Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Detectando Sites de Entidades em Larga Escala: o Caso de Sites de Imóveis

Duhan Caraciolo Maia Souza

Trabalho de Graduação

Recife
Dezembro de 2016

Universidade Federal de Pernambuco
Centro de Informática

Duhan Caraciolo Maia Souza

Detectando Sites de Entidades em Larga Escala: o Caso de Sites de Imóveis

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Prof. Luciano de Andrade Barbosa*

Recife
Dezembro de 2016

Agradecimentos

Primeiramente, gostaria de agradecer a todos os funcionários do Centro de Informática da Universidade Federal de Pernambuco, por fazerem dele um centro de excelência. Durante toda minha graduação, eu tive a oportunidade de ter contato com professores, funcionários e alunos que amam o que fazem, o que contribuiu diretamente para que eu pudesse alcançar meus sonhos e objetivos.

Agradeço ao professor Luciano de A. Barbosa por me orientar neste trabalho e estar sempre disposto a me ajudar e tirar dúvidas. Sem ele este trabalho não seria possível.

Agradeço a todos os docentes com quem tive oportunidade de aprender. Em especial, agradeço ao professor Pedro M. Manhães de Castro, por sempre nos desafiar em suas aulas, e ao professor Paulo G. S. da Fonseca que, além de ter sido de grande importância para o meu aprendizado, sempre esteve disposto a compartilhar sua experiência e me ajudar em algumas decisões.

Por quatro anos da minha graduação participei da Maratona de Programação, programa extracurricular oferecido pelo Centro de Informática. Com a Maratona eu pude conhecer pessoas incríveis, com quem obtive um grande conhecimento, agradeço a todas elas. Particularmente, agradeço às professoras Katia Guimarães e Liliane Salgado, por estarem à frente do programa enquanto participei, e a todos que participaram de algum time comigo: Felipe Andrade, Gustavo Stor, Lucas Lima, Mário Henrique e Mateus Moury.

Agradeço aos amigos que fiz ao longo da minha graduação. Pessoas extremamente inteligentes que sempre estiveram dispostas a me ajudar. Tenho certeza que sem elas meu rendimento acadêmico não teria sido o mesmo. Especialmente, gostaria de agradecer a Bertha, Eduardo, Guilherme, João Pedro, Larissa, Leonardo, Lucas Lima, Lucas Netto, Maria Gabriela, Marina, Mateus, Rafael Acevedo, Rafael Francisco, Raíssa e Vinícius, com os quais, desde o começo da graduação, dividi momentos de tensão pré-prova e compartilhei momentos de alegria.

Agradeço a todas as pessoas que passaram por minha vida e ajudaram direta ou indiretamente a eu chegar onde estou. Em especial, agradeço a Álvaro Mitchell, Hugo Lima, José Guilherme, Lucas Halliday e Nathália Oliveira, amigos que sei que posso contar sempre.

Por fim, mas especialmente importante, agradeço à toda a minha família. Por sempre terem acreditado em mim e me apoiarem em todas as minhas decisões. Agradeço principalmente ao meu sobrinho, Caíque, por me fazer rir nos momentos mais inesperados; à minha irmã, Thaiane, por cozinhar magnificamente; e aos meus pais, Dayse e Rivaldo, por serem exemplos de pessoas nos quais eu posso me inspirar.

Happiness can be found, even in the darkest of times, if one only remembers to turn on the light.

—J.K. ROWLING

Resumo

A *World Wide Web* é um repositório que contém inúmeras páginas com diversos tipos de informação. Algumas dessas páginas contêm informação sobre instâncias de entidades estruturadas com seus atributos e valores associados. Por exemplo, uma página que contém informação sobre um computador informando seus atributos como seu sistema operacional e seu processador. O principal objetivo deste trabalho é detectar, efetivamente e eficientemente, se um site da *Web* contém páginas de entidades estruturadas no domínio de imóveis. Com o auxílio de um coletor focado que navega pela *Web* explorando a vizinhança do grafo de sites já conhecidos, utilizamos o detector para guiá-lo em encontrar inúmeros sites de imóveis na *Web*. Uma vez que esses sites sejam localizados, pode-se, por exemplo, fazer uma coleta posterior dentro deles para a extração da informação das entidades e criar uma base de dados com o conteúdo.

Palavras-chave: recuperação de informação, coleta de página, coletor focado, classificação de página, *backlinks*

Abstract

The World Wide Web is a repository with innumerous pages of all kind of information. Some of these pages contain information about instances of structured entities along with its attributes and associated values. For example, a webpage that contains information about a laptop showing its attributes, such as its operational system and processor. The primary goal of this work is to detect, effectively and efficiently, whether a website contains pages with structured entities in the real estate domain. With the aid of a focused crawler that crawls through the Web graph exploring the neighborhood of already known sites, we use the detector to guide it in finding innumerous real estate sites across the Web. Once these sites are located, it is possible to, for example, do a follow-up crawling inside them to extract the information about the entities and create a database with them.

Keywords: information retrieval, crawling, focused crawler, page classification, backlinks

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

A *World Wide Web* é um repositório com bilhões de páginas[1] em diversos tópicos e em constante modificação. Muitas dessas páginas contêm conteúdo de entidades estruturadas [CHM11]. Páginas com entidades estruturadas possuem informações sobre uma determinada entidade contendo atributos e valores associados a esses atributos. Na Figura 1.1, apresentamos um exemplo de parte de uma página que contém características de um celular como sua marca, cor, memória, etc.

Figura 1.1 Exemplo de página com informação estruturada.

informações técnicas

Código	124126254
Marca	ASUS
Cor	Prata
Tipo de Chip	Micro Chip
Quantidade de Chips	Dual Chip
Memória Interna	32GB
Memória RAM	4GB

Fonte: http://www.americanas.com.br/produto/124126254

O conteúdo de entidades estruturadas na *Web* tem sido usado por diversas aplicações. Por exemplo, engenhos de busca têm usado esses dados para adicionar informação estruturada de uma dada consulta às suas páginas de respostas; empresas como Factual[2] têm comercializado dados estruturados coletados de páginas *Web*, etc.

Chamamos um site da *Web* de site de entidades caso ele contenha páginas de entidades. Por exemplo, o site Peixe Urbano[3] possui diversas páginas de entidades para shows, restaurantes, hotéis, entre outros.

A motivação para este trabalho vem da ideia de se construir uma base de dados com informações de entidades estruturadas contidas na *Web*. Para a construção dessa base de dados

[1] http://www.worldwidewebsize.com/
[2] https://www.factual.com/
[3] https://www.peixeurbano.com.br/

é necessário possuir as entidades, que estão contidas em diversos sites de entidades pela *Web*. Por isso, o primeiro passo para se alcançar esse objetivo é encontrar sites de entidades.

Devido ao grande volume de dados na *Web*, encontrar um grande número de sites manualmente em um determinado domínio é inviável. Para isso, este trabalho propõe um detector de sites de entidades que pode ser utilizado em um *focused crawler* que *automaticamente* e em *larga escala* encontra sites de entidades em um determinado domínio.

O detector é responsável por decidir se um site da *Web* é ou não de entidades. Um dos desafios de se desempenhar essa tarefa é que existe uma grande variedade de sites de entidades. É necessário que o detector diferencie sites de entidades dos sites que não são de entidades, assim como seja capaz de diferenciar sites de entidades de domínios diferentes do desejado. Outro desafio é a quantidade de páginas do site que o detector precisa visitar para gerar uma decisão. Poucas páginas devem ser visitadas.

Neste trabalho, focamos na construção de um detector de sites de imóveis em português. O detector precisa ser efetivo: sites detectados como de imóveis devem ser de imóveis e sites detectados como não sendo de imóveis não devem ser; e eficiente: o detector deve visitar poucas páginas do site para tomar sua decisão, de forma a não sobrecarregá-lo. Uma vez com o detector implementado, podemos utilizá-lo em um *focused crawler* que navega pela *Web* à procura de sites de imóveis em português.

Este trabalho está estruturado da seguinte maneira. O capítulo 2 contém os fundamentos básicos para este trabalho, assim como a descrição de alguns dos trabalhos relacionados ao proposto aqui. O capítulo 3 descreve a solução proposta para a detecção dos sites. Em seguida, o capítulo 4 mostra a avaliação do detector, experimentos feitos sobre os dados coletados e explica como os sites foram localizados na *Web*. Já no capítulo 5, mostramos alguns dos possíveis trabalhos futuros.

CAPÍTULO 2

Fundamentos

Neste capítulo explicamos alguns fundamentos básicos para o trabalho, como o que são *focused crawlers* e aprendizado supervisionado. Em seguida, mencionamos alguns trabalhos relacionados e fazemos um resumo sobre os mesmos.

2.1 Focused Crawlers

Crawlers são aplicações utilizadas para coletar páginas da Web. Eles são utilizados, por exemplo, por engenhos de busca para criar e atualizar suas bases de páginas.

O processo de coleta pode ser descrito da seguinte maneira. Inicialmente, um conjunto inicial de links é dado como entrada para o *crawler*. Esses links são inseridos numa lista de *links* a serem visitados pelo coletor, chamada de fronteira. O *crawler* então seleciona uma URL da fronteira, visita-a e armazena-a em sua base de páginas, e insere seus *outlinks* na fronteira, continuando esse processo até que alguma condição de parada seja alcançada, como o número de páginas visitadas ou a fronteira esteja vazia. [CGMP98]

Existem alguns aspectos que devem ser considerados na implementação de um *crawler*. São eles: (1) Seleção de conteúdo: a ordem em que as páginas da fronteira são selecionadas afeta o desempenho do crawler [BYCMR05]; (2) Sobrecarga de sites: é importante que o *crawler* não sobrecarregue os sites ao visitar suas páginas; (3) Conteúdo adversário: sites contendo spam, por exemplo, possuem conteúdo que não é útil para o usuário; e (4) Escalabilidade: como existe um grande número de páginas na *Web*, o coletor precisa ser escalável para ter uma boa cobertura da *Web*.

Um *crawler* que visita páginas sem levar em consideração o tópico delas é chamado de genérico. Esses são os crawlers utilizados por engenhos de busca de caráter genérico como Google[1] e Bing[2].

Focused crawlers, por outro lado, são coletores que têm como objetivo encontrar páginas em um determinado tópico. Para isso, analisam as páginas visitadas para decidir quais *links* da fronteira são os mais prováveis de serem relevantes para o tópico selecionado, evitando assim páginas irrelevantes.

[1] https://www.google.com/

[2] http://www.bing.com/

2.2 Aprendizado Supervisionado

Aprendizagem de máquina é um campo da computação que tenta dar ao computador a capacidade de aprender sem ser explicitamente programado[3]. Existem várias categorias de aprendizagem de máquina, entre elas está a aprendizagem supervisionada.

Aprendizagem supervisionada [Kot07] tem como objetivo inferir uma função a partir de um conjunto de exemplos rotulados, para predizer resultados sobre novos exemplos. Nesse tipo de aprendizado, cada exemplo é composto de um par contendo características do objeto a ser avaliado e o resultado esperado que a função produza para aquele objeto.

As características (*features*) escolhidas para representar o objeto dependem do problema a ser resolvido, e servem para que o algoritmo consiga analisar os objetos. É importante que *features* informativas e discriminativas sejam escolhidas. Por exemplo, para o problema de identificar se um e-mail é spam podem ser utilizadas algumas *features* como a estrutura do e-mail, a frequência de determinadas palavras e a quantidade de erros gramaticais.

Existem dois tipos de aprendizado supervisionado, a depender do tipo do valor gerado pela função. Caso o valor gerado seja numérico, então o problema é chamado de regressão. Já se o valor gerado for categórico, o problema é chamado de classificação.

É esperado que a função gerada produza os resultados esperados inclusive para dados que não estão no conjunto de treinamento. Contudo, a função inferida pode ter alguns problemas. É possível que ela não produza o resultado esperado nem para os dados do conjunto de treinamento. Nesse caso, dizemos que a função possui *underfitting* dos dados e é sugerido adicionar mais *features* dos objetos, pois as que estão sendo usadas não são informativas o suficiente. Outra possibilidade é que a função produza os resultados esperados para os dados do conjunto de treinamento, mas não para os fora dele. Nesse caso, dizemos que a função possui *overfitting* dos dados, ou seja, as *features* utilizadas não generalizam para os dados que não pertencem ao treinamento, e é sugerido que algumas sejam removidas.[4]

2.3 Trabalhos Relacionados

Devido à grande utilidade de *focused crawlers*, vários trabalhos têm sido propostos aplicando-os a vários problemas diferentes. Nesta seção, falamos sobre alguns trabalhos relacionados ao proposto aqui.

Por exemplo, no trabalho proposto por [BBS11] foi criado um detector de sites bilíngues. Sites bilíngues são sites que possuem a mesma informação em dois idiomas diferentes. Encontrá-los é útil para a criação de dicionários bilíngues, a partir do alinhamento dos textos encontrados. O detector proposto por [BBS11] identifica um site como bilíngue se a raiz do site possuir *links* para conteúdo nos idiomas procurados, ou se a raiz estiver em um dos dois idiomas e possuir um *link* para o conteúdo no outro idioma desejado. Para evitar visitar muitas páginas desnecessariamente, apenas as páginas consideradas relevantes por um classificador de *links* são visitadas. O classificador de *links* utiliza aprendizado supervisionado com as se-

[3] https://en.wikipedia.org/wiki/Machine_learning

[4] http://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

guintes *features*: (1) palavras na *URL*; (2) palavras na âncora; (3) palavras ao redor da âncora; (4) *image alt*; e (5) *image src*, e a saída é a probabilidade do *link* apontar para uma página de um dos idiomas. As *features* (4) e (5) são atributos da *tag img*, caso a âncora possua uma imagem (normalmente da bandeira do país). Para determinar o idioma da página é utilizado um classificador de idiomas.

Já no trabalho proposto por [Bar16] foi criado um detector de sites de fóruns. O objetivo desse detector é determinar se um site da *Web* possui um fórum e, em caso positivo, encontrar o *link* de entrada para o mesmo. A estratégia de detecção é baseada em duas observações: (1) sites de fórum normalmente têm uma página de entrada para o conteúdo do fórum, que mostra um resumo das discussões atuais; e (2) essas páginas de entrada são a própria raiz do site ou são acessíveis a partir da raiz. Páginas de entrada para fóruns possuem um vocabulário em comum independente do domínio do fórum. Portanto, foi criado um classificador que utiliza como *features* as palavras presentes nas páginas. O classificador é utilizado na detecção da seguinte maneira. Primeiramente, a raiz do site é enviada ao classificador. Se for classificada como relevante, o site é identificado como de fóruns. Se a raiz não for classificada como relevante, as páginas que ela aponta são visitadas e enviadas ao classificador. Se alguma dessas páginas for classificada como relevante, então o site é identificado como de fórum. Se nenhuma dessas páginas for classificada como relevante, o site é identificado como não-fórum. Para evitar visitar muitas páginas apontadas pela raiz, apenas páginas que possuem palavras indicativas na *URL* e âncora são visitadas. Exemplos de palavras indicativas são `forum` e `community`.

CAPÍTULO 3

Detector

Neste capítulo descrevemos como foi construído o detector de sites de imóveis em português. O detector é o módulo responsável por determinar se um site da *Web* é relevante. No nosso caso, dado qualquer site da *Web*, queremos saber se ele é ou não um site de imóveis em português. Um site é considerado de imóveis se ele possui entidades de imóveis em suas páginas.

Os dois requisitos básicos para o detector são: (1) acurácia alta: é importante que os sites identificados como de imóveis realmente sejam, devido à possível coleta posterior das entidades nele, assim como os sites identificados como não sendo de imóveis de fato não sejam, uma vez que queremos coletar o máximo de sites possível; e (2) custo baixo: é desejável que a detecção de um site visite o menor número possível de páginas deste, para não sobrecarregá-lo.

O detector é dividido em três sub-módulos: (1) o Priorizador de *Links*; (2) o Classificador de Índice; e (3) a Busca. O Priorizador de *Links* é responsável por atribuir uma prioridade de visitação à um *link*, a fim de visitar páginas de índice primeiro, terminando a detecção com poucas páginas visitadas. O Classificador de Índice é o sub-módulo responsável por dado uma página do site classificá-la como sendo uma página de índice de imóveis ou não. Já a Busca é o sub-módulo responsável por navegar pelas páginas do site, eficientemente, de forma a decidir se o site é de imóveis. Para tal, são utilizados os outros dois sub-módulos: o Priorizador é utilizado para saber quais as páginas que devem ser visitadas primeiro, a fim de encontrar páginas de índice o quanto antes, e o Classificador é utilizado para saber se a página que está sendo visitada é uma página de índice, de fato, ou não.

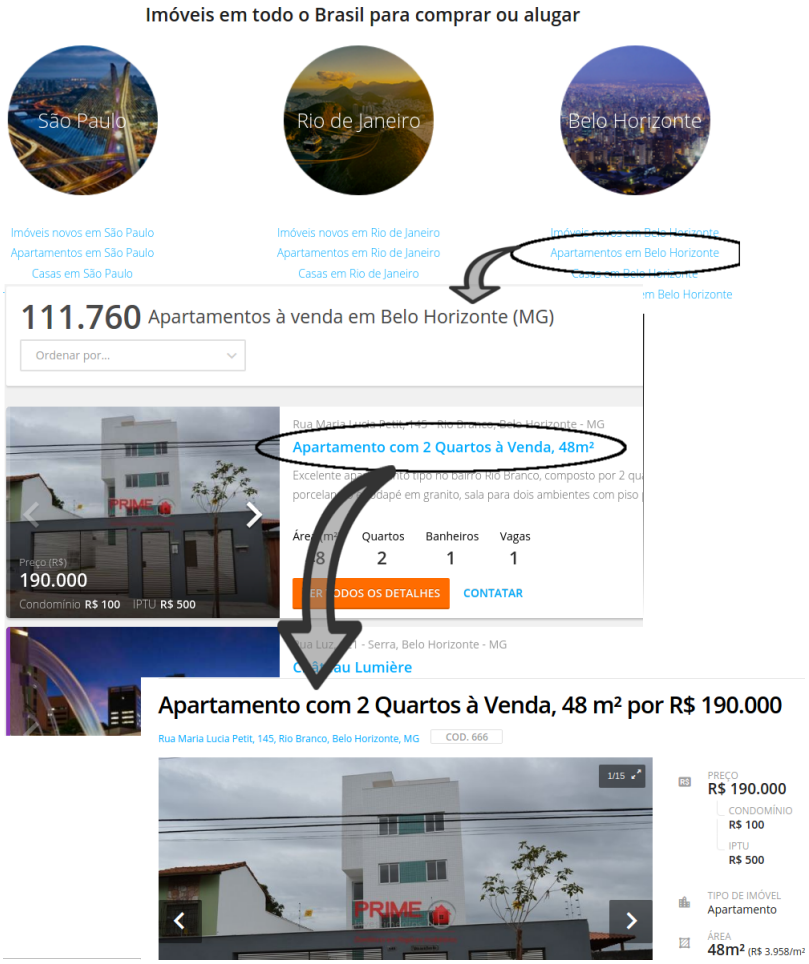
3.1 Arquitetura

Dado qualquer site da *Web* queremos identificá-lo como sendo um site de imóveis ou não. Além dessa detecção, queremos ser capazes de encontrar um *link* de entrada para o conteúdo de imóveis no site, ou seja, a região do site onde estão as páginas de imóveis. O principal desafio é visitar poucas páginas do site.

Uma solução seria visitar as páginas do site até encontrar alguma página que contenha uma entidade de imóveis. Um problema dessa abordagem é que para sites que não contenham entidades a busca pode executar indefinidamente sem que encontre uma página de entidades e, mesmo que o site contenha as entidades desejadas, a busca pode visitar muitas páginas desnecessariamente até encontrá-las. Em ambos os casos, há um grande desperdício de recursos. Um segundo problema é que páginas de entidade estão muito profundas no site, i.e. muitas páginas são visitadas para encontrá-las. A figura Figura 3.1 mostra o fluxo de visita para encontrar uma página de entidade: começando na raiz do site, é necessário visitar uma página, para só depois

visitar a página de entidade. Outro problema é que a partir de uma página de entidade não é possível encontrar facilmente um *link* de entrada para o conteúdo de imóveis no site.

Figura 3.1 Fluxo de visita da raiz do site até uma página de entidade.

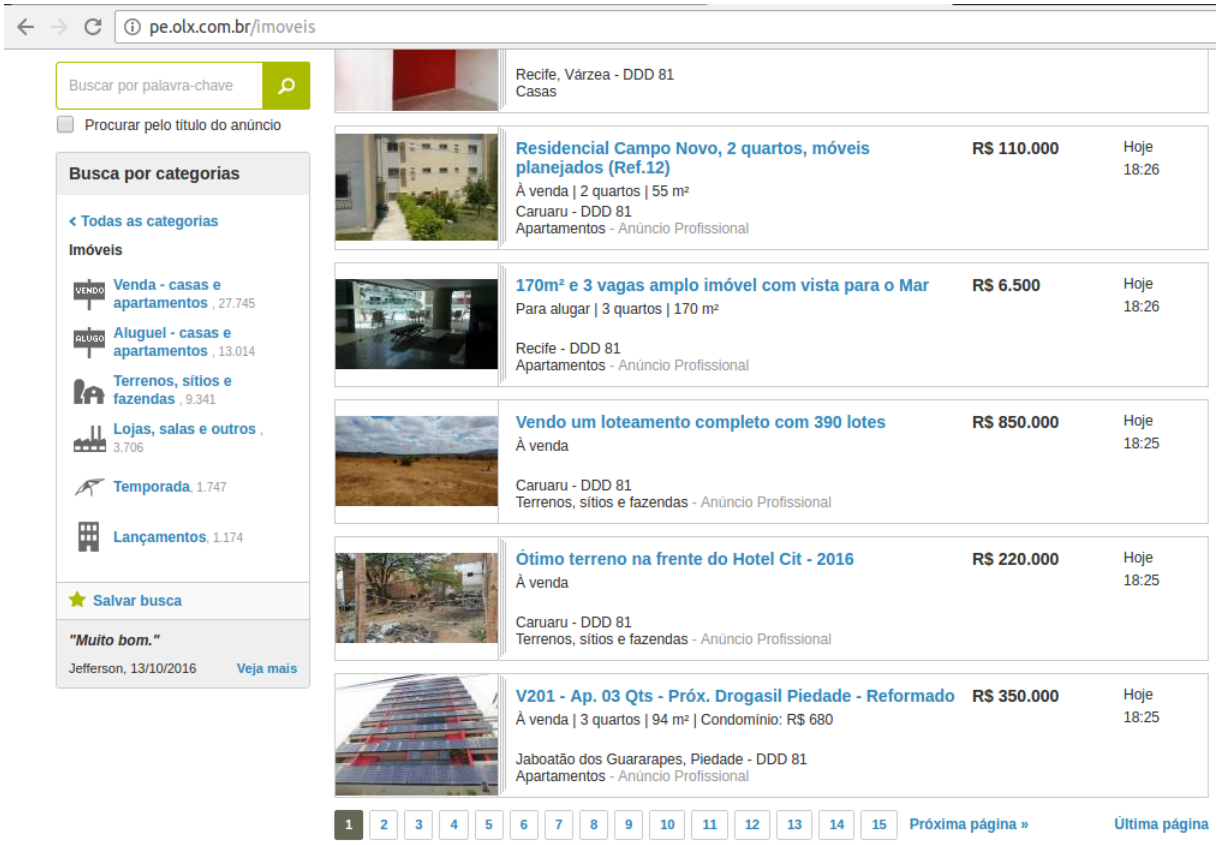


Fonte: http://www.vivareal.com.br/

Para resolver os problemas da solução descrita, procuramos por páginas de índice ao invés de páginas de entidade. No fluxo da Figura 3.1, a página central é uma página de índice. Páginas de índice possuem *links* que apontam para várias páginas de entidade e normalmente possuem paginações, como pode ser visto em detalhes na Figura 3.2, logo, temos acesso à um *link* de entrada para o conteúdo de imóveis. Mas, mais importante ainda, páginas de índice estão mais perto da raiz do site do que páginas de entidade, portanto, uma busca superficial no site é o suficiente para encontrá-las.

Trabalhos anteriores em detecção de sites bilíngues [BBS11] ou de fóruns [Bar16] envolvem visitar, no pior caso, as páginas que são apontadas pela *home page* do site. Para aqueles domínios isso faz sentido. Porém, as informações necessárias para identificar sites de entida-

Figura 3.2 Exemplo de página de índice de imóveis.



Fonte: http://pe.olx.com.br/imoveis

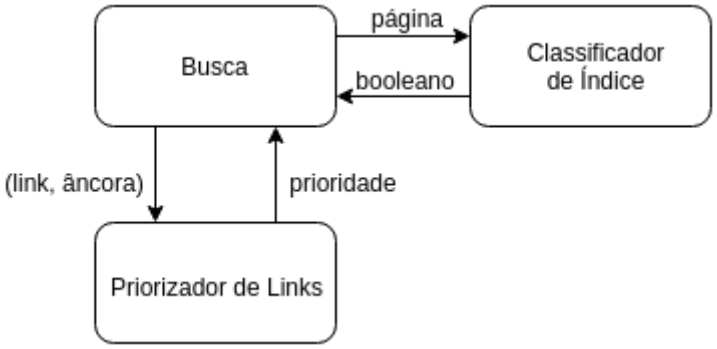
des nem sempre estão a apenas um passo da raiz do site. Por exemplo, sites como OLX [1] e Super Classificados [2] necessitam que uma página de algum estado do Brasil seja visitada para só depois ser possível visitar a página de índice do domínio desejado. Por esse motivo, não podemos restringir a visita às páginas que se localizam no nível 1 do site (a *home page* tem nível 0), como feito em [BBS11] e [Bar16].

A arquitetura geral da solução pode ser vista na Figura 3.3. O módulo principal é a Busca. Dado a raiz do site, são extraídos os *links* que ela aponta, juntamente com suas âncoras. O Priorizador de link, baseando-se na âncora e na *URL*, calcula a prioridade de visitação que um *link* possui de forma que *links* de páginas de índice sejam visitados primeiro, e a Busca utiliza essas prioridades para encontrar páginas relevantes de forma mais eficiente. Quando uma página é visitada, a mesma é passada para o Classificador de Índice, que informa se a página é ou não de índice de imóveis. Em caso positivo, a Busca é encerrada e o site é detectado como

[1] http://www.olx.com.br
[2] http://www.superclassificados.com/

de imóveis. Em caso negativo, os *links* e âncoras da página são extraídos e têm suas prioridades calculadas. Para evitar visitar o site indefinidamente existe um limite máximo de páginas que podem ser visitadas.

Figura 3.3 Arquitetura da solução proposta.



Fonte: Elaborada pelo autor.

3.2 Classificador de Índice

Como dito anteriormente, buscamos por páginas de índice e, portanto, precisamos ser capazes de saber se uma página *Web* é ou não uma página de índice de imóveis. Assim, caso o site possua pelo menos uma página de índice de imóveis ele é identificado como um site relevante para o detector, ou seja, um site de imóveis.

Na resolução desse problema utilizamos aprendizagem de máquina supervisionada, onde os objetos são páginas da *Web* e o resultado esperado é se a página é de índice de imóveis ou não. Para o conjunto de treinamento rotulamos manualmente 520 páginas da *Web*, sendo 151 páginas de índice de imóveis (relevante) e 369 páginas que não são índice de imóveis (não-relevante). As páginas foram extraídas de 33 sites de imóveis diferentes, além de 26 sites que não são de imóveis, como http://www.uol.com.br e http://exame.abril.com.br.

Existem dois tipos de *features* para o classificador: (1) *features* gerais de páginas de índice que independem do domínio das entidades; e (2) *features* específicas para o domínio em questão.

3.2.1 Features Gerais

Por definição, páginas de índice contêm vários *links* para páginas de entidade. Portanto, um bom indicativo que uma página é de índice é o número desses *links* que ela possui. Dessa forma, seja `P` a página que deseja-se classificar como índice, queremos saber quantos *links* de `P` apontam para páginas de entidade. Uma solução seria criar um classificador de páginas de entidade e, para cada *link* que `P` aponta, visitá-lo e classificar a página como de entidade ou

não. Infelizmente, essa solução visita muitas páginas, que é uma tarefa bastante custosa, além de sobrecarregar o site desnecessariamente.

Para uma solução mais eficiente, analisamos as páginas do conjunto de treinamento e observamos que os *links* que apontam para as páginas de entidade são bem semelhantes. Por exemplo, na página de índice da Figura 3.2 existem os seguintes *links* para páginas de entidade:

- http://pe.olx.com.br/grande-recife/imoveis/vendo-casa-258709656
- http://pe.olx.com.br/grande-recife/imoveis/3-quartos-258245647
- http://pe.olx.com.br/grande-recife/imoveis/casa-em-jardim-paulista-alto-270991391

Pode-se ver que todos os 3 *links*, assim como os outros *links* de páginas de entidade, possuem `grande-recife/imoveis` como prefixo em comum, bem como estão hospedados no mesmo domínio da *Web*. Portanto, todas as *URLs* da página, que estão hospedadas no mesmo domínio da página atual, são extraídas e agrupadas por seus prefixos (nesse caso, agrupamos por `grande-recife`). Criando, assim, vários conjuntos de *URLs* disjuntos.

Em posse dos conjuntos disjuntos são criadas as duas primeiras *features* para o classificador. A primeira é o tamanho do maior conjunto, pois representa bem a quantidade de entidades que o índice possui; a segunda é a quantidade total de *links* que apontam para páginas do mesmo domínio. A separação entre o maior conjunto e a quantidade total é devido ao fato que existem páginas de índice que possuem mais de um agrupamento nas *URLs*. Por exemplo, em uma página de índice do VivaReal[3], existem os seguintes *links* para páginas de entidade:

- https://www.vivareal.com.br/imoveis-lancamento/villaggio-shangrila-7223/
- https://www.vivareal.com.br/imovel/apartamento-2-quartos-vila-nova-bairros-campinas-com-garagem-60m2-venda-RS320000-id-74633216/

Podemos ver que eles possuem prefixos diferentes: `imoveis-lancamento` e `imovel`; todavia, esses padrões são repetidos nos outros *links* da página, gerando dois conjuntos que representam entidades. Por esse motivo, além de considerar o tamanho do maior conjunto, também consideramos a quantidade total.

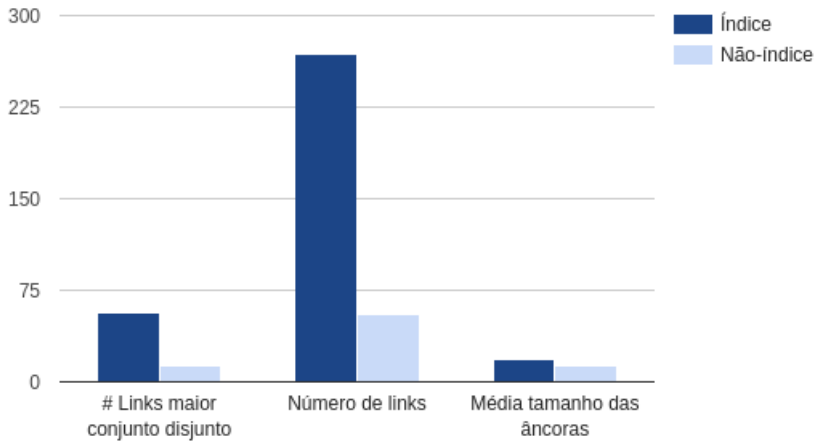
A última *feature* geral é a média dos tamanhos das âncoras. Podemos ver na Figura 3.2 que as âncoras para páginas de entidade têm um tamanho semelhante entre si e maiores que das outras âncoras, por esse motivo, calculamos a média de todas as âncoras da página e utilizamos como *feature*.

Uma comparação das médias dos valores das *features* discutidas entre páginas de índice e não-índice pode ser vista na Figura 3.4. Podemos ver que o número de *links* no maior conjunto disjunto e o número de *links* do mesmo domínio são bem discriminativos.

Vale a pena notar que todas essas *features* podem ser calculadas apenas com as informações contidas na página que o classificador está avaliando, não sendo necessário visitar nenhuma outra página da *Web*, economizando banda e não sobrecarregando o site, bem como não dependem do domínio das entidades em questão.

[3] https://www.vivareal.com.br/

Figura 3.4 Valores das *features* gerais de páginas de índice e não-índice.



Fonte: Elaborada pelo autor.

3.2.2 Features Específicas

As *features* gerais funcionam bem para separar as páginas de índice das demais. Porém, existem muitas páginas de índice na *Web* nas quais as entidades não são do domínio desejado, inclusive dentro de um mesmo site. Por exemplo, o site possui páginas de índice de veículos, eletrônicos, entre outros. Para isso, temos que criar *features* que sejam específicas do domínio, de modo que o classificador possua uma alta acurácia.

Inicialmente, observamos que as palavras utilizadas nos títulos das páginas são um bom indicativo do que o site contém, como podemos ver pelo *word cloud* da Figura 3.5, gerado a partir das páginas relevantes do conjunto de treinamento. Na maioria delas encontramos as palavras `imoveis`, `imobiliaria` e `venda` em seus títulos. Portanto, a primeira *feature* específica do domínio é extrair as palavras do título da página. Podemos ver que `pernambuco` aparece bastante, isso deve-se ao fato de várias páginas do conjunto de treinamento serem de sites que contêm entidades do estado de Pernambuco. Porém, como mostrado na Subseção 4.2.2, o classificador é genérico para os demais estados do Brasil.

Outra *feature* interessante são as palavras usadas na *URL* da página, não considerando as palavras do *host* do site (pois essas palavras se repetem em todas as páginas do site, relevantes ou não). Além de ser um bom indicativo que a página é de índice, também possuem palavras específicas do domínio. Podemos ver na Figura 3.6 um *word cloud* gerado a partir das palavras usadas nas *URLs* das páginas relevantes do conjunto de treinamento que `busca`, `imoveis` e `venda` são frequentes.

Figura 3.5 *Word cloud* das palavras mais utilizadas nos títulos das páginas de índice de imóveis.

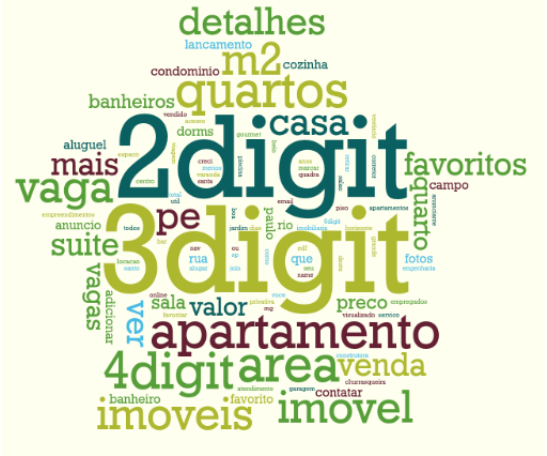
Fonte: Elaborada pelo autor.

Figura 3.6 *Word cloud* das palavras mais utilizadas nas *URLs* das páginas de índice de imóveis.

Fonte: Elaborada pelo autor.

As palavras utilizadas no corpo da página também são bastante úteis para identificar o domínio em questão. Na Figura 3.7 temos um *word cloud* das palavras que mais aparecem no corpo das páginas. Podemos ver que palavras relacionadas ao domínio como `apartamento`, m^2 e `quartos` são frequentes.

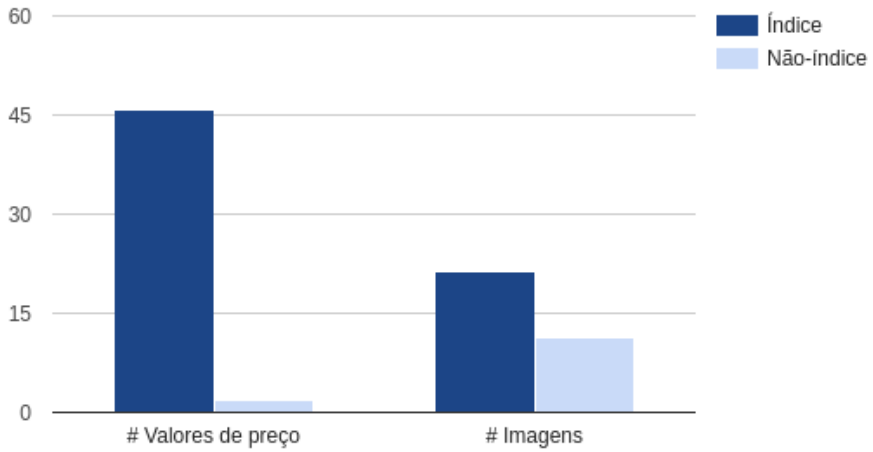
Figura 3.7 *Word cloud* das palavras mais utilizadas no corpo das páginas de índice de imóveis.



Fonte: Elaborada pelo autor.

É importante notar que tanto no título, como na *URL* e no corpo da página, existem valores numéricos. O valor do número dificilmente tem alguma informação útil, por isso, substituímos cada número de `x` dígitos por uma palavra `xDIGIT`. Também transformamos todas as letras em minúsculas.

Figura 3.8 Valores das *features* específicas de páginas de índice e não-índice.



Fonte: Elaborada pelo autor.

Por último, mas não menos importante, outra característica a se notar é que páginas de índice, no domínio de produtos, normalmente possuem os preços de venda/aluguel das entidades, assim como fotos dos produtos. Logo, também extraímos quantos valores de preço e quantas

imagens aparecem na página, criando mais duas *features* para o classificador: número de imagens e número de preços. Uma comparação entre a média dos valores dessas duas *features* entre páginas de índice e não-índice pode ser vista na Figura 3.8.

3.3 Priorizador de Links

A priorização de *links* tem como objetivo priorizar a visita das páginas que têm maior chance de serem classificadas como relevante pelo classificador, ou seja, de serem uma página de índice de imóveis. A chance calculada pelo Priorizador de *Links* é um número inteiro que cria uma ordem total sobre os *links* a serem visitados. Isso é útil pois antes de visitar a página, consumindo banda, temos um indicativo de que a página com maior prioridade é a página que tem a maior chance de ser relevante e, portanto, merece ser visitada primeiro. Com isso, é possível não visitar páginas irrelevantes para o classificador, economizando banda e tempo.

Como solução, poderíamos utilizar aprendizagem supervisionada, como feito na seção 3.2. Porém, coletar os dados para o conjunto de treinamento seria muito trabalhoso, uma vez que não podemos apenas considerar as âncoras e *URLs* que apontam para páginas de índice: temos que considerar todas as âncoras e *URLs* que pertencem a um caminho da raiz do site até uma página de índice. Por esse motivo, criamos vários filtros para saber a prioridade que um *link* tem para ser visitado, quanto maior a prioridade maior a chance da página ser de índice de imóveis. Devido à grande variedade da estrutura e palavras usadas nos sites da *Web*, temos que saber se os filtros escolhidos generalizam bem. Utilizamos os sites usados no conjunto de treinamento do classificador para decidir quais filtros usar.

Existem dois tipos de sites de imóvel: (1) sites que apenas contêm entidades de imóvel; e (2) sites que contêm outros tipos de entidades, além das de imóveis.

Para decidir quais filtros usar, vimos quais palavras mais aparecem nas âncoras que levam às páginas de índice, assim como quais dessas palavras também levam à páginas que não são de índice no domínio desejado ou não são de índice. Por exemplo, `imóveis` é uma palavra que aparece bastante nas âncoras que levam às páginas de índice, `comprar` também, porém, esta última também leva à páginas de índice de outros domínios. Por esse motivo, *links* que possuem `imóveis` na âncora devem ser visitados antes de *links* que possuem `comprar`. Também tivemos que analisar os *links* que não levam à páginas de índice. *Links* que possuem `login`, `logout` ou `cadastro` na âncora não levam à páginas de índice e, portanto, devem ter a prioridade na visita diminuída. Além das âncoras, também vimos quais palavras em comum eram usadas nas *URLs* que levavam à páginas de índice de imóveis e às que não eram.

Uma outra característica, como podemos ver na Figura 3.2, é que âncoras de páginas de entidade podem possuir informação de hora, o que não acontece em âncoras de página de índice. Portanto, *links* que possuem informações de hora devem ter uma prioridade de visita menor.

Por fim, pode-se ver que âncoras para páginas de índice tem tamanho pequeno, sendo normalmente uma ou duas palavras: `Temporada` e `Lançamentos`. Logo, o tamanho da âncora também afeta a prioridade de visita.

Unindo todas essas observações, criamos filtros que dado a âncora e a *URL* de um *link* atribuem um número inteiro que representa a chance do *link* apontar para uma página de índice

de imóveis. Existem filtros positivos e filtros negativos.

Todas as palavras em *URLs* de páginas de índice também apareciam nas demais, normalmente páginas de entidade. Por esse motivo, só utilizamos palavras de *URLs* para filtros negativos. Os filtros positivos só levam em consideração as palavras presentes nas âncoras dos *links* e se a página atual é a raiz do site. Os filtros negativos consideram as palavras presentes nas âncoras e nas *URLs*, o tamanho da âncora, se a âncora contém um valor indicando hora e se a âncora contém um valor de preço.

Em alguns dos sites analisados, a *home page* possui *links* para acessar as informações por estado brasileiro, para só depois acessar as informações das entidades. Por esse motivo, o primeiro filtro positivo checa se a âncora é igual à sigla de algum estado brasileiro e se a página atual é a raiz do site e, em caso positivo, a prioridade do *link* é incrementada em 3. Os outros filtros positivos consideram se a âncora possui alguma das seguintes palavras e em caso positivo incrementam a prioridade do *link* no valor mostrado no modelo (palavra, valor). São eles: (imóveis, 6), (apartamentos, 6), (casas, 6), (comerciais, 6), (rurais, 6), (comprar, 5), (alugar, 5), (empresarial, 5), (vender, 4), (imóvel, 4), (apartamento, 4), (casa, 4), (busca, 4), (ver todos, 4) e (lançamentos, 4).

O primeiro filtro negativo verifica se o tamanho da âncora é maior que 13 e, em caso positivo, a prioridade do *link* é decrementada em 2. O segundo filtro checa se a âncora contém alguma hora e, caso sim, a prioridade é decrementada em 1. O terceiro verifica se a âncora ou *URL* contém preço e, em caso positivo, a prioridade é decrementada em 2 (4 caso apareça em ambas). Os demais filtros negativos checam se a âncora ou *URL* contêm as seguintes palavras e, caso sim, decrementam a prioridade no valor mostrado. São eles: (about, 10), (anuncie, 10), (faq, 10), (support, 10), (suporte, 10), (copyright, 10), (login, 10), (logout, 10), (password, 10), (privacy, 10), (privacidade, 2), (cadastro, 5), (anúncio, 3), (contato, 5), (sair, 5), (conosco, 3), (cartão, 8), (cartões, 8), (banner, 8), (encomenda, 8) e (cadastrar, 8). É possível notar que mesmo que os sites analisados tenham sido em português, muitas das palavras usadas nas *URLs* são em inglês.

Algoritmo 1: PrioridadeLink

```
   entrada: link, ancora, estaNaRaiz, filtrosPositivos, filtrosNegativos
   saída  : Prioridade do link
 1 prioridade ← 0
 2 for filtro in filtrosPositivos do
 3     if filtro.aceita(ancora, estaNaRaiz) then
 4         prioridade ← prioridade + filtro.valor
 5 for filtro in filtrosNegativos do
 6     if filtro.aceita(ancora) then
 7         prioridade ← prioridade - filtro.valor
 8     if filtro.aceita(link) then
 9         prioridade ← prioridade - filtro.valor
10 return prioridade
```

Todos os valores utilizados foram escolhidos empiricamente. Por exemplo, `comprar` tem

um valor menor que `imóveis` pois é possível comprar entidades diferentes de imóveis. Já se a âncora contém `imóveis` é muito provável que a página que ela aponta contenha imóveis.

Como pode ser visto no Algoritmo 1, inicialmente, a prioridade do *link* é 0 (linha 1). Após isso, a âncora é passada pelos filtros positivos, e, caso o filtro a aceite, a prioridade do *link* é incrementada em um valor específico daquele filtro (linhas 2 à 4). Em seguida, a âncora e a *URL* são passadas pelos filtros negativos, caso o filtro aceite, a prioridade do *link* é decrementada em um valor específico daquele filtro (linhas 5 à 9).

3.4 Busca

A Busca é o sub-módulo responsável por navegar pelas páginas do site a fim de decidir se ele é ou não um site de imóveis. Devido à ajuda da priorização dos *links*, a busca utiliza quatro heurísticas para tentar ser mais eficiente na detecção:

1. Um limite máximo de páginas que podem ser visitadas no site.

    - Caso o site não fosse de imóveis e não houvesse um limite de páginas para serem visitadas, a Busca iria continuar visitando páginas indefinidamente. Por isso, escolhemos um valor limite de páginas que podem ser visitadas. Caso o limite seja atingido, a Busca é terminada e o site é identificado como irrelevante. Devido à prioridade dos *links*, é esperado que se as primeiras páginas visitadas não forem relevantes, as demais também não serão.

2. Um limite máximo de páginas filhas que podem ser visitadas a partir de uma página do site.

    - Seguindo o raciocínio da heurística (1), se os *links* de maior prioridade que são apontados por uma página `P` não são classificados como relevante pelo Classificador de Índice, então, o restante dos *links* provavelmente também não serão. Por esse motivo, caso um limite de páginas que são apontadas por `P` forem irrelevantes, nenhuma das páginas restantes que `P` aponta serão visitadas.
    - Existem dois limites distintos. Dependendo se `P` é ou não a raiz do site.

3. *Links* com profundidade 1 só são seguidos caso possuam uma prioridade não-negativa, e *links* que estão em profundidades maior que 1 só são seguidos caso possuam uma prioridade positiva.

    - O Priorizador de *Links* atribui prioridades maiores que 0 para páginas de índice. Se a Busca está na raiz do site (ou seja, os *links* que a página aponta possuem profundidade 1), pode ser que esse *link* leve-a para uma página que aponta para uma página de índice. Se a Busca não está na raiz do site (ou seja, os *links* que a página aponta estão em uma profundidade maior que 1), é esperado que um *link* com prioridade 0 não a leve para uma página de índice.

- *Links* com prioridade negativa dificilmente ajudam a Busca e, por isso, não são visitados.

4. Como páginas de índice estão próximas à raiz do site, apenas páginas até uma determinada distância da raiz são visitadas pela Busca.

    - As páginas de índice estão à uma pequena distância da raiz, por esse motivo, não há sentido em visitar páginas muito profundas no site. E assim, páginas mais próximas à raiz são priorizadas.

Algoritmo 2: Detector

```
    entrada: raiz, classificador, limiteVisita, limiteFilha, limiteFilhaRaiz,
             limiteProfundidade
    saída  : URL de uma página de índice do site ou null
 1  fronteira ← {(url: raiz, referrer: null, prioridade: 0, level: 0)}
 2  paginasVisitadas ← 0
 3  contadorFilha ← Map(URL para Inteiro)
 4  contadorFilha.insereChaveValor((chave: raiz, valor: limiteFilha - limiteFilhaRaiz))
 5  while paginasVisitadas < limiteVisita and fronteira.naoVazia() do
 6      (url, referrer, prioridade, lvl) ← fronteira.maiorPrioridae()
 7      if contadorFilha.valor(referrer) < limiteFilha then
 8          pagina ← download(url)
 9          paginasVisitadas ← paginasVisitadas + 1
10          if classificador.relevante(pagina) then
11              return url
12          contadorFilha.incrementaValor(referrer)
13          if lvl < limiteProfundidade then
14              for (link, ancora) in pagina.extraiURLs() do
15                  p ← PrioridadeLink(link, ancora, lvl)
16                  if p > 0 or (p = 0 and lvl = 0) then
17                      fronteira.ins((url: link, referrer: url, prioridade: p, level: lvl + 1))
18  return null
```

A busca funciona como descrita Algoritmo 2. Inicialmente, a raiz do site é inserida na fronteira (linha 1). Um contador de páginas filhas irrelevantes para cada página do site é criado (linha 3) e o limite de filhas irrelevantes da raiz é setado (linha 4). Enquanto o limite de páginas não tiver sido excedido e houver *links* na fronteira (linha 5), o *link* de maior prioridade é selecionado (linha 6). Se dois *links* tem a mesma prioridade calculada pelo Priorizador de *Links*, o *link* que possui menor distância da raiz é visitado primeiro, pois páginas de índice estão próximas da raiz.

Caso a quantidade de páginas filhas irrelevantes não tenha sido atingida para o pai do *link* atual (linha 7), a página é baixada e enviada ao classificador (linhas 8 à 10). Se a página for

relevante, a busca é encerrada e a *URL* é retornada como sendo o *link* de entrada para o conteúdo de imóveis (linha 11). Se a página não for relevante, o contador de páginas filhas irrelevantes é incrementado (linha 12) e caso a página esteja a uma distância da raiz do site menor que o limite, suas *URLs* são extraídas, as prioridades calculadas e inseridas na fronteira (linhas 13 à 17).

Por fim, se não houver mais *links* na fronteira ou o limite de páginas definido na heurística (1) for atingido, o site é identificado como não sendo de imóveis, retornando-se *null* (linha 18).

É importante lembrar que *URLs* proibidas pelo *robots.txt* do site não são visitadas.

CAPÍTULO 4

Estudo de Caso

Na primeira seção deste capítulo mostramos como foi feita a avaliação do Detector em um conjunto selecionado de sites. Já na segunda seção, utilizamos o Detector em um *focused crawler* para encontrar sites na *Web* e mostramos algumas análises feitas sobre os sites encontrados.

4.1 Avaliação do Detector

Nesta seção mostramos como foi feita a avaliação do detector. Primeiramente mostramos a avaliação feita para o Classificador de Índice, e em seguida mostramos a avaliação do Detector em um conjunto selecionado de sites. Os filtros utilizados foram mencionados na Seção 3.3.

4.1.1 Classificador de Índice

Para a avaliação do Classificador de Índice foi utilizado um conjunto de testes com 201 páginas da *Web*, sendo 100 páginas de índice de imóveis (positivas) e 101 páginas negativas, como páginas de índice que não são de imóveis e páginas de entidades. As páginas foram extraídas de 65 sites de imóveis não presentes no conjunto de treinamento, além de 16 sites da *Web* que não são de imóveis.

Para a construção do classificador foi utilizado a biblioteca Weka[1]. Weka é uma coleção de algoritmos de aprendizagem de máquina para tarefas de mineração de dados que contém, entre outras coisas, inúmeros classificadores já implementados. Treinamos alguns desses classificadores com nosso conjunto de treinamento e os avaliamos com nosso conjunto de testes. Inicialmente, existiam mais de 20000 *features*. Algumas técnicas de *feature selection* foram utilizadas: utilizar palavras que apareciam em pelo menos duas páginas positivas ou duas páginas negativas foi a que obteve o melhor resultado. Após essa filtragem das palavras, sobraram aproximadamente 5000 *features*.

Como pode ser visto na Tabela 4.1, o *SMO* possui o F1 e a acurácia superiores aos demais. Apesar de estarmos mais interessados em uma alta precisão do que uma alta cobertura, a cobertura do *Simple Logistic* é muito baixa para sua precisão valer a pena e, por isso, utilizamos o *SMO* no Detector.

Realizamos uma análise de erros para verificar as limitações do classificador. Por causa das *features* específicias, algumas páginas que possuem conteúdo de imóveis, mas não são de índice, são classificadas como relevante. Na Figura 4.1 temos uma página que possui palavras similares às encontradas em páginas de índice de imóveis, como `compra`, `venda` e `imoveis`.

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

Tabela 4.1 Métricas dos classificadores no conjunto de testes.

	Precisão	Cobertura	F1	Acurácia
J48	0.90	0.54	0.68	0.74
Simple Logistic	0.98	0.54	0.70	0.77
Naïve Bayes	0.78	0.87	0.82	0.81
SMO	0.90	0.97	0.93	0.93

Porém, a página claramente não é de índice. Já por causa das *features* gerais, algumas páginas de índice que não são de imóveis também são classificadas como relevante.

Figura 4.1 Página de índice de imóveis falso-positva.

Fonte: http://www.mgimobiliaria.com.br/quem-somos.php.

4.1.2 Detector

Para avaliar o Detector rotulamos manualmente 200 sites da *Web*, sendo 100 sites que contêm entidades de imóvel e 100 sites que não contêm entidades de imóvel, diferentes dos sites usados

para treinar o classificador.

Como métricas para a avaliação utilizamos a precisão, cobertura e F1 da detecção. Para avaliar o custo, calculamos quantas páginas precisam ser visitadas por site para o detector decidir se o site é ou não de imóveis. Para uma melhor avaliação, separamos a quantidade de páginas visitadas em 2 subconjuntos: média de páginas visitadas por site relevante e por site irrelevante. É importante lembrar que apesar de usarmos a mesma quantidade de sites relevantes e irrelevantes, na *Web*, a quantidade de sites irrelevantes é muito maior que a de sites relevantes. Alguns sites de imóvel contêm *links* para páginas de entidade em sua *home page*, para melhor avaliar como a busca funciona, a *home page* não é considerada uma página de índice. Só são considerados os sites identificados corretamente pelo detector para o cálculo das médias.

Foram escolhidos sites variados, entre eles: (1) sites que contêm apenas entidades de imóvel, como o Caracciolo Imóveis[2]; (2) sites que contêm entidades de imóvel e também contêm outros tipos de entidades, como o VivaLocal[3]; (3) sites que não contêm entidades de imóvel, mas contêm entidades de outro domínio, como Americanas[4]; (4) sites que não contêm entidades de forma alguma, como ONU[5].

Inicialmente, como *baseline*, foi implementado um *crawler* que não utiliza priorização nos *links*. O *baseline* apenas visita os *links* utilizando uma busca em largura. Ainda assim, é necessário definir um limite máximo de páginas que podem ser visitadas no site, como mostrado na Tabela 4.2, os limites 20, 13, 10, 7 e 5 foram escolhidos para comparação.

Como esperado, a medida que diminuímos o limite de páginas a precisão aumenta e a cobertura diminui. A precisão aumenta, pois, como o classificador não possui uma precisão 100%, quanto mais páginas forem classificadas, maior a chance de uma delas ser falso-positiva. A cobertura diminui, uma vez que se menos páginas são visitadas, maior a chance de não se encontrar uma página de índice.

Na Seção 3.4 discutimos as heurísticas utilizadas na Busca. Para a heurística (1) temos que definir qual valor utilizar no limite máximo de páginas a ser visitado para identificar o site como não sendo de imóveis. Na heurística (2) temos que decidir qual valor utilizar para decidir que as páginas filhas de uma página do site são improdutivas: são dois valores, o valor para a raiz e o valor para as demais páginas. Já na heurística (3) foi dito que *links* com prioridade negativa não são visitados e *links* com prioridade 0 só são visitados caso sejam apontados pela *home page*. Por fim, a heurística (4) define um valor de profundidade máxima para se visitar o site.

Vários valores para as heurísticas foram testados. Para a heurística (1) testamos os valores 20, 13, 10, 7 e 5. Para a heurística (2) testamos o limite 6 para raiz e 3 para as demais; 3 para raiz e 3 para as demais; sem limite para raiz e 3 para as demais; e cada uma das variações para raiz e 2 para as demais. Na heurística (3) testamos seguir *links* com prioridade 0 independente da Busca estar ou não na raiz. E na heurística (4) testamos a profundidade máxima sendo 1, semelhante à [Bar16] e, também, a profundidade máxima sendo 2.

[2]http://www.caraccioloimoveis.com.br/

[3]http://www.vivalocal.com/

[4]http://www.americanas.com.br/

[5]http://www.onu.org.br/

Na Tabela 4.3 mostramos a variação do limite máximo de páginas para a melhor combinação dos valores das outras heurísticas. Os valores que tiveram a melhor combinação, e são usados na Seção 4.2, foram: na heurística (2), os limites 6 para raiz e 3 para as demais; na heurística (3) só seguir *links* com prioridade 0 se os mesmos forem apontados pela raiz; e na heurística (4) a profundidade máxima sendo 2. Para a heurística (1) foi utilizado o valor 10.

Tabela 4.2 Métricas do detector *baseline*.

				Média de páginas visitadas por site	
Limite de páginas	Precisão	Cobertura	F1	Relevante	Irrelevante
20	0.91	0.93	0.92	3.84	16.38
13	0.92	0.90	0.91	3.40	11.29
10	0.92	0.85	0.89	2.92	8.86
7	0.93	0.82	0.87	2.71	6.36
5	0.93	0.78	0.85	2.50	4.62

Tabela 4.3 Métricas do detector com heurística.

				Média de páginas visitadas por site	
Limite de páginas	Precisão	Cobertura	F1	Relevante	Irrelevante
20	0.92	0.90	0.91	2.33	8.02
13	0.92	0.90	0.91	2.33	7.33
10	0.92	0.90	0.91	2.33	7.28
7	0.93	0.89	0.90	2.26	6.35
5	0.93	0.89	0.91	2.26	4.61

Fazendo uma comparação entre a Tabela 4.2 e Tabela 4.3, vemos que a precisão da heurística nunca é inferior à precisão do *baseline*. E que, apesar da quantidade de páginas visitadas ser semelhante, a cobertura do Detector utilizando as heurísticas é superior à cobertura do Detector sem as heurísticas. Portanto, conseguimos ver que as heurísticas ajudam na detecção.

4.2 Avaliação do Detector em um Focused Crawler

Nesta seção, primeiramente, mostramos como funciona a localização dos sites relevantes na *Web* utilizando um *focused crawler* proposto por [BBS11], assim como quantos sites foram coletados e qual foi a precisão do detector no mundo real. Na segunda parte, mostramos algumas análises sobre os sites encontrados.

4.2.1 Localização dos Sites

Para a localização dos sites utilizamos o algoritmo proposto por [BBS11]. O algoritmo baseia-se na ideia que sites de conteúdo semelhantes são apontados pelas mesmas páginas da *Web*. Portanto, se os *backlinks* de um site de imóvel forem extraídos, é provável que as páginas que esses *backlinks* apontam sejam de outros sites de imóvel. O algoritmo funciona como mostrado no Algoritmo 3. A entrada consiste de um conjunto de sites relevantes, as sementes, e do detector que identifica se um site da *Web* é ou não relevante, como o descrito no Capítulo 3. O algoritmo começa inserindo as sementes na fronteira (linha 2). Em seguida, enquanto houver *links* na fronteira (linha 3), um *link* dela é extraído (linha 4) e enviado ao detector, caso o detector determine que o *link* é de um site relevante, os *backlinks* do mesmo são coletados (linha 8) e inseridos na fronteira. Se o *link* for o *backlink* de algum site, então seus *outlinks* são extraídos e inseridos na fronteira (linhas 10 à 12). Apenas *links* com domínio `.br` são inseridos na fronteira.

Algoritmo 3: Localizador

```
   entrada: sementes, detector
   saída  : sites relevantes
 1 sites ← {}
 2 fronteira ← { sementes }
 3 while fronteira.naoVazia() do
 4     link ← fronteira.proximo()
 5     page ← download(link)
 6     if detector.éRelevante(page) then
 7         sites.insere(link)
 8         backlinks ← coletaBacklinks(page)
 9         fronteira.insere(backlinks)
10     if link.éBacklink() then
11         outlinks ← extraiOutlinks(page)
12         fronteira.insere(outlinks)
13 return sites
```

A coleta dos *backlinks* pode ser feita utilizando *APIs* disponíveis na *Web*, para estre trabalho, utilizamos a *API* Moz[6]. Quando é necessário coletar os *backlinks* de um *link* na linha 8, uma requisição é enviada ao Moz e 15 *backlinks* são retornados.

A entrada do algoritmo foi composta por 12 sementes, que podem ser vistas na Tabela 4.4, e do detector descrito no Capítulo 3. Foram visitados 67181 sites da *Web* a partir dessas sementes e 6975 foram identificados como relevante pelo detector, apenas 10.4% dos sites visitados, o que mostra como existem muito mais sites irrelevantes do que relevantes na *Web*.

Dos 6975 sites identificados como de imóveis, é provável que existam falso-positivos. Espera-se que os primeiros sites identificados como relevante pelo detector sejam de fato relevantes, porém, à medida que mais sites são visitados, é esperado que os dados comecem a ficar

[6]http://moz.com

Tabela 4.4 Sementes utilizadas no localizador.

http://www.caraccioloimoveis.com.br/	http://www.rejanemenoncorretora.com.br/
http://www.rizzoimobiliaria.com.br/	http://www.mercadolivre.com.br/imoveis
http://www.gerdalimoveis.com.br/	http://www.mouradubeux.com.br/
http://www.olx.com.br/imoveis	http://www.zapimoveis.com.br/
http://www.wimoveis.com.br/	http://www.imovelweb.com.br/
http://www.znimovel.com.br/	http://www.vivareal.com.br/

muito esparsos e a precisão comece a cair. Por isso, a fim de calcular a precisão do detector na prática, pegamos uma amostra de 300 sites dos 6975 encontrados, distribuídos da seguinte forma: os 100 primeiros encontrados, os 100 últimos encontrados e 100 aleatoriamente escolhidos dentre os 6775 restantes, e verificamos cada um manualmente. A precisão obtida nos primeiros 100 sites encontrados foi 95%, nos 100 sites aleatórios foi 82%, já nos últimos 100 sites foi 67%.

Não há como calcular a cobertura dos sites precisamente, uma vez que existem bilhões de sites na *Web*. Contudo, podemos fazer uma avaliação considerando apenas os sites visitados. Muitos dos sites identificados como relevante possuem a palavra `imoveis` em seu *host*. Assumindo que todo site que possui `imoveis` em seu *host* é um site relevante, dos 60206 identificados como irrelevante apenas 627 sites possuem `imoveis` em seu *host* e pode-se dizer que são falso-negativos, obtendo uma cobertura de 92%. Porém, alguns desses 627 sites só podem ter suas páginas de índice acessadas se formulários de busca forem preenchidos ou códigos em *JavaScript* executados, o que está além do escopo deste trabalho. Portanto, com essas restrições, espera-se que menos que 627 desses sites tenham sido de fato identificados como falso-negativos.

4.2.2 Análise dos Sites

Nesta subseção é mostrado quantos dos sites encontrados possuem entidades para cada estado brasileiro (mais o Distrito Federal).

Dentre os 6975 sites encontrados, apenas aqueles que possuíam `imoveis` em seus *hosts* foram utilizados (mesmo existindo vários sites que não possuem `imoveis` e ainda assim são relevantes), para termos uma maior certeza de que apenas sites de imóvel fossem considerados. Após o filtro, restaram 2296 sites para fazer o mapeamento geográfico por estado brasileiro.

Para cada um dos 2296 sites, deseja-se saber quais estados ele faz parte. Para isso, as seguintes heurísticas foram aplicadas à página raiz do site e à página de índice encontrada pelo detector.

- É verificado se existe alguma palavra capitalizada seguida da sigla de um estado. Caso sim, aquele estado é considerado para o site.
- Os telefones da página são extraídos e os estados dos DDDs encontrados são considerados.

- As âncoras da página são extraídas e caso sejam igual à sigla de um estado ou ao seu nome, o estado é considerado.
- Se a página for a raiz do site, as *tags HTML select* são extraídas.
  - Caso mais da metade das opções sejam cidades, os estados daquelas cidades são considerados. (Apenas cidades que pertencem a um único estado são consideradas)
  - Caso mais da metade das opções sejam estados, cada estado é considerado.

Os nomes das cidades e estados brasileiros foram extraídos do site do IBGE[7].

A avaliação dessa heurística foi feita utilizando 50 sites de imóvel. Para cada estado pertencente ao site, foi criado, manualmente, um item para ser avaliado, informando que aquele site possui aquele estado. Sobre esses itens são calculadas as métricas de precisão e cobertura. Foram criados 107 itens no total. A precisão obtida foi 100% e a cobertura 71%, ou seja, um F1 de 83%.

A distribuição geográfica dos sites encontrados encontra-se na Figura 4.3. Essa análise pode ser vista como um censo de sites de imóveis no Brasil. Podemos ver que a grande maioria dos sites encontrados possui imóveis de São Paulo, o que é esperado dado que é o estado mais populoso do Brasil[8]. Outra característica é que a menor quantidade de sites em um estado foi apenas 8, o que mostra que existe uma grande diversidade de sites de imóvel no Brasil.

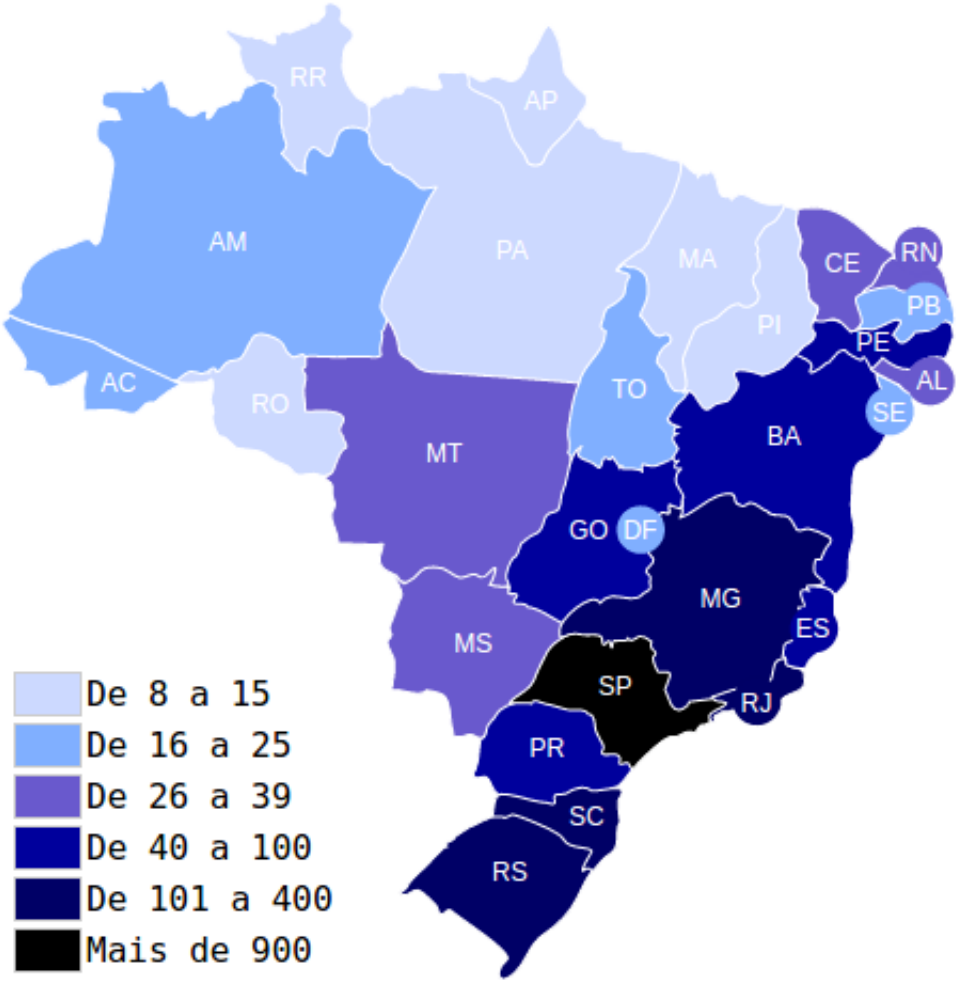
Fizemos uma análise das palavras que mais aparecem nos títulos dos sites encontrados, mostrado no *word cloud* da Figura 4.2. Como podemos ver, apesar do nome do estado de Pernambuco aparecer bastante no *word cloud* da Figura 3.7, já não aparece mais. Vemos que palavras como `imoveis`, `venda` e `imobiliarias` são as que mais aparecem. Indicando que o Detector criado não é enviesado para um estado específico.

Figura 4.2 *Word cloud* dos títulos dos sites encontrados.

Fonte: Elaborada pelo autor.

[7] http://cidades.ibge.gov.br/xtras/home.php
[8] https://pt.wikipedia.org/wiki/Lista_de_unidades_federativas_do_Brasil_por_população

Figura 4.3 Distribuição geográfica da quantidade de sites de imóvel por estado.



Fonte: Elaborada pelo autor.

CAPÍTULO 5

Conclusão

A motivação para este trabalho envolve criar um banco de dados com informações de entidades, em especial, de imóveis. O primeiro passo é encontrar inúmeros sites da *Web* que contenham tais entidades. Para que as informações de cada entidade possam ser extraídas.

Neste trabalho, foi proposto um detector para sites de imóvel em português. Dado qualquer site da *Web*, o detector é capaz de decidir, *efetivamente* e *eficientemente*, se ele contém, ou não, entidades de imóveis. Com o auxílio de um *focused crawler* que explora o grafo da *Web*, utilizamos nosso detector e foram encontrados mais de 5000 sites de imóvel a partir de apenas 12 sites iniciais.

Muitos sites de entidades necessitam que formulários *HTML* sejam preenchidos para que as páginas de índice sejam encontradas. Outros necessitam que códigos em *JavaScript* sejam executados. Ambos os casos estão fora do escopo deste trabalho. Uma proposta de trabalho futuro é resolver algum desses dois problemas.

Uma outra proposta de trabalho futuro é a criação de um Priorizador de *Links* que utilize aprendizado supervisionado, ao invés dos filtros utilizados. Dessa forma, é possível saber a quantos passos de uma página de índice um *link* está, fazendo com que a Busca priorize os *links* de forma mais precisa. Outra melhora que essa abordagem traz é que o classificador pode aprender os padrões utilizados com o passar do tempo, melhorando sua cobertura à medida que mais páginas de índice são encontradas.

Por último, vimos na Subseção 4.1.1 que, por utilizarmos um único classificador com as *features* gerais e específicas, algumas páginas que possuem apenas um desses dois conjuntos de *features* são falso-positivas. Uma ideia proposta por [BF07] mostra que separar um classificador complexo em vários classificadores mais simples pode ser mais efetivo. No nosso caso, poderíamos criar dois classificadores: (1) um classificador de página de índice; e (2) um classificador para saber se a página é de imóveis. Dessa forma, as funções a serem aprendidas por cada classificador ficam mais simples e é esperado que a efetividade do detector melhore.

Referências Bibliográficas

[Bar16] L. Barbosa. Harvesting forum pages from seed sites. 2016.

[BBS11] L. Barbosa, S. Bangalore, and V. K. R. Sridhar. Crawling back and forth: Using back and out links to locate bilingual sites. In *IJCNLP*, pages 429–437, 2011.

[BF07] Luciano Barbosa and Juliana Freire. Combining classifiers to identify online databases. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 431–440, New York, NY, USA, 2007. ACM.

[BSYB12] L. Barbosa, V.K.R. Sridhar, M. Yarmohammadi, and S. Bangalore. Harvesting parallel text in multiple languages with limited supervision. In M. Kay and C. Boitet, editors, *COLING*, pages 201–214. Indian Institute of Technology, Bombay, 2012.

[BYCMR05] Ricardo Baeza-Yates, Carlos Castillo, Mauricio Marin, and Andrea Rodriguez. Crawling a country: Better strategies than breadth-first for web page ordering. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, WWW '05, pages 864–872, New York, NY, USA, 2005. ACM.

[CGMP98] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through url ordering. *Comput. Netw. ISDN Syst.*, 30(1-7):161–172, April 1998.

[CHM11] M. J. Cafarella, A. Halevy, and J. Madhavan. Structured data on the web. *Communications of the ACM*, 54(2):72–79, 2011.

[CvdBD99] S. Chakrabarti, M. van den Berg, and B Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16):1623–1640, 1999.

[JSYL13] J. Jiang, X. Song, N. Yu, and C. Lin. Focus: Learning to crawl web forums. *IEEE Trans. Knowl. Data Eng.*, 25(6):1293–1306, 2013.

[Kot07] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[PTS09] Anshika Pal, Deepak Singh Tomar, and S. C. Shrivastava. Effective focused crawling based on content and link structure analysis. *CoRR*, abs/0906.5034, 2009.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).