5.4.6 Testes para métricas

Para finalizar as análises aos testes, foram ainda testadas as métricas escolhidas para calcular as similaridades de Estrutura/Estilo e as similaridades de Conteúdo. Os resultados para a Estrutra e Estilo estão apresentados nas tabelas 5.13 e 5.14 usando o Cosseno e o Método de Griazev respetivamente.

Tanto o método com o Cosseno dos vetores de características para a Estrutura e Estilo, como o método de Griazev, obtêm os mesmos resultados em Precisão, Recall, F-Measure e Accuracy comparando com o X-CEX, no entanto, ambos os métodos acabam por melhorar em tempo de execução para todos os Web Sites testados, como pode ser visto na Figura 5.16, ao contrário do que seria expectável.

Ainda assim, podemos observar que o método de Griazev consegue alcançar tempos de execução menores face aos tempos com o Cosseno. Este facto explica-se, pela simplicidade do método de Griazev e por ser eliminado o fator do nível do bloco do mesmo. Ainda assim, mesmo melhorando o tempo de execução, este acaba por não fazer uma grande diferença face aos tempos de execução do X-CEX.

Os resultados para os testes de Similaridade de Conteúdo são apresentados nas tabelas 5.15, 5.16 e 5.17, sendo a primeira para a distância de Levenshtein, a segunda para a distância de Damerau-Levenshtein e a terceira para a distância de Hamming. É importante explicar que a distância de Jaro-Winkler também foi implementada para este teste, no entanto, não apresentou nenhum bloco com conteúdo mostrando assim que esta distância não funciona para esta categoria de análise textual.

Os tempos de execução com estas métricas para a similaridade de Conteúdo foram uma grande

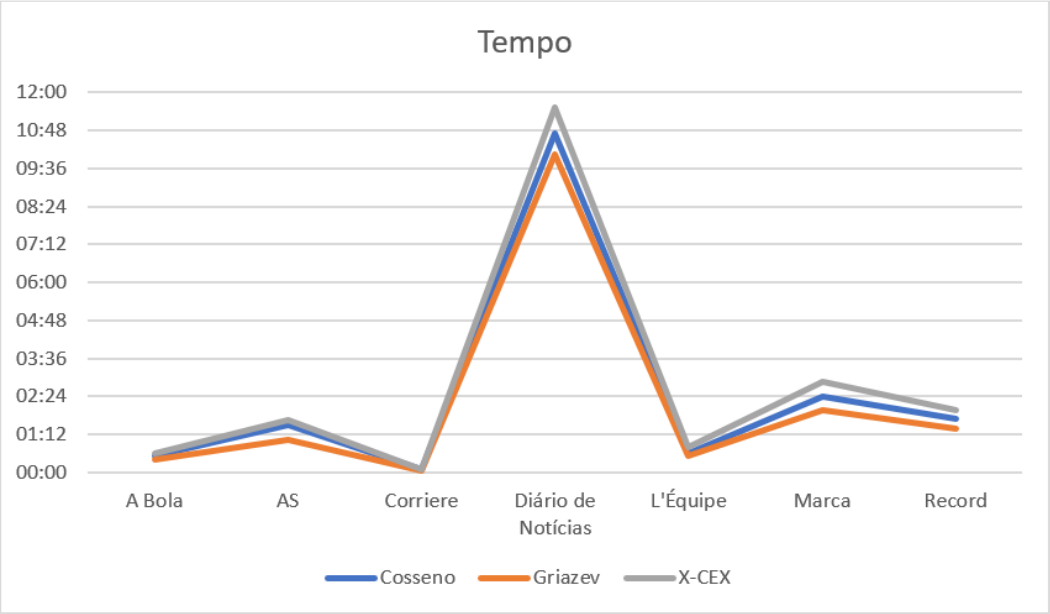

Figura 5.16: Tempo de execução para método do Cosseno, método de Griazev e X-CEX

surpresa, primeiro percebe-se logo pelas tabelas que a distância de Levenshtein e a distância de Damerau-Levenshtein tiveram um crescimento exponencial no tempo de execução, tornando o algoritmo praticamente não executável, muitos dos sites acabam mesmo por serem executados durante muitas horas, retirando toda a eficácia do algoritmo. Ainda assim, apesar do tempo de execução estes dois métodos alcançam melhores resultados em Recall para o template do L'Équipe, piorando por outro lado, para o jornal Marca e Record.

Na Figura 5.17, onde não incluí os tempos das distâncias de Levenshtein e de Damerau-Levenshtein pelos motivos acima mencionados, podemos observar que a distância de Hamming consegue obter uma redução enorme no tempo de execução face ao algoritmo X-CEX, mantendo os resultados para a Precisão, Recall, F-Measure e Accuracy. Achei surpreendente os resultados obtidos, só conseguindo explicá-los talvez pelo fator da distância de Hamming ter em conta apenas o número de posições que não correspondem nas duas 'strings' fornecidas. Desta forma, creio que, calcular uma similaridade de conteúdo através do Cosseno de vetores binários para os termos usados nos blocos, acaba por ser mais exaustivo por ter de percorrer antecipadamente todas as palavras, ao contrário da distância de Hamming.

Para concluir, podemos afirmar que a mudança de métodos para o cálculo das similaridades, não é crucial para a obtenção de bons resultados de Precisão, Recall, F-Measure e Accuracy. O fator que mais influência tem nestes resultados, é a incorporação da importância das componentes Estrutura/Estilo e Conteúdo. Assim, a estrutura dos blocos passou a ser mais relativizada e foi dada uma maior relevância ao conteúdo de cada bloco, graças a esta melhora conseguimos ver resultados bem mais fiáveis.

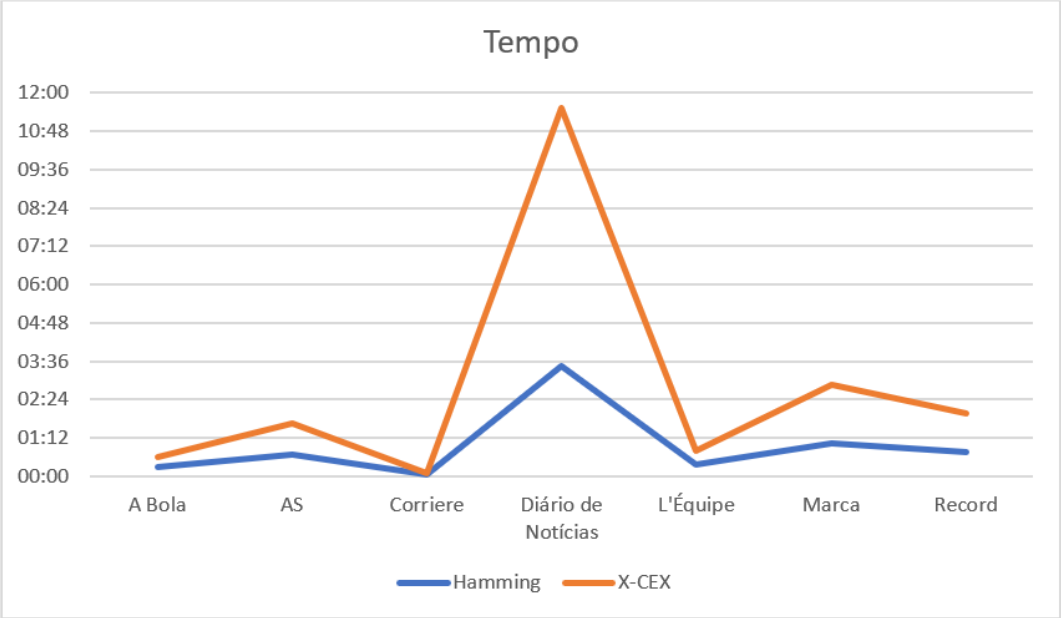

Figura 5.17: Tempo de execução para X-CEX e distância de Hamming

5.5 Sumário

Neste capítulo, foram descritos todos os datasets usados e como foram extraídos, foram ainda apresentados todos os testes realizados e os seus respetivos resultados. Por fim, foram analisados todos os resultados e tiradas certas conclusões sobre todas as inovações propostas.

Podemos concluir que, o algoritmo X-CEX apresenta grandes resultados e uma excelente fiabilidade, sendo que, as melhorias que mais efeito tiveram para estes resultados, foram, a inclusão de um sistema de importâncias e a mudança da lista de etiquetas para a extração dos blocos. Para finalizar, foi possível ainda concluir que, usando a distância de Hamming para medir a similaridade dos blocos quanto ao seu conteúdo, se obtém melhores resultados ao nível do tempo de execução não comprometendo de todo com a eficácia do mesmo.

6

Conclusão

Conteúdo

Neste capítulo, apresento as principais conclusões deste trabalho, tanto a nível do que foi alcançado, como também a nível das limitações. Por fim, aponto ainda possíveis caminhos futuros para quem quiser desenvolver ou aprofundar mais este trabalho.

6.1 Conclusões

Para concluir este trabalho tenho de dividi-lo em duas partes, primeiro em termos do que foi proposto, implementado e testado, e explicar o que alcançamos com esta solução, e em segundo lugar, falar das limitações desta solução, há sempre espaço para algumas lacunas e essas têm de ser descritas e anunciadas para se poderem resolver no futuro.

6.1.1 Discussão

Durante este último ano, desenvolvi a solução para o problema descrito no capítulo 1, neste processo, foram ultrapassadas imensas barreiras, tendo apanhado pelo meio a pandemia da Covid-19, o que dificultou em parte todo o processo de pesquisa, implementação e de testes. Apesar de tudo, o trabalho foi desenvolvido com todo o gosto e toda a vontade de conseguir criar algo melhor e importante para o desenvolvimento desta àrea.

Como tal, creio chegar ao fim deste ano bastante contente com a solução proposta. O algoritmo implementado e proposto por mim para extrair conteúdo informativo de páginas Web de notícias, foi o algoritmo X-CEX, sendo este, uma versão melhorada e adaptada do algoritmo Content Extractor. Conseguimos concluir que este algoritmo não é afetado pelas línguas das notícias, e consegue apresentar bons resultados para vários linguagens, esta que era, uma componente importante para ser avaliada.

Agora no fim, consigo responder às perguntas que se impõem, as componentes que mais influenciam esta extração é, por todo o trabalho desenvolvido e analisado, o filtro/extração inicial dos blocos, ou seja, a fase do início em que, uma página é dividida em várias secções, havendo desde logo uma filtragem quanto aos blocos que iam para a fase de comparação. E depois, e não menos importante, o sistema usado para o cálculo da similaridade entre dois blocos, esta é das fases mais preponderantes do processo em que se tem de otimizar ao máximo de forma a obter os melhores resultados possíveis, o sistema que foi usado neste trabalho foi um sistema de importâncias para cada uma das similaridades (Estrutura/Estilo e Conteúdo) ao contrário do Content Extractor que usava um sistema de junção de ambas as similaridades juntas.

Relativamente à escolha de aprimorar o algoritmo Content Extractor, chego à conclusão que provavelmente teria sido melhor aprofundar os algoritmos Feature Extractor e o K-Feature Extractor que também têm algumas bases deste Content Extractor, no entanto, a complexidade computacional desses fez com que me afastasse dos mesmos. Com mais tempo disponível, certamente seria um caminho

a fazer pelas potencialidades que teria. Ainda assim, esta escolha revelou-se boa pelos resultados que conseguimos obter e por todas as investigações que foram feitas no mesmo, foi feita muita pesquisa e muita análise ficando-se agora a perceber bem o caminho a seguir para a evolução do algoritmo, resta alguém dar seguimento.

Por fim referir que, após todos os testes realizados percebi que a melhor solução seria usar o sistema de importâncias por mim sugerido, mas mudar a métrica de similaridade de conteúdo para a distância de hamming, que se revelou uma excelente métrica em tempo de execução e resultados obtidos. Os resultados obtidos para todos os testes realizados permitem-nos obter mais informação sobre possíveis alternativas e desta forma perceber quais delas podem ou não ser exploradas no futuro.

Em jeito de conclusão, consegui com isto perceber, que esta área está muito analisada e já existem muitas alternativas à solução deste problema, no entanto, também tenho de referir que me parece haver demasiadas opções pouco exploradas, ou seja, na minha opinião, presumo que as soluções existentes deviam sim, ser mais aprofundadas e analisadas, devia haver um maior foco nos métodos existentes e não haver tantas opções diferentes.

6.1.2 Limitações

O algoritmo X-CEX, tem obviamente as suas limitações, esta minha proposta acaba por ficar bastante limitado, por exemplo, quanto ao tipo de notícias que pode analisar, este algoritmo acaba por não estar adaptado a notícias de acompanhamentos de eventos em direto, também não está adaptado a notícias que acabam por ser repetitivas, porque quando assim é, acaba por desvalorizar o conteúdo das mesmas.

A maior limitação do meu algoritmo é, o ficar limitado a analisar templates de páginas, ou seja, só funciona para um conjunto de páginas da mesma classe e, como tal, deixa de funcionar corretamente quando o número de páginas fornecido é baixo, perde a eficiência por não conseguir identificar o template do Web Site.

Por último, outra das grandes limitações deste algoritmo é não ser capaz de extrair os blocos com conteúdo que estejam fora das etiquetas usadas para a fase de extração de blocos. Esta é uma grande limitação, pois quando os Web Sites mudam radicalmente de estrutura, este algoritmo deixa de obter bons resultados. Por esse motivo, em alguns Web Sites, os blocos dos títulos e sub-títulos, por exemplo, podem ficar de fora erradamente, a mesma coisa acontece para imagens ou publicações externas ao site, relativas à notícia.

6.2 Trabalho Futuro

O trabalho proposto respondeu à questão central levantada e contribuiu para a extração dos blocos principais com conteúdo, contudo, existem alguns pontos que podem ser melhorados e podem ser explorados daqui para a frente.

Como trabalho futuro para este algoritmo, recomendo explorar bem a fase de extração de blocos, esta é a maior lacuna do algoritmo por estar demasiado restrita a certos templates. Agora, esta fase de extração usa um sistema de lista de etiquetas ordenadas, no entanto, creio que este não é o melhor método acabando por retirar muitos blocos relevantes da fase de comparação, deve sim, ser usado mais algum critério em conjunto com este. É importante remover automaticamente certas zonas das páginas, no entanto, nas secções principais devem ser usados outros métodos que, por exemplo, peguem no texto dessas mesmas secções ou noutras componentes da secção.

Outro dos pontos que pode e deve ser explorado para trabalho futuro é a extração final do conteúdo, não só de blocos com conteúdo, mas sim do conteúdo efetivo desses mesmos blocos, e assim, tentar apresentá-lo num ambiente gráfico ou numa 'interface'. Esta seria uma excelente inovação futura, pois significaria, a continuação deste projeto para algo possível de ser utilizado para os utilizadores comuns.

Para finalizar, penso que, no futuro, deve ser melhorado o processo de extração dos datasets e da verdade absoluta dos mesmos (conteúdo efetivo das páginas), para desta forma obter os resultados mais fiáveis possível, juntando a esse facto é importante testar o algoritmo para datasets mais diversificados, ou seja, tentar tirar vários tipos de notícias do mesmo template, mas dentro de muitos tópicos diferentes para perceber se essa pode ser uma componente relevante para o algoritmo ou não.

Bibliografia

[1] L. Yi, B. Liu, and X. Li, “Eliminating noisy information in Web pages for data mining,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[2] S. Debnath, P. Mitra, N. Pal, and C. L. Giles, “Automatic identification of informative sections of Web pages,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 9, pp. 1233–1246, 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1490530/

[3] S. Goyvaerts, J., Levithan, “Regular Expressions Cookbook,” *Network Security*, 2012.

[4] J. Jiang, “Information extraction from text,” in *Mining Text Data*, 2013.

[5] V. K. Govindan and A. P. Shivaprasad, “Character recognition - A review,” *Pattern Recognition*, 1990.

[6] J. Piskorski and R. Yangarber, “Information Extraction: Past, Present and Future,” 2013.

[7] L. Epstein and G. King, “The rules of inference,” *University of Chicago Law Review*, 2002.

[8] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, 2013.

[9] L. Yu and H. Liu, “Efficient feature selection via analysis of relevance and redundancy,” *Journal of Machine Learning Research*, 2004.

[10] D. Greene, P. Cunningham, and R. Mayer, “Unsupervised learning and clustering,” in *Lecture Notes in Applied and Computational Mechanics*, 2008.

[11] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning. Adaptive Computation and Machine Learning*, 2010.

[12] D. Freitag, “Machine learning for information extraction in informal domains,” *Machine Learning*, vol. 39, no. 2, pp. 169–202, 2000.

[13] S. Vijayarani and K. Geethanjali, “Web Page Noise Removal - A Survey,” *International Journal of Scientific Research in Scienceand Technology*, vol. 3, no. 7, pp. 172–181, 2017.

[14] S. S. Bhamare and B. V. Pawar, “Survey on Web Page Noise Cleaning for Web Mining,” *Imternational Journal of Computer Science and Information Technology*, vol. 4, no. 6, pp. 766–770, 2013.

[15] A. K. Rajashri Shinde, Ashwini Bolli, “Methods For Extracting Content Blocks From Web Pages,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 4, pp. 1–10, 2013.

[16] A. Finn, N. Kushmerick, and B. Smyth, “Fact or Fiction: Content Classification for Digital Libraries,” in *Proceedings of the Second {DELOS} Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries, {DELOS} 2001, Dublin, Ireland, June 18-20, 2001*, ser. {ERCIM} Workshop Proceedings, A. F. Smeaton and J. Callan, Eds., vol. 01/W03. ERCIM, 2001. [Online]. Available: http://www.ercim.org/publication/ws-proceedings/DelNoe02/AidanFinn.pdf

[17] D. Pinto, M. Branstein, R. Coleman, W. B. Croft, M. King, W. Li, and X. Wei, “QuASM: A system for question answering using semi-structured data,” in *Proceedings of the ACM International Conference on Digital Libraries*, 2002.

[18] J. Robie and S. Ag, “Document Object Model (DOM) Level 2 Core Specification,” *W3C Recommendation*, 2000.

[19] P. U. Jadhav, “Extracting Information from website using Site Style Tree,” *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 6, pp. 1654–1656, 2014.

[20] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm, “DOM-based content extraction of HTML documents,” in *Proceedings of the 12th International Conference on World Wide Web, WWW 2003*, 2003.

[21] D. Cai, S. Yu, J. R. Wen, and W. Y. Ma, “VIPS: a visionbased page segmentation algorithm,” *Beijing Miciosoft Research Asia*, 2003.

[22] W. Liu and X. Meng, “Vision-based Web Data Records Extraction,” *In Proceedings of the 9th SIGMOD International Workshop on Web and Databases SIGMODWebDB2006*, 2006.

[23] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data Second Edition*. Springer, 2011.

[24] N. Kushmerick, “Learning to remove Internet advertisements,” *Proceedings of the International Conference on Autonomous Agents*, pp. 175–181, 1999.

[25] S. H. Lin and J. M. Ho, “Discovering informative content blocks from Web documents,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 588–593, 2002.

[26] Z. Bar-Yossef and S. Rajagopalan, “Template detection via data mining and its applications,” in *Proceedings of the eleventh international conference on World Wide Web - WWW ’02*. New York, New York, USA: ACM Press, 2002, p. 580. [Online]. Available: http://portal.acm.org/citation.cfm?doid=511446.511522

[27] M. F. Porter, “An algorithm for suffix stripping,” 1980.

[28] M. Marathe, S. H. Patil, G. V. Garje, and M. S. Bewoor, “Extracting Content Blocks from Web Pages,” *International Journal*, vol. 2, no. 4, pp. 62–64, 2009.

[29] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley; 1st edition, 1999.

[30] J. Gibson, B. Wellner, and S. Lubar, “Adaptive web-page content identification,” in *International Conference on Information and Knowledge Management, Proceedings*, 2007.

[31] T. Gowda and C. Mattmann, “Clusteringweb pages based on structure and style similarity,” in *Proceedings - 2016 IEEE 17th International Conference on Information Reuse and Integration, IRI 2016*, 2016.

[32] K. Griazev and S. Ramanauskaitė, “HTML block similarity estimation,” in *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering, AIEEE 2018 - Proceedings*, 2018.

[33] C.J. Van Rijsbergen, “Information Retrieval,” *Butterworth- Heinemann*, vol. 30, no. 3, 1979.

[34] D. L. Olson and D. Delen, *Advanced data mining techniques*. Springer, 2008.

A

Detalhes da Implementação

Para a implementação da solução proposta foi usada a linguagem de programação Python por ser não só uma linguagem de programação orientada a objetos como também por ser a linguagem que mais usei no curso, tendo por isso uma adaptação mais facilitada.

Para analisar as páginas Web fornecidas usei a biblioteca do Python Beautiful Soup[1], esta biblioteca é usada com o objetivo de navegar, procurar e modificar mais intuitivamente uma página Web pois esta cria uma árvore de análise da página podendo assim percorrê-la. A versão utilizada foi a versão 4.

A.1 Extração de Datasets

Ambos os scritps têm de usar a codificação ”UTF-8”para poder extrair todas as informações dos web sites visto que caso contrário eram extraídos caracteres impossíveis de identificar.

No primeiro script foram dados por mim vários links da mesma fonte, ou seja, da mesma classe. Este script foi usado para a leitura dos links, sendo que cada um deles era uma página que continha várias notícias sendo por isso uma coletânea de notícias, como tal, este primeiro script tinha como objetivo

[1] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

extrair o ficheiro HTML de cada uma dessas notícias e guardá-lo, resultando assim numa coleção de páginas Web da mesma classe. Este script navega pela árvore até encontrar a secção exata das notícias listadas dada por um certo identificador e por uma etiqueta, encontrando essa secção acaba por ler as várias etiquetas <a> estando cada uma delas associada diretamente a um url de uma notícia. Esse url é extraído através do campo ”href”e o mesmo é lido, criando-se assim um ficheiro HTML para cada uma das notícias encontradas.

Este script nem sempre foi fácil de implementar visto que havia várias fontes (web sites) que dividiam as notícias em várias secções o que dificultava imenso a extração das mesmas, cada fonte teve de ser analisada uma a uma para poder extrair o máximo de notícias possíveis. Outra das grandes dificuldades na realização deste script era a remoção ou mais corretamente o filtrar de algumas notícias, muitas delas nestes web sites são pagas, são vídeos ou são mesmo só galeria de fotos posicionando-se nos mesmos locais das outras notícias ”normais”e como tal, esta foi uma dificuldade acrescida. Na altura da implementação do script tive de analisar cada web site pormenorizadamente para verificar se tinha algum deste tipo de notícias, caso tivesse, tive de procurar formas de as evitar como por exemplo, detetar a palavra *premium* dentro da informação da notícia, ver se tinha alguma referência a uma galeria de fotos/vídeos para poder generalizar a extração a todo o web site. Tendo em conta que este trabalho foi testado em mais do que um template, ou seja, para mais do que um web site então este processo de análise teve de ser feito para todas as classes escolhidas para os Datasets.

Depois desta extração foi implementado o segundo script, que basicamente, servia para extrair os conteúdos relevantes de cada uma destas notícias, ou seja, este script lê todos os ficheiros HTML guardados do primeiro script e retira todos os blocos com conteúdo dos mesmos. Este script é executado logo automaticamente ainda antes do ínicio do processo do algoritmo X-CEX ficando assim com a lista de blocos com conteúdo da coleção de documentos fornecida ao algoritmo proposto. Esta lista não é mais do que os resultados que o algoritmo proposto deveria extrair, portanto esta serve no fim para poder comparar os blocos gerados pelo algoritmo calculando nessa altura os resultados para as métricas de avaliação descritas na secção 5.2. Da mesma forma que o primeiro script, para este também tive de analisar a estrutura das páginas individualmente para perceber qual era o identificador dos blocos com o conteúdo relevante.

Apesar de resultar perfeitamente, este script também teve os seus problemas, principalmente na deteção de alguns ficheiros que não exatamente notícias como acompanhamentos em direto das mais variadas situações e, como tal, esses ficheiros teriam de ser descartados pois não eram compostos por uma estrutura igual aos ficheiros de notícias do site impossibilitando assim a extração dos blocos com conteúdo dessas notícias. Outra dificuldade, mais facilmente ultrapassável, foi o facto de certos web sites não manterem a mesma estrutura em todas as notícias ou seja, têm sempre 3 ou 4 identificadores diferentes para os blocos com conteúdo dependente da notícia que era, como tal, tive de criar 3 ou 4

panoramas diferentes para estes web sites e assim poder extrair o conteúdo adequadamente.

A.2 Implementação do Algoritmo

Esta implementação foi baseada em classes e objetos criados no Python, foram criadas 4 classes. A primeira é a classe Site que correspondia diretamente às informações do web site usado para o fornecimento das notícias, esta tinha como atributos o nome, uma lista de Páginas e o corpus da coleção. Foi também usada a classe Página que correspondia a todas as informações que cada notícia tinha, ou seja, era composta por um identificador, uma lista de blocos, o número de blocos que essa página tinha, a linguagem da página e ainda qual o Site da mesma. A terceira classe é a classe Bloco, a mais importante e a que contém mais informações, esta tem atributos como o identificador, o nível do bloco na árvore da página, o conteúdo do bloco, o seu valor de IBDF, a página a que pertence, o vetor de características do bloco, um vetor binário para as palavras, o texto do bloco e ainda o índice do bloco na página em questão. Por fim, foi criada ainda a classe Parâmetros que basicamente serve para aglomerar todos os parâmetros usados na execução deste algoritmo, parâmetros como a lista de etiquetas, o tipo do Dataset (se treino ou teste), a medida de Similaridade usada para o Estilo/Estrutura e para Conteúdo, a importância da Similaridade do Estilo/Estrutura e do Conteúdo e por fim o limite (treshold) para a Similaridade.

A.2.1 Extração de Blocos

O processo da Extração de Blocos foi implementado de forma muito parecida ao do algoritmo Content Extractor, sendo que foi seguido com bastante rigor o pseudo-código do mesmo para esta fase do algoritmo.

Este procedimento começa por retirar a primeira etiqueta da lista e gerar todos os blocos com essa mesma etiqueta, estes são adicionados à lista de blocos final e remove-se assim o bloco pai, ou seja, o bloco que originou estes novos blocos. Depois, prossegue-se para a segunda etiqueta e gera-se os blocos dessa mesma etiqueta a partir dos blocos gerados anteriormente, ou seja, o bloco pai nesta segunda iteração será um dos blocos gerados na primeira iteração. Os novos blocos são também inseridos na lista e volta-se a remover o bloco pai que os gerou, este procedimento ocorre consecutivamente até terminar toda a lista de etiquetas e assim ficam gerados os blocos de uma página. Repete-se este processo para a segunda página e assim consecutivamente chegando ao final desta fase com todos os blocos finais desta coleção de páginas. É importante realçar que caso o bloco pai não tenha qualquer filho gerado com a etiqueta do momento então esse bloco não será removido da lista pois ainda poderá ter outras da lista dentro do seu conteúdo.

Pelo meio deste processo, implementei uma mudança face ao algoritmo original, esta mudança

consiste na inclusão de apenas blocos que contenham algum tipo de texto, ou seja, implementei uma função que contasse o número de palavras desse bloco e caso o bloco não tenha qualquer palavras então esse bloco não será adicionado à lista final, eu considero este um passo importante e fundamental visto que muitos dos blocos extraídos não têm qualquer conteúdo apresentado na notícia. Esta verificação permite reduzir imenso o número de blocos extraídos e como tal reduz também o número de blocos comparados na fase seguinte.

A.2.2 Comparação de Blocos

Para a criação do vetor binário de termos é usado o CountVectorizer da biblioteca sklearn[2] sendo fornecido o corpus extraído inicialmente já com o pré-processamento do texto descrito na secção 4.3.2. Através desta biblioteca conseguimos assim obter o vetor binário, no entanto, é necessário explicar que para a formação deste vetor todas as palavras são convertidas para minúsculas e todas as que estão repetidas ficam apenas com uma entrada no vetor.

É importante dizer que o CountVectorizer retorna uma lista de vetores binários, pelo que, tive de criar uma função que me retornasse o índice do vetor binário do bloco dado como argumento nessa tal lista. Esta foi uma barreira não muito fácil pois era necessário saber quantos blocos tinha as várias páginas para assim poder perceber qual era o vetor pretendido para o bloco em questão.

[2]https://scikit-learn.org/stable/index.html