Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Classificação de páginas Web no domínio de Setor Imobiliário

Ana Caroline Ferreira de França

Trabalho de Graduação

Recife
06 de Dezembro de 2017

Universidade Federal de Pernambuco
Centro de Informática

Ana Caroline Ferreira de França

Classificação de páginas Web no domínio de Setor Imobiliário

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
06 de Dezembro de 2017

Agradecimentos

Em primeiro lugar gostaria de agradecer à minha família, especialmente minha mãe Jovelina, que me apoiou nos momentos mais difíceis da graduação e sempre me incentivou a nunca desistir dos meus objetivos. Agradecimento especial também ao meu irmão André Luiz, que acompanhou meus momentos de estresse e sempre procurou me manter animada com seu jeito singular de ser.

Ao professor Luciano de A. Barbosa que me orientou no decorrer do semestre e sempre se preocupou em me apresentar novas tecnologias e métodos que não conhecia. Obrigada pela orientação, pelo conhecimento transmitido neste curto período de tempo e por sempre estar disposto a ajudar.

Aos amigos que tenho desde do ensino médio Natália, Guilherme e Mariana que compartilham da mesma rotina desde 2010.

Aos amigos feitos durante a graduação, especialmente: Edipo, Rodrigo, Chen, Lucas e Dic. Com vocês as viradas de noite no CIn para finalizar projetos se tornaram mais especiais.

Ao CESAR que me acolheu desde do começo da graduação e me proporcionou (e ainda proporciona) experiências inesquecíveis e super construtivas para minha carreira profissional.

Por fim, e não menos importante, a todos os funcionários do Centro de Informática que a cada dia contribuem para que este seja um lugar de excelência.

It always seems impossible until it's done.
—NELSON MANDELA

Resumo

A quantidade de informações presentes na *World Wide Web* tem aumentado rapidamente nos últimos anos. Neste conjunto de dados encontramos as mais diversas categorias (entretenimento, moda, ciência, tecnologia, entre outros) e formatos. Devido ao enorme volume e variedade as tarefas de indexação e categorização se tornam inviáveis de serem realizada manualmente. A automatização da categorização das páginas web torna essas tarefas factíveis e eficaz. O objetivo deste trabalho é realizar a implementação de uma solução eficiente para classificar páginas web no setor imobiliário. Mais especificamente, dada uma página de anúncio de imóveis, pretende-se classificá-la como venda ou aluguel, e de acordo com seu tipo (casa, apartamento, outros).

Palavras-chave: classificação de texto, processamento de linguagem natural, aprendizado supervisionado, web

Abstract

The amount of information on the World Wide Web has increased rapidly in recent years. In this set of data we find the most diverse categories (entertainment, fashion, science, technology, among others) and formats. Due to the sheer volume and variety of indexing and categorization tasks they become unfeasible to be performed manually. Automating categorization of web pages makes these tasks feasible and effective. The objective of this work is to implement an efficient solution to classify web pages in real estate industry. More specifically, given a property listing page, it is intended to classify it as sale or rent, and according to its type (house, apartment, other).

Keywords: text classification, natural language processing, supervised learning, web

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Com a popularização da *World Wide Web* e a facilidade de disponibilizar documentos passamos a lidar com uma "explosão" de dados. A organização deste grande volume de dados é fundamental para garantir facilidade de acesso à informação e ao conhecimento disponível, porém a organização destes dados distribuídos em crescimento continuo tem se tornado um desafio. Uma solução é a realização de categorização manual, porém esta é impraticável, com alto custo e requer muito tempo devido ao seu volume.

Muitos métodos que possam prover organização, filtragem e controle desses dados estão sendo bastante explorados. Um destes métodos é a classificação automática de texto, o qual se baseia nas características do texto presentes em um determinado documento para associa-lo a uma ou mais classes. No contexto de páginas *web* além do texto temos *hiperlinks*, *tags* HTML e metadados que podem conter fortes indicativos da classe a qual o documento pertence. Muitas pesquisas realizadas mostram que os componentes não textuais das páginas *web* podem ser utilizados para aprimorar a performance do classificador, porém os componentes textuais são considerados fundamentais [28].

O processo de automatização de categorização de texto é caracterizado por algumas propriedades que são encaradas como desafio, sendo as principais propriedades: a alta dimensionalidade, vetores esparsos, complexidade linguística e a redundância de dimensões. Para o algoritmo de classificação lidar com essas propriedades, os documento são submetidos a um conjunto de técnicas de pré-processamento e seleção de características.

O objetivo deste trabalho é desenvolver um classificador de páginas *web*, em português, voltado para o setor de imóveis. A classificação se divide em duas tarefas: uma para identificar se o imóvel está anunciado para aluguel ou venda, denominada Tarefa 1, e o outra para identificar o tipo do imóvel (casa, apartamento e outros tipos de imóveis), denominada Tarefa 2. Na solução apresentada, a página *web* é submetida a técnicas de pré-processamento tradicionais e técnicas com foco na classificação de páginas *web*, como por exemplo, utilização de conteúdo de *tags* especificas como uma *feature*, para garantir um bom desempenho.

CAPÍTULO 2

Fundamentos

2.1 Aprendizado Supervisionado

A área de Aprendizagem de Máquina é responsável pelo estudo e construção de algoritmos capazes de fazer previsões e aprender a partir de seus próprios erros. Os modelos desenvolvidos a partir desses algoritmos são definidos através de *inputs* amostrais com a finalidade de fazer previsões ou decisões orientadas pelos dados e não somente por instruções programadas. Aprendizagem de máquina é subdividida em algumas categorias de acordo com a base de dados utilizada para realizar o treinamento; uma delas é o aprendizado supervisionado.

No aprendizado supervisionado o algoritmo de aprendizado recebe um conjunto de exemplos de treinamento rotulado, ou seja, cada exemplo está associado a um valor de saída [13]. Podemos dividir os problemas de aprendizado supervisionado em regressão ou classificação. No primeiro lidamos com rótulos de valores contínuos, onde o algoritmo produz um resultado numérico. Já no problema de classificação lidamos com rótulos discretos, onde o algoritmo determina a que grupo um exemplo pertence [13].

Neste trabalho iremos focar no problema de classificação, onde cada exemplo é representado por um vetor de característica e um rótulo indicando a qual classe pertence. A meta do algoritmo é desenvolver um modelo capaz de determinar a classe de exemplos que não estão rotulados.

Uma característica importante que deve ser observada nos dados que são utilizados como entrada para os modelos de classificação é a capacidade do modelo, a partir do conjunto de treinamento, generalizar exemplos nunca vistos antes. A generalização é uma propriedade fundamental, pois os dados utilizados para treinamento são apenas uma amostra (incompletos e podem apresentar ruídos). Em Aprendizado de Máquina existem conceitos que definem quão bem um modelo aprendeu e generalizou novos dados, são elas: *underfitting* e *overfitting*.

Overfitting e *underfitting* são as duas maiores causas de baixa performances dos algoritmos de aprendizado de máquina [14]. *Overfitting* ocorre quando o algoritmo aprende os "detalhes" e ruídos do conjunto de treinamento apresentando baixo desempenho no conjunto de testes e alto desempenho no conjunto de treinamento. Quando o modelo não consegue aprender os dados de treinamento e nem generalizar novos dados nos referimos a ocorrência de *underfitting* dos dados.

Neste capítulo serão apresentados os tipos de classificação presentes no Aprendizado Supervisionado (Seção 2.1) e as métricas mais utilizadas para realizar a avaliação de desempenho de um modelo/algoritmo para determinado conjunto de dados (Seção 2.2).

2.1.1 Tipos de classificação

Nesta seção será introduzido os conceitos básicos sobre os tipos de classificação existentes no Aprendizado Supervisionado e como alguns desses problemas podem ser simplificados através da decomposição em subproblemas.

Os problemas de classificação podem ser divididos em subcategorias de acordo com a quantidade de classes que esta relacionada a um exemplo. As categorias são:

- **Binária**: no problema de classificação binária o algoritmo de classificação lida somente com duas classes (0 ou 1), que varia seu significado de acordo com o conjunto de dados que se deseja classificar. Um exemplo clássico é a detecção de SPAM, onde o algoritmo deve definir, através da análise da mensagem presente no email, se determinado email é ou não SPAM. Já no contexto de *web* a classificação binária pode ser utilizada em um *crawler* para filtrar páginas de interesse do mesmo, como por exemplo, identificar páginas de venda/aluguel de imóveis e ignorar as demais. Apesar de ser um problema simples é considerado um dos mais importantes devido a capacidade de problemas mais complexos serem reduzidos a ele.

- **Multiclasse**: neste problema, o algoritmo de classificação recebe um conjunto de dados que possui mais de duas classes. Na *web*, por exemplo, um site de nóticias pode conter diversos tipos de informações: esporte, culinária, moda, saúde, dentre outros. Um classificador de páginas *web* construído para identificar esses "tipos" de informações deve lidar com mais de duas classes, associando a cada página apenas um dos "tipos".Existem algoritmos de classificação que permitem a resolução direta desse problema, porém a maneira mais eficiente é decompor o problema em subproblemas de classificação binária.

- **Multi-label**: Quando um exemplo pertence a mais de uma categoria simultaneamente estamos lidando com um problema multi-label. Esses problemas são frequentes e mais complexos. Os algoritmos que são comumente utilizados para resolução desse problema são: Árvores de Decisão, Redes Neurais e k-NN [4]. Como citado no tópico anterior o método mais comum para resolução dos problemas que envolvem mais de uma classe é a decomposição em um conjunto de sub-classificadores.

2.1.1.1 Métodos de decomposição de problemas multiclasse e multi-label

Existem métodos de classificação que são utilizados para reduzir problemas multiclasse e multi-label a problemas binários. Geralmente são utilizados quando o algoritmo que será utilizado para classificar determinado conjunto de dados não suporta diretamente o problema de classificação com mais de duas classes.

As estratégias decomposicionais mais comuns são:

- Um contra todos (*one-against-all*): neste método é criado um classificador binário por classe, onde os exemplos da classe são considerados da classe positiva e os demais da classe negativa [15]. Esse conjunto de classificadores recebe um novo exemplo e reage de forma distinta, dependendo do tipo de problema, para definir a(s) classe(s) associada(s) ao exemplo apresentado. Caso seja problema multiclasse é atribuída uma única classe ao

exemplo que pertence ao classificador que produziu a maior nota. Em problemas multi-label é atribuído ao exemplo de entrada todas as classes que os classificadores foram capazes de reconhecer.

- Um contra um (*one-against-one*): neste método de decomposição cada classificador binário é responsável pela classificação de uma dupla de categorias. Para realizar a classificação o exemplo desconhecido (sem rótulo) é apresentado ao conjunto de classificadores. Caso seja um problema multiclasse o rótulo atribuído ao documento é o que obteve a maior soma dos votos dos classificadores. Já em problemas multi-label o exemplo recebe os rótulos que tiverem um número mínimo de votos; outra possibilidade é atribuir rótulos definindo um *threshold* a partir do mais votado. Para esse segundo problema não existem muitos estudos que tratam especificamente a distribuição de rótulos neste tipo de decomposição. Embora o número de classificadores gerados seja de ordem k^2, onde k é o número de classes, o treinamento de cada um envolve apenas duas classes. Mesmo que o número de classes seja alto, o tempo total despendido na geração dos classificadores geralmente não é grande [16].

2.1.2 Medidas de desempenho

Para avaliar o desempenho dos classificadores não podemos considerar apenas a porcentagem de acerto (acurácia). Ao utilizar somente essa contagem como métrica em uma base desbalanceada, a qual lidamos na maioria dos casos, podemos ter um classificador com alta taxa de acerto porém inútil na prática, como por exemplo: se um classificador responde sempre 1 (classe positiva) e a maioria das instâncias de teste pertencem a essa classe teremos um classificador com alta taxa de acerto, porém inutilizável em uma aplicação real. Para evitar a falsa impressão de bom desempenho outras medidas são utilizadas, como por exemplo, precisão, *recall* e *F-Measure*.

Utilizamos a Matriz de Confusão para calcular as medidas de desempenho citadas. Nesta matriz a diagonal principal representa os acertos, enquanto as demais representam os erros na classificação. Para simplificar, na Tabela 1 temos o exemplo de uma matriz de confusão para um problema de classificação binária de uma classe c_i.

	Predita c_i	Predita $\neg c_i$
Verdadeira c_i	VP	FP
Verdadeira $\neg c_i$	FN	VN

Tabela 2.1 Matriz de confusão de um classificador binário

Cada documento do conjunto de teste pode ser classificado em uma dessas quatro categorias presentes na Tabela 1:

- Verdadeiro Positivo (VP): exemplo classificado corretamente em relação a uma categoria.
- Falso Positivo (FP): exemplo está relacionado a categoria de forma incorreta.

- Falso Negativo (FN): exemplo não está relacionado a uma determinada categoria, mas deveria estar.
- Verdadeiro Positivo (VP): exemplo não deveria estar relacionado a determinada categoria e de fato não está.

Abaixo temos as definições das métricas e como calculá-las através da matriz de confusão baseadas em [6].

- Precisão: probabilidade de um exemplo ser classificado na categoria correta em relação ao exemplos que foram classificados nesta categoria (corretos ou não). Para isso, a medida é calculada através da divisão do número de acertos (VP) pelo número de exemplos classificados como positivos (VP + FP), ou seja:

$$P = \frac{VP}{VP + FP}$$

- *Recall* ou cobertura: conhecido também por revocação, essa medida avalia a proporção de exemplos corretos em relação ao total de documentos que deveriam ser classificados em determinada categoria.

$$R = \frac{VP}{VP + FN}$$

- *F-measure*: medida única para realizar uma comparação mais direta dois ou mais classificadores. Essa medida é baseada na média harmônica ponderada da precisão e o recall.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

Quando lidamos com mais de duas classes precisamos agregar as métricas para realizar uma avaliação do desempenho do classificador no geral. Existem duas formas de calcular:

- Média macro: calculada para cada classe localmente e em seguida é feita uma média geral.
- Média micro: através de uma matriz de confusão é feita a soma de verdadeiros/falsos positivos para cada classe e o valor das medidas é feito a partir desse somatório.

Essas medidas podem variar dependendo da quantidade de categorias. Na média micro os valores são dominados pelas categorias que possuem mais exemplos e a média macro faz uma avaliação individual atribuindo o mesmo peso para as categorias.

2.2 Classificação de texto

Limitações da classificação manual ou a utilização de regras definidas por especialistas do domínio para realização da classificação de texto, proporcionaram interesse no desenvolvimento e utilização de técnicas de aprendizado de máquina tendo em vista a classificação automática de texto.

A utilização de técnicas de aprendizado supervisionado possibilita o aprendizado automático a partir de um conjunto de dados rotulados. Essa técnica não anula a intervenção humana, pois é necessária para rotulação de dados de treinamento/teste. Porém os modelos de classificação desenvolvidos são de fácil manutenção e atualização quando comparado com a classificação manual ou geração de regras por especialistas [17].

Neste capítulo serão abordados conceitos referentes às formas de representação de documentos, especificidades da classificação de texto, técnicas de pré-processamento e seleção de *features*, algoritmos de classificação e por fim particularidades da classificação de páginas *web*.

2.2.1 Representação dos documentos

Antes de realizar a classificação do documento é preciso representá-lo de outra forma, pois os dados brutos não são entradas apropriadas para os algoritmos de classificação. Existem muitas maneiras de representar um texto que agregam mais informações semânticas, mas não acrescentam valor ao modelo estatístico baseados nelas. A representação mais simples e mais utilizada é a *bag-of-words*.

Bag-of-Words (BoW) é um modelo onde o documento passa a ser representado como uma coleção de palavras, desconsiderando a ordem das palavras e até mesmo a estrutura gramatical, porém sempre mantendo a multiplicidade das palavras [18]. Para facilitar o entendimento das diferentes sub-formas desse modelo são necessárias algumas definições:

- Termo: sequência de caracteres de um determinado alfabeto.
- Vocabulário: todas as palavras que podem ser utilizadas para classificação. Essas palavras são extraídas do conjunto de documentos utilizados como base de dados. Pode ser previamente estabelecido ou definido a partir dos documentos utilizados para treinamento.
- Representação vetorial do documento: vetor com dimensão do tamanho do vocabulário, onde cada dimensão se refere a uma palavra. Caso o documento não possua determinada palavra é atribuído àquela dimensão do vetor o valor 0.

A partir desses conceitos podemos definir sub-formas que são utilizadas em conjunto com *bag-of-words* para se ter a definição de um documento.

- Binária: nesta representação a presença ou ausência de determinada palavra é utilizada para formar o vetor, ou seja, temos um vetor onde cada dimensão só pode apresentar os valores 0 (ausência) ou 1 (presença).

- Frequência dos termos (*Term Frequency* -TF): cada dimensão do vetor contém um inteiro correspondente a quantidade de vezes que a palavra ocorre no documento.
- Log da Frequência do termo (Log TF) - cada dimensão guarda o logaritmo da frequência dos termo no documento.
- Frequência do termo - inversa dos documentos (*Term frequency - inverse document frequency* - TF-IDF): essa é a representação mais utilizada em classificação de texto, pois não considera apenas a frequência do termo no documento, mas também a quantidade de documentos que determinada palavra aparece. Nesta representação quanto mais documentos uma palavra/termo aparecer, menos significativa para classificação ela é, pois não é uma característica discriminante de uma classe. Para um termo i em um documento j o TF-IDF será:

$$w_{i,j} = tf_{i,j} \cdot \left(\frac{N}{df_i} \right)$$

onde: $tf_{i,j}$ é o número de ocorrência de i em j, N número de documentos e df_i é o número de documentos com o termo i.

2.2.2 Desafios da Classificação de Texto

Nesta Seção serão apresentadas as principais propriedades, consideradas desafios, que são comuns a todos os documentos em relação a Classificação de Texto.

Alta dimensionalidade

Independentemente da forma de representação dos documentos problemas de Classificação de Texto apresentam muitos atributos (termos). Devido a essa quantidade de atributos o vetor que representa o documento apresenta uma alta dimensionalidade. Se cada palavra que pertence ao conjunto de treinamento for utilizada como atributo, caso o conjunto de treinamento seja composto por alguns milhares de documentos, a representação do documento pode ser composta por milhares de atributos [9].

É possível reduzir um pouco essa dimensão através da aplicação de técnicas de pré-processamento ou seleção de atributos (ver Seção 3.4), mesmo assim o valor ainda poderá ser elevado. Ao escolher o algoritmo de classificação é preciso se certificar que ele consiga lidar com vetores de alta dimensionalidade para evitar a ocorrência de *overfitting* (ver Capítulo 2) ou estouro da capacidade de memória.

Vetores Esparsos

Apesar de lidar com alta dimensionalidade cada documento contém um pequeno subconjunto de atributos dessas dimensões [9], ou seja, os vetores que representam os documentos são muito esparsos (possuem muitos valores 0). Isso ocorre devido ao número de palavras distintas em um único documento ser medido na ordem de centenas. O algoritmo de classificação pode usar essa propriedade a seu favor para melhorar a velocidade do treinamento e diminuir o uso de memória.

Complexidade linguística

As linguagens naturais possuem muitos sinônimos, ambiguidade e figuras de linguagem. Portanto, podemos selecionar três documentos relacionados a um mesmo tema e eles possuírem poucos termos em comum, exceto por termos que são extremamentes comuns em qualquer tipo de documento, definidos *stopwords* (ver Seção 3.3.2), que não são discriminantes de classes [12]. Outra característica bastante comum é a presença de termos similares em documentos distintos, porém com significados diferentes dependendo do contexto.

Redundância

Além de lidar com a complexidade linguística os algoritmos de classificação deve ser capaz de lida com a redundância da linguagem natural. Devido a esta característica a representação vetorial do documento (ver Seção 3.1) algumas dimensões podem conter distribuições bastante semelhantes que não agregam valor a discriminação das classes; que implica na alta dimensionalidade [19].

2.2.3 Pré-processamento

Lidar com a linguagem natural apresenta algumas dificuldades, dentre ela está a falta de estruturação. A etapa de pré-processamento tem um alto custo computacional e é fundamental para obter bons resultados de classificação.

Páginas *web* são essencialmente compostas por hipertexto. Além dos componentes de multimídia e texto, as páginas *web* possuem hiperlinks, *tags* HTML e metadados que podem ser utilizados como *features* para classificação. Muitas pesquisas realizadas mostram que os componentes textuais destas páginas possuem mais informações úteis para realização da classificação do que componentes não textuais [4], estes podem ser utilizados para aprimorar a performance do classificador .

Nesta seção serão apresentados os processos de pré-processamento mais comuns na atividade de categorização de textos.

2.2.3.1 Tokenização

O processo de tokenização é o primeiro passo para iniciar o pré-processamento e tem como objetivo segmentar um documento textual em unidades menores. Essas unidade menores são denominadas *tokens*, que na maioria das vezes se refere a uma única palavra do texto. Geralmente esses *tokens* são obtidos através da identificação de delimitadores como espaço em branco e quebra de linha.

We must all face the choice between what is right, and what is easy.						
We	must	all	face	the	choice	...

Tabela 2.2 Exemplo de tokenização

2.2.3.2 *Stopwords*

Uma vez realizado o processo de tokenização, o próximo passo é a identificação de *tokens* que podem ser desconsiderados nos passos posteriores. Um documento contém muitos *tokens* que não possuem valor semântico, sendo úteis apenas para compreensão geral do texto.

Esses *tokens*, denominados *stopwords*, que não constituem conhecimento no texto que está sendo analisado são desconsiderados. Geralmente são palavras conectivas ou auxiliares (por exemplo: e, para, de , em, uma, o, a) que não são consideradas atributos discriminantes na atividade de categorização do texto. Normalmente, 40 a 50% do total de palavras de um texto são removidas com uma lista de *stopwords* [11].

2.2.3.3 *Stemming*

Para reduzir o número de dimensões dos documentos que serão classificados é comum a aplicação da técnica de *stemming*.

Esta técnica consiste em reduzir os *tokens* para sua forma básica, removendo prefixos e sufixos obtendo assim apenas os radicais (*stems*) de acordo com as regras gramaticais da linguagem. Se aplicarmos *stemming* a palavras como: computador, computar e computação estas palavras serão reduzidas a "computa", por exemplo.

Os algoritmos de *stemming* são extremamente dependentes do idioma para qual foram escritos. A maioria desses algoritmos por terem sido desenvolvidos para língua inglesa não apresentam bons resultados para outras línguas. A língua portuguesa não possui nenhum algoritmo de *stemming* padrão por ser uma língua morfologicamente mais complexa que o inglês, apresentando modificações que alteram o radical da palavra.

Para o Português temos três algoritmos que se destacam: algoritmo de Porter para português, "Removedor de Sufixos da Língua Portuguesa" (RSLP) e o algoritmo STEMBR [20].

2.2.4 Seleção de Características

Como citado na Seção 3.2.1 umas das características da classificação de texto é a alta dimensão dos vetores que representam os documentos. Muitos algoritmos de aprendizado de máquina não são capazes de lidar com esta característica. Na seleção de características o objetivo é remover as dimensões menos significativas para melhorar o resultado dos algoritmos de classificação e diminuir a complexidade computacional [21].

Algumas técnicas de pré-processamento que são utilizadas ajudam a reduzir a dimensionalidade (ver Seção 3.3.2 e 3.3.3), porém não são capazes de identificar as características que mais discriminam uma classe da outra.

Yang e Pedersen [8] avaliaram e compararam quatro métodos de seleção de características aplicados em problemas de categorização de texto. Suas conclusões foram que os métodos Ganho de Informação (GI) e X^2 Estatístico (CHI) obtiveram os melhores resultados sem alterar a acurácia do classificadores utilizados nos experimentos e os resultados obtidos com a Frequência do Documento (FD) são comparáveis a métodos de maiores custos computacionais (como GI e CHI).

Os métodos mais utilizados para seleção de características são:

- Frequência de Documento (FD): trata-se de uma função que indica a quantidade de documentos que um atributo aparece. Através desta contagem atributos com a contagem menor que determinado *threshold* são removidos. É o método mais simples de redução de vocabulário com uma complexidade computacional aproximadamente linear de acordo com a quantidade de documentos no conjunto de treinamento.

- Ganho de Informação (GI): essa abordagem mede o ganho de informação do atributo para a predição de uma classe, através da presença ou ausência do mesmo em um documento. O ganho de informação é computado através da estimação de probabilidades condicionais da categoria dado um termo (atributo) e a entropia. Pode ser calculado sa seguinte forma:

$$GI(S,A) \equiv Entropia(S) \; - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} Entropia(S_v)$$

  onde o calculo da entropia é definido como:

$$Entropia(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

  , onde S é uma amostra de exemplos de treinamento, $p_{\oplus}$ é a proporção de exemplos positivos em S e $p_{\ominus}$ é a proporção de exemplos negativos em S.

- Informação Mútua (IM): é uma medida de dependência entre variáveis. Está relacionada a quantidade de informação que uma variável aleatória tem acerca da outra. Comumente utilizada em modelagem estatística de associações de palavras e aplicações relacionadas.

- X^2 Estatístico (CHI): medida estatística com complexidade quadrática, similar a MI e GI. Usada para verificar quando a ocorrência de um termo e de uma classe específica são independentes. Possui vantagem em relação a MI por apresentar um valor normalizado de mais fácil comparação entre palavras de mesma categoria [22].Quando se atinge um alto valor no teste de CHI a hipótese nula (H0) de independência deve ser rejeitada, ou seja, a ocorrência do termo e da classe são dependentes indicando que o atributo deve ser selecionado.

2.2.5 Algoritmos de classificação

Existem diversas abordagens para se aproximar de uma solução ideal na área de classificação/categorização de textos. Essas abordagens também são utilizadas para outros tipos de domínios de dados (tanto quantitativos quanto categóricos). Tratar o texto como um dado quantitativo (ver Seção 3.1) torna possível o uso de diversos métodos que lidam com este tipo de dado diretamente. Mesmo com essa alteração de representação quantitativa, o texto continua sendo um tipo particular de dado (ver Seção 3.2). Devido a estas particularidades os algoritmos de classificação encontram algumas dificuldades para manter uma alta eficiência.

Nesta seção serão apresentadas características de alguns algoritmos que podem ser utilizados para classificação de texto baseados em [22].

2.2.5.1 Árvores de decisão

Árvores de decisão são uma decomposição hierárquica dos dados de treinamento. A realização da divisão hierárquica é feita baseada nos atributos; que são as palavras no contexto de texto. Para inserir uma nova ramificação na árvore é utilizado o cálculo de Ganho de Informação (ver Seção 2.2.4) para escolha do atributo que mais distingue os dados. No contexto de dados textuais as predições são feitas através de condições formadas pela presença ou ausência de uma ou mais palavras no texto.

A divisão dos dados na árvore é feita de forma recursiva até que os nós da folha contenham um número mínimo de registros. A partir de uma instância de teste, é aplicada uma sequência de predicados nos nós a fim de percorrer um ramo da árvore até que se alcance um nó folha que contém a classe a qual a instância pertence.

2.2.5.2 *Random Forest*

É um algoritmo baseado no método de árvores de decisão. Em vez de ter como objetivo de construir uma única estrutura a partir de um conjunto de dados o *Random Forest* tem como objetivo criar um conjunto árvores de decisões em paralelo onde cada uma utiliza um subconjunto de atributos selecionados aleatoriamente a partir do conjunto original.

A partir desse conjunto de árvores é verificada qual possui maior ganho de informação para solução de um dado problema. Para realizar essa verificação é selecionado um subconjunto que apresenta mais vantagens para a tomada de decisão. Cada subconjunto possui um voto para qual classe/categoria o atributo deve pertencer, esse voto possui um peso que é afetado pela igualdade entre as árvores. Cada árvore é especializada para classificar seu subconjunto, isso faz com que a generalização individual das árvores seja comprometida. Porém através da combinação delas o *Random Forest* consegue um bom desempenho.

2.2.5.3 *Support Vector Machine*

Support Vector Machine (SVM) é uma técnica utilizada para classificação e reconhecimento de padrões e regressão de dados que tem alta capacidade de generalização e suporte a dados de alta dimensionalidade. O SVM representa os exemplos como pontos no espaço e tem como principal objetivo encontrar uma linha de separação (denominada hiperplano) entre dados de duas classes distintas. Essa linha busca maximizar a distância entre os campos mais próximos em relação a cada uma das classes.

Um SVM pode lidar com classes linearmente não separáveis através das funções de Kernel [23]. Essas funções realizam uma transformação nos exemplos de entrada (exemplos de treinamento e teste) para um novo espaço. Desta forma problemas mais complexos podem ser solucionados a partir de um SVM linear.

2.2.5.4 *Naive Bayes*

O classificador *Naive Bayes* é um classificador probabilístico baseado no Teorema de Bayes [45]. Este classificador estima a probabilidade de um determinado documento pertencer a uma categoria. A principal característica deste algoritmo é não levar em consideração a correlação

entre os atributos, estes são tratados de forma independente dado a classe.

É considerado um algoritmo rápido que apresenta bom desempenho na previsão de categorias, mesmo com poucos dados de treinamento. Para lidar com classificação binária e multiclasse existem variações do *Naive Bayes* como: multinomial e Bernoulli. No modelo de Bernoulli é considerada apenas a presença ou a ausência dos termos no documento. Já no multinomial é considerada a frequência das palavras e os documentos de cada categoria são modelados como amostras extraídas de uma distribuição multinomial dos termos [20].

2.2.5.5 XGBoost

O algoritmo *Extreme Gradient Boosting* (XGBoost) implementa o *Gradient Boosting* baseado em árvores de decisão [24]. Podemos comparar XGBoost com o *Random Forest*, pois ele é um ensemble de árvores de decisão. As árvores individuais são modelos "fracos", mas em conjunto podem apresentar boas performances.

Com o *Random Forest* são desenvolvidas árvores de decisão em paralelo a partir de subconjuntos dos dados como citado na Seção 3.5.2. Já o XGBoost, cria um conjuntos de árvores de decisões pequenas e simples de forma iterativa. Inicialmente é criada uma árvore simples de baixo desempenho, em seguida constrói outra árvore com a capacidade de prever o que a anterior não conseguiu. Em cada iteração é criada uma árvore que corrige a anterior até que se atinja uma condição de parada, como por exemplo, o número de árvores a serem construídas.

Este algoritmo é bastante utilizado em aprendizado de máquina e ficou famoso a partir do Kaggle [25], um site de competição de aprendizagem de máquina. Além da alta performance, o XGBoost também é reconhecido pela sua flexibilidade e velocidade. Apesar do *Gradient Boosting* construir as árvores uma por uma de forma sequencial, o XGBoost paraleliza o treinamento de cada árvore, facilitando a etapa de treinamento.

2.2.5.6 *Ensemble*

O desempenho dos classificadores podem variar de acordo com o problema, quantidade de exemplos, atributos selecionados, entre outras características. Em geral, os algoritmos utilizados para construir um classificador não são capazes de apresentar um desempenho superior aos outros em todas as situações, pois o desempenho está relacionado a fatores que contextualizam o problema [27]. A partir disto foram feitos alguns estudos que mostram a possibilidade de utilização de vários classificadores para lidar com um único problema combinando vantagens dos algoritmos, denominado ensemble de classificadores.

Ensemble de classificadores é um paradigma de aprendizado de máquina em que um conjunto de classificadores são selecionados para resolver um mesmo problema. A capacidade de generalização destes classificadores, em geral, é maior do que os classificadores que o compõe (denominados classificadores-base). Dietterich [27] demonstra que para que um *ensemble* tenha uma acurácia maior que de seus classificadores-base eles devem possuir erros individuais não correlacionados (ou seja, cometem erros em exemplos distintos) e tenham uma acurácia superior a 50%.

2.2.5.7 AutoML (*Automated Machine Learning*)

A utilização de vários classificadores em conjunto para resolução de um problema requer um conjunto de testes muito grande para se aproximar da solução ótima, pois existem muitas combinações possíveis. Para essas aplicações serem eficientes na prática é preciso realizar a escolha automática de um algoritmo que apresente um ótimo desempenho para o problema, isso inclui encontrar os melhores parâmetros e etapas de pré-processamento.

AutoML é a área de pesquisa que visa a automação progressiva da aprendizagem automática de máquinas. O objetivo do AutoML é tornar o aprendizado da máquina mais acessível através da geração automática de um pipeline de análise de dados.

2.2.6 Classificação de páginas web

Nesta seção serão abordadas algumas características particulares da classificação de página web.

2.2.6.1 Estrutura de uma página web

Podemos dividir uma página web em dois elementos basicamente: cabeçalho e corpo. O cabeçalho contém informações como título da página e elementos opcionais. Já o corpo é composto pelo texto da página, listas, parágrafos e outros elementos.

No cabeçalho, o título é de suma importância para classificação da página, pois as palavras presentes no título de um documento costumam ser mais representativas do que uma palavra presente em seu corpo. Ainda no cabeçalho podem conter os meta-dados, compostos de *keywords* e descrição geral do conteúdo da página, porém este campo muitas vezes apresenta informações inconsistentes e não homogêneas.

No corpo da página está presente o texto do documento e outros elementos não textuais. Muitas pesquisas realizadas mostram que os componentes textuais destas páginas possuem informações suficientes para realização da classificação, os componentes não textuais podem ser utilizados para aprimorar a performance do classificador [31].

2.2.6.2 Características

A classificação de páginas web contém algumas peculiaridades que a difere da classificação de texto tradicional e a torna um problema mais difícil de obter bons resultados:

- Páginas web são semi-estruturadas através da linguagem de marcação HTML (*HyperText Markup Language*). Existem outros tipos de texto que possuem estrutura, porém numa classificação tradicional, geralmente, ele é removida. Através do HTML podem ser extraídas características consideradas cruciais para classificação das páginas, portanto deve analisada antes de realizar a remoção total.

- Documentos web possuem a estrutura de hiperlinks que forma uma estrutura de grafo, onde cada aresta representa a conexão de uma página com outra(s).

- Muitas vezes as páginas web não possuem informações textuais que sejam suficientes para realizar a classificação e não mencionam explicitamente o conteúdo ao qual se refere, ao contrário de documentos textuais que apresentam conteúdo necessário para realização correta da classificação.

- Muitos documentos são fragmentados devido a forma com que a web está estruturada. Esse é um dos motivos que explica baixo desempenho dos classificadores em página web. Um exemplo disto é o resultado de um experimento em [33], onde Chakrabarti comenta sobre a realização de um teste com duas bases, uma de documentos e outra com alguns exemplos do Yahoo!, que foram utilizadas como entrada para um mesmo classificador, porém a primeira apresentou 87% de acurácia e a segunda 32%.

2.2.6.3 Tipos de classificação

A classificação de páginas web podem ser feitas através de diferentes aspectos presentes na página. De acordo com Sun et at. [33] podemos dividir os tipos de classificação nas seguintes abordagens:

- Utilização de links: analisa conteúdo presente no link da página ou utiliza a estrutura de link para classificar a página baseada na proximidade da mesma com páginas que já foram classificadas previamente

- Hipertexto: neste método além da parte textual são utilizadas informações presentes na estrutura da página web, como título e *tags* HTML.

- Utilizar somente o texto: método mais simples que utiliza apenas conteúdo textual presente na página.

- Páginas vizinhas (links): neste método além das informações da página são utilizadas informações extraídas das páginas vizinhas.

2.2.6.4 Trabalhos Relacionados

Muitas abordagens foram desenvolvidas para resolver o problema de classificação de páginas web. Essas abordagens apresentam um conjunto de técnicas das áreas de processamento de linguagem natural, mineração de texto e aprendizagem de máquina. Abaixo temos alguns trabalhos e seus resultados desenvolvidos nesta área.

- Chakrabarti at. al. [30] foi propuseram a realização da classificação de páginas web baseada exclusivamente na estrutura das páginas, ignorando completamente o seu conteúdo. A quantidade de acerto através desta abordagem foi considerada baixa quando comparada a classificadores que consideram o conteúdo da página. Através deste trabalho é perceptível que no contexto web análise de estrutura não é suficiente, diferentemente de alguns problemas de classificação de texto tradicional.

- Xue at. al. [35] analisaram diferentes formas de representar uma página web utilizando SVM, com variação na função de *kernel*, como classificador. Foi analisado o uso de metadados, título, texto, informações do cabeçalho e a combinação dessas características para melhorar o resultado da classificação. Os autores concluíram que o uso conjunto das *features* citadas aprimoram bastante o resultado. Essas *features* foram incluídas como novas dimensões, mesmo se já houver uma dimensão com a mesma *feature* no corpo da página. Outra característica observada que difere da classificação tradicional foi o modo de representação da matriz termo-documento. Os melhores resultados obtidos foram com a utilização da frequência dos termos (TF) sem o uso de TF-IDF.

- Em [37] foi realizado um estudo sobre a utilização de conteúdo apenas de sites vizinhos. O método proposto obteve bons resultados, para que ele funcione é necessário realizar uma coleta de uma boa quantidade de páginas que estão relacionadas com a página que deve ser classificada. Apesar de obterem bons resultados, na web lidamos com um grande volume de informação portanto na prática seria inviável.

- Kan [36] propôs a classificação de páginas web através das informações presentes na URL. Para obter informações a URL é dividida em *tokens* com a finalidade de identificar quais deles são úteis para classificação. Apesar de melhorar a performance a avaliação da proposta foi realizada através de uma base bastante limitada, portanto não foi levado em consideração a escalabilidade.

CAPÍTULO 3

Solução

Neste trabalho foi desenvolvido um classificador de páginas web em português restrito ao domínio específico de imóveis. A classificação se divide em duas categorias: um classificador para identificar se o imóvel está anunciado para aluguel ou venda e o outro para identificar o tipo do imóvel (casa, apartamento, outros).

A solução proposta tem como base de classificação apenas o texto, título e a URL presente na página *web*. Na classificação de texto lidamos com alguns desafios (ver Seção 2.2.2) como: alta dimensionalidade, vetores de características esparsos, redundância e complexidade linguística. Além desses desafios presentes na classificação de texto tradicional existem as dificuldades da classificação de páginas web através do texto, muitas abordagens tem sido utilizadas para se obter melhores resultados nesta área (ver Seção 2.2.6.4).

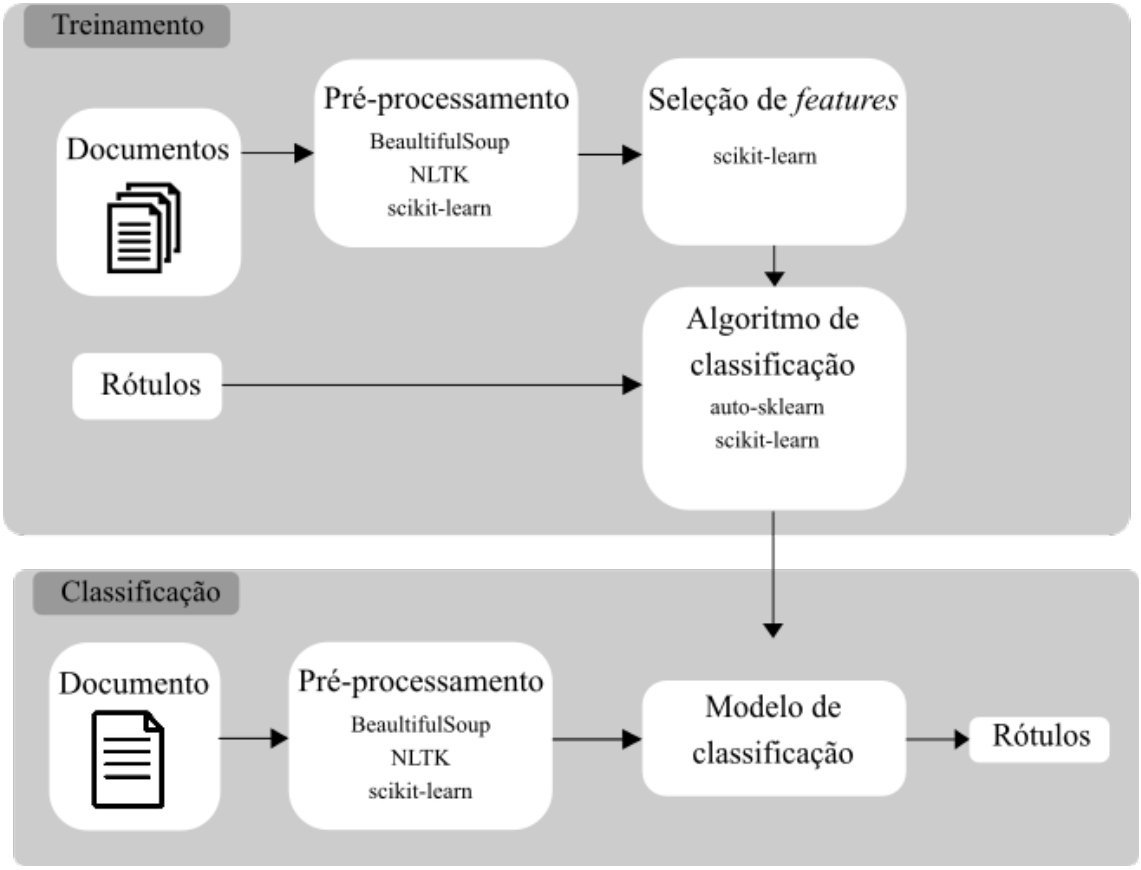

Figura 3.1 Etapas para desenvolvimento da solução.

Na Figura 3.1 temos o passo a passo da solução desenvolvida. A construção da solução se divide em duas etapas: o treinamento e a classificação propriamente dita. Na fase de treinamento um conjunto de documentos é submetido às técnicas de pré-processamento, onde passa a ser representado de forma apropriada para classificação. Em seguida passam por uma redução de dimensionalidade, com o intuito de tornar o problema menos complexo considerando as dimensões (*tokens*) que mais distinguem as classes. Após essas duas etapas os documentos serão utilizados como *input* para o algoritmo de classificação que apresentará um modelo para o problema. Na fase de classificação são apresentados documentos que não possuem rótulos. Estes documentos são submetidos às mesmas técnicas de pré-processamento da fase de treinamento e em seguida são apresentados ao modelo. O modelo construído na fase anterior é responsável por fornecer os rótulos que mais caracteriza o(s) documento(s) apresentado(s).

Neste capítulo será apresentada detalhes de implementação de cada etapa necessária para construção da solução.

3.1 Páginas web no domínio de imóveis

Para construção de um classificador é necessário obter um conjunto de dados (amostra) para realizar a etapa de treinamento. Nesta etapa os dados são utilizados como *inputs* com a finalidade do algoritmo de classificação "aprender"sobre o problema, ou seja, ser capaz de identificar cada classe através das características das páginas apresentadas.

A base de dados utilizada para realização do treinamento foi obtida através de um coletor intra-site para entidades imobiliárias, que tem como objetivo localizar páginas web de entidades estruturadas em um dado site, apresentado em [32]. Este coletor possui amostras de diferentes domínios (sites) o que é fundamental para o classificador ter a capacidade de generalizar cada categoria.

Figura 3.2 Exemplo de página web no setor de imóveis. Fonte:imobiliariacasaeimoveis.com.br

3.2 Ferramentas

Nesta seção será dada uma breve descrição das bibliotecas utilizadas para desenvolvimento deste trabalho.

BeaultifulSoup [40]

Biblioteca python projetada para facilitar a extração de dados nos documentos HTML e XML. Possui um conjunto de métodos que facilita a manipulação, navegação e pesquisa da informação no HTML. Apresenta alta robustez quando lida com arquivos mal formatados.

NLTK (*Natural Language Toolkit*) [41]

É um conjunto de ferramentas *open source* em python para a manipulação de linguagem natural, originalmente criado em 2001 no *Department of Computer and Information Science da University of Pennsylvania*. Tem sido desenvolvido e ampliado com a ajuda de dezenas de contribuintes. O NLTK é organizado em uma coleção de componentes para tarefas específicas. Cada módulo é uma combinação das estruturas de dados para a representação de um tipo particular de informação. Nesta biblioteca podemos encontrar inúmeros recursos, como: processamento de strings, *stemming*, *tagging parse* e raciocínio semântico.

scikit-learn [42]

Originada de um projeto do *Google Summer of Code* por David Cournapeau, o scikit-learn é uma biblioteca composta por uma grande variedade de algoritmos de Aprendizado de Máquina, tanto supervisionados quanto não supervisionados. Esses algoritmos são utilizados através de uma interface orientada à tarefas que permite fácil comparação entre os métodos. Pode ser facilmente integrada aos mais diversos tipos de aplicação. Os algoritmos implementados em alto nível podem ser aplicados com maior facilidade para casos de uso específicos.

XGBoost [43]

XGBoost é uma biblioteca *open source* de Aprendizado de Máquina (AM) composta por vários algoritmos de classificação, regressão e agrupamento. Esta biblioteca tem sido amplamente utilizada em vários desafios da área de AM e Mineração de Dados, como por exemplo, os desafios organizados pelo site Kaggle [25]. As característica mais importantes desta biblioteca é o alto poder de escalabilidade em diferentes cenários e o tempo de execução. Executa o algoritmo 10 vezes mais rápido do que as soluções tradicionais e escala para bilhões de exemplos em configurações distribuídas ou com limitação de memória.

Auto-sklearn [44]

O *auto-sklearn* é um tipo de AutoML composto por um total de 15 classificadores, 14 métodos de pré-processamento de atributos, lida com a escala de dados, codificação de parâmetros categóricos e valores faltantes. Este AutoML utiliza meta-aprendizado para identificar conjunto de dados semelhantes e utilizar o conhecimento previamente desenvolvido. Para melhorar a generalização ele constrói um conjunto de todos os modelos testados durante o processo de otimização.

3.3 Pré-processamento

Antes de representar cada documento como um vetor de características foram analisadas as estruturas das páginas presentes na amostra. Páginas *web*, em geral, podem ser compostas de *javascript* e *css*, além do conteúdo. Esses dois elementos, identificados através das *tags* <script> e <style>, são removidos nesta etapa por não acrescentarem valor para atividade de classificação.

O cabeçalho das páginas muitas vezes possuem fortes indicativos das classes. Estas informações podem estar na descrição, título ou até mesmo URL [34]. Para os classificadores desenvolvidos estas informações foram identificadas e inseridas posteriormente como novas dimensões na matriz termo-documento. Após a identificação destas características foram removidas as *tags* HTML.

Em seguida foram removidas as *stopwords* e aplicado o método de *stemming*. A *stoplist* utilizada, através do pacote nltk.corpus, é composta por 203 *stopwords* em português. O algoritmo de *stemming* utilizado foi o RSLP [20], disponível no pacote nltk.stem, que através de um conjunto de regras aplicadas sucessivamente remove o sufixo de uma dada palavra. Além de possuir um conjunto de regras especificas para o português possui um dicionário de exceções para evitar a remoção de palavras com terminação similar a um sufixo.

Por fim o documento passa a ser representado através do modelo *bag-of-words*, ou seja, uma coleação de palavras. Através desta coleção é definida a matriz termo-documento com o auxilio do método *TfIdfVectorize* do sklearn. Os valores presentes em cada posição da matriz podem variar de acordo com a *feature* escolhida (ver Seção 2.2.1). Neste trabalho foram utilizadas as *features* TF-IDF (bastante utilizada na classificação de texto tradicional) e Log TF (que apresentou bons resultados em páginas web com o SVM [35]).

Portanto podemos resumir a etapa de pré-processamento nos seguintes passos:

1. Remover javascript e css
2. Identificar palavras-chave na URL e título
3. Remover *stopwords*
4. Aplicar *stemming* ao *body* do documento
5. Gerar matriz termo-documento de acordo com a *feature* estabelecida

3.4 Seleção de *features*

Um dos desafios da classificação de texto é a alta dimensionalidade. Isso se deve ao fato da utilização do vocabulário, definido no conjunto de treinamento, como dimensões do vetor que representa o documento. As técnicas de remoção de *stopwords* e aplicação de *stemming*, aplicadas na etapa anterior, ajudam na redução da dimensionalidade. A primeira removendo os termos que são comuns a qualquer documento independente da classe; e a segunda reduzindo as palavras aos seus radicais, ou seja, agrupando as palavras através da redução para sua forma básica.

Na Tabela 3.1 podemos ver o efeito da aplicação dessas técnicas no vocabulário no conjunto de treinamento utilizado neste trabalho.

	Pré-processamento com redução de dimensão	**Pré-processamento sem redução de dimensão**
Tarefa 1	49562	59293
Tarefa 2	50102	59952

Tabela 3.1 Quantidade de *tokens* gerados por classe com aplicação de técnicas de redução de dimensionalidade

Apesar da redução de características no pré-processamento ainda é preciso realizar uma seleção mais criteriosa para se obter as características que mais distinguem os documentos: a seleção de *features* (ver Seção 2.2.4). Nesta etapa foi utilizado o método SelectKBest, da biblioteca sklearn, que realiza um ranqueamento das k melhores *features* de acordo com a medida escolhida. A medida utilizada para selecionar as melhores características neste problema foi a medida estatística X^2 (ver Seção 2.2.4) com o valor de k fixado em 4000 dimensões.

3.5 Técnica de classificação

Para definir as melhor técnica de classificação foram realizados um conjunto de experimentos (Ver Capítulo 5) com 5 tipos de técnicas diferentes (SVM, Naive Bayes, Random Forest, XGBoost e ensamble/AutoML). A técnica que obteve melhor resultado para solução do problema foi o ensamble/AutoML.

O classificador final é composto por um comitê de classificadores que obtiveram seus parâmetros e etapas de pré-processamento definidas de forma automática. Tanto para tarefa 1, que tem como objetivo definir o tipo do anúncio, quanto para tarefa 2, que tem como objetivo definir o tipo do imóvel, o desenvolvimento do modelo foi feito através da biblioteca auto-sklearn (ver Seção 3.2).

Para tarefa 1 o ensemble gerado é composto pelos algoritmos AdaBoost, LDA e SVM. Já para tarefa 2 o ensemble é composto apenas por dois algoritmos AdaBoost e SGD. Os algoritmos utilizados na solução final foram selecionados d forma automática através da biblioteca auto-sklearn.Na Seção 4.2 são apresentados detalhes técnicos desta solução e a comparação com as técnicas tradicionais em Classificação de Texto.

CAPÍTULO 4

Experimentos

Para definir a configuração de representação dos documento e os melhores classificadores para o problema de classificação de páginas *web* no setor imobiliário foi realizado um conjunto de experimentos com variações na forma de representação do documento aplicadas a diferentes técnicas de classificação.

Neste capítulo serão dadas informações mais especificas sobre a base de dados utilizada para treinamento/teste, quais variações na forma de representação do documento foram testadas e a comparação de algumas técnicas de classificação aplicadas a este problema.

4.1 Detalhes experimento

Base de dados

A base de dados obtida através do coletor apresentado em [32] é composta por diferentes domínios (sites). Para o desenvolvimento deste projeto foram selecionadas cerca de 3000 páginas web de 77 domínios distintos. A cada página foram atribuídos dois rótulos: um referente a tarefa 1 (venda/aluguel) e outro referente a tarefa 2 (apartamento/casa/outros).

Venda	**Aluguel**
2392	804
Total 3196	

Tabela 4.1 Distribuição de documentos da tarefa 1

Apartamento	**Casa**	**Outros**
1675	1492	117
Total 3284		

Tabela 4.2 Distribuição de documentos da tarefa 2

A partir das Tabelas 4.1 e 4.2 é possível perceber que o coletor possui pouca variação de algumas categorias que possivelmente seriam utilizadas neste projeto. Na Tabela 4.1 temos como maioria dos exemplos a classe de Venda. Já na Tabela4.2 temos como minoria páginas que pertencem a classe de Outros, que é composta por imóveis do tipo terreno, fazenda, loja

e galpão. Uma possível explicação para escassez de tipos imóveis é a seleção de *seeds* (urls base) que foram utilizadas pelo coletor para realizar a coleta das páginas.

Campo	**Descrição**
domain	site ao qual o documento pertence
document	body do documento html com remoção das tags e caracteres especiais
url	url referente a página
title	título da página
class1	rótulo referente a Classe 1
class2	rótulo referente a Classe 2

Tabela 4.3 Campos do arquivo csv com documentos e suas respectivas classes

Para facilitar o acesso aos documentos foi criado um arquivo csv com as informações presentes na Tabela 4.3 antes de aplicar as técnicas de pré-processamento. O conjunto de treinamento e teste foram divididos por domínios: 70% dos domínios foram utilizados para treinamento e 30% para teste. A partir dessa divisão é garantido que no conjunto de testes o classificador irá lidar com páginas de sites nunca vistos antes. Nas Tabelas 4.4 e 4.5 temos as distribuições dos documentos para o conjunto de treinamento e de teste das tarefas 1 e 2 respectivamente.

	Treinamento	**Teste**
Venda	1732	654
Aluguel	576	227

Tabela 4.4 Distribuição de documentos da tarefa 1 para treinamento e teste

	Treinamento	**Teste**
Apartamento	1120	552
Casa	1200	288
Outros	76	41

Tabela 4.5 Distribuição de documentos da tarefa 2 para treinamento e teste

Pré-processamento

Nesta etapa de experimentos foi construído um conjunto de matrizes termo-documento através das variações dos parâmetros do método TFIdfVectorize. Os tipos de *features* analisados foram Log TF e TF-IDF. Com o proposito de analisar o efeito da aplicação de *stemming* e a remoção de *stopwords* foram geradas matrizes com e sem a aplicação destas técnicas. Alguns parâmetros do TfIdfVectorize foram fixados durante os experimentos sendo eles:

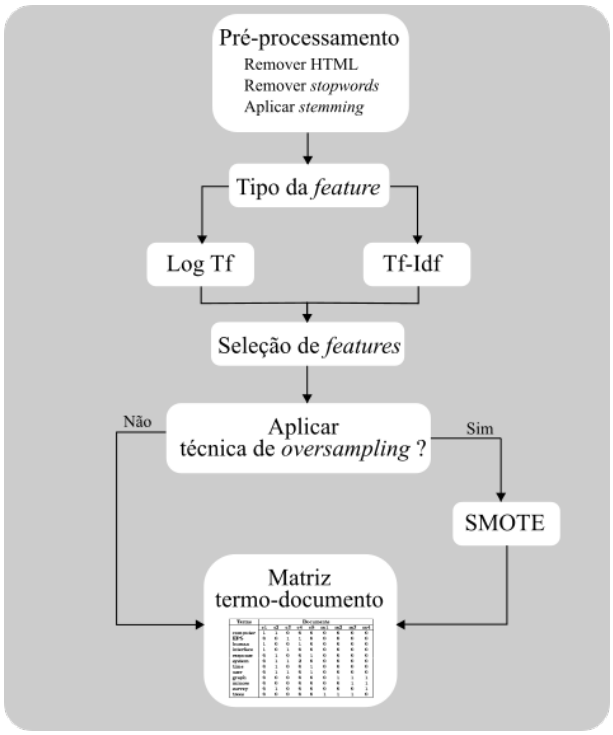

Figura 4.1 *Pipeline* de criação das matrizes termo-documento para os experimentos

- ngram_range: define os valores mínimo e máximo de n para criação de n-grams. Os valores de mínimo e máximo foram fixado em 1 e 2.
- max_df/min_df: foram definidos como 800 e 1, respectivamente. Estes parâmetros ignoram os termos que possuem uma frequência de documento superior e inferior ao *threshold* recebido como entrada.

Como o conjunto de dados é desbalanceado foi aplicada uma técnica de *oversampling*: *Synthetic Minority Over-sampling Technique* (SMOTE). Esta técnica de *oversampling* envolve a criação de dados artificiais da classe minoritária com base nas características espaciais dos exemplos de treinamento com o objetivo de melhorar o desempenho da classificação [39]. Os novos exemplos são criados separando, de forma aleatória, algumas amostras da classe minoritária e utilizando os vizinhos mais próximos de cada padrão da amostra, calculados a partir do algoritmo *k-Nearest Neighbors* (kNN).

Técnicas de classificação

A escolha do melhor classificador para este problema foi feita baseada nos resultados dos experimentos com as diferentes formas de representação presentes na Figura 4.1. Foram selecionadas três técnicas de classificação tradicionais em Classificação de Texto (SVM, Naive Bayes e Random Forest) e duas técnicas que estão se destacando bastante em competições de Aprendizado de Máquina (XGBoost e auto-sklearn).

Durante o treinamento dos classificadores foi utilizada a biblioteca Scikit-learn [42]. Um dos desafios para se obter uma boa acurácia é a definição dos parâmetros dos classificadores. Para encontrar os melhores parâmetros para este problema de classificação foi utilizado o GridSearchCV, que realiza uma busca exaustiva considerando todas as combinações possíveis, para todos os classificadores exceto o auto-sklearn que já possui otimização de parâmetros durante a etapa de treinamento.

4.2 Resultados

Na primeira etapa do experimento foram testadas todas as matrizes termo-documento com os cinco classificadores com a configuração *default*, ou seja, não foi realizada nenhuma otimização dos parâmetros. Nesta primeira etapa cada algoritmo de configuração teve 8 tipos distintos de *inputs* definidos na Figura 4.1. Na segunda etapa foi realizada a otimização dos parâmetros dos classificadores com a finalidade de aprimorar os resultado obtidos. Para avaliação dos classificadores foram utilizadas as métricas: acurácia, f1, precisão e recall (ver Seção 2.1.2).

Nesta seção serão mostrados os resultados obtidos e uma avaliação mais detalhada dos classificadores que obtiveram melhor resultado.

Melhores Resultados

Nas Tabelas 4.6 e 4.7 estão as melhores configurações da forma de representação dos documentos para cada algoritmo para a tarefa 1 e tarefa 2 respectivamente. Nesta etapa foram analisadas somente a acurácia como métrica de desempate dos classificadores.

Classificador	**Stemming/Remover stopwords**	**SMOTE**	**Feature**	**Acurácia**
SVM	Sim	Sim	Log TF	80.70%
Random Forest	Sim	Não	TF-IDF	81.61%
Naive Bayes	Sim	Sim	TF-IDF	80.36%
XGBoost	Sim	Sim	Log TF	87.74%
ensemble/autoML	Sim	Não	TF-IDF	88.60%

Tabela 4.6 Melhores resultados dos classificadores para tarefa 1

Na Tabela 4.6, que apresenta melhores resultados para tarefa 1, podemos observar que existe uma unanimidade em relação a aplicação das técnicas de pré-processamento (remoção de *stopwords* e aplicação de *stemming*). A aplicação do método de SMOTE nem sempre apresentou uma melhora significativa nos classificadores. Como esperado, com base no trabalho trabalho [35], o SVM obteve seu melhor resultado com a frequência dos termos (TF). Apesar dos métodos tradicionais apresentarem um bom desempenho o XGBoost e o ensemble gerado pelo auto-sklearn obtiveram melhores resultados com a diferença de cerca de 7% na acurácia. O classificador gerado através do auto-skleran apresentou o melhor resultado para tarefa 1 com uma acurácia de 88.60%.

Classificador	Stemming/Remover stopwords	SMOTE	Feature	Acurácia
SVM	Sim	Não	Log TF	81.72%
Random Forest	Sim	Sim	TF-IDF	80.02%
Naive Bayes	Não	Sim	TF-IDF	70.06%
XGBoost	Sim	Não	TF-IDF	91.60%
ensemble/autoML	Sim	Não	TF-IDF	92.73%

Tabela 4.7 Melhores resultados dos classificadores para tarefa 2

Na Tabela 4.7 temos os melhores resultados para tarefa 2 (casa, apartamento e outros). A maioria dos classificadores obtiveram bons resultados com a aplicação de *stemming* e remoção de *stopwords* com exceção do *Naive Bayes*. Novamente a melhor configuração dos documentos para o SVM foi a frequência dos termos. Similar aos resultados da tarefa 1, o melhor classificador obtido foi o ensemble gerado pelo auto-sklearn, sem a utilização do SMOTE, alcançando uma acurácia de 92.73%.

Piores Resultados

Classificador	Stemming/Remover stopwords	SMOTE	Feature	Acurácia
SVM	Não	Sim	Log TF	72.19%
Random Forest	Não	Não	Log TF	75.02%
Naive Bayes	Não	Não	Log TF	74.34%
XGBoost	Não	Sim	Log TF	84.44%
ensemble/autoML	Sim	Sim	Log TF	83.54%

Tabela 4.8 Piores resultados dos classificadores para tarefa 1

Classificador	Stemming/Remover stopwords	SMOTE	Feature	Acurácia
SVM	Não	Não	TF-IDF	32.69%
Random Forest	Não	Não	TF-IDF	60.49%
Naive Bayes	Não	Sim	Log TF	41.74%
XGBoost	Sim	Sim	TF-IDF	85.13%
ensemble/autoML	Sim	Sim	TF-IDF	83.42%

Tabela 4.9 Piores resultados dos classificadores para tarefa 2

Nas Tabelas 4.8 e 4.9 percebemos que a maioria dos piores resultados obtidos ocorreram com a retirada de técnicas de pré-processamento que auxiliam na redução da dimensionalidade. Nas duas tarefas o pior classificador foi o SVM. Para tarefa 1 a configuração que mais

prejudicou o desempenho da maioria dos classificadores foram as configurações que utilizaram o método de SMOTE e a *feature* Log TF. Já para tarefa 2 o uso da *feature* TF-IDF sem aplicação de técnicas de pré-processamento foram as que mais prejudicaram a atividade de classificação.

Classificadores selecionados para o problema e suas métricas

Através da analise dos melhores resultados podemos perceber que o ensemble gerado pelo auto-sklearn apresentou melhor desempenho para ambas as classes. Tanto para classe 1 quanto para classe 2 a melhor forma de representação do documento foi a utilização da *feature* TF-IDF e a aplicação das técnicas de pré-processamento. Na Tabela 4.11 estão presentes alguns parâmetros, que foram fixados, para geração do classificador.

time_left_for_this_task	tempo máximo, em segundos, para realizar a busca de modelos apropriados para o conjunto de dados apresentados. O valor utilizado foi 3600s.
per_run_time_limit	tempo limite, em segundos, para uma única chamada ao modelo de aprendizado. O ajuste do modelo é encerrado quando o algoritmo de aprendizado atinge esse tempo. Este valor foi fixado para 3000s.
ensemble_size	número máximo de modelos que podem ser adicionados ao ensemble foi fixado para 50.

Tabela 4.10 Parâmetros fixados para o ensamble/AutoML

Para tarefa 1 (aluguel/venda) o ensemble gerado é composto por quatro classificadores e três algoritmos distintos: AdaBoost, LDA e SVM (ver Seção 2.2.5). Abaixo temos um breve resumo dos dois primeiros algoritmos.

- **AdaBoost**: também conhecido como Boosting Adaptativo, é um meta-estimador que pode ser utilizado em conjunto com diversos algoritmos de aprendizado de máquina com objetivo de melhorar o desempenho em determinado conjunto de dados. Ele utiliza um algoritmo base em varias iterações, onde em cada iteração é atualizado a distribuição de pesos do conjunto de treinamento. O objetivo é aumentar o peso de exemplos incorretamente classificados em confronto com os pesos das instancias corretamente classificadas[46].

- **LDA** (*Linear Discriminant Analysis*): também conhecido como método de Fisher, tem como proposta encontrar o hiperplano que tem maior capacidade de separação entre as classes analisadas. O cálculo desse hiperplano de separação considera o conhecimento prévio da classe ou grupo de cada amostra [47]. Esse método se baseia na diminuição de espalhamento de instancias que pertencem a mesma classe e na maximização da distancia da média entre as diferentes classes.

Na Tabela 4.11 percebemos que o algoritmo de maior influencia de classificação para tarefa 1 é o AdaBoost composto por 307 árvores de regressão com profundidade máxima

Classificador	**Peso**	**Parâmetros**
AdaBoost	0.6	*n_estimators*: 307; *algorithm*: SAMME; *preprocessor*: select_percentile_classification
AdaBoost	0.2	*n_estimators*: 472; *algorithm*: SAMME; *preprocessor*: select_rates_fpr
LDA	0.1	*n_components*: 37; *preprocessor*: densifier
SVM	0.1	*kernel*: poly; *preprocessor*: no_preprocessing

Tabela 4.11 Detalhes do ensemble para tarefa 1

de 4 nós e com a utilização do algoritmo 'SAMME', que atualiza o modelo aditivo através da classificação. O ensemble possui duas instancias do algoritmo AdaBoost que utilizam diferentes técnicas de pré-processamento de *features*. O *select_percentile_classification* seleciona as *features* de acordo com um percentil de pontuação mais alta. Já o *select_rate_fpr* filtra as *features* com valores abaixo de um limiar com base no teste de FPR (*False Positive Rate*). O SVM deste ensemble utiliza o *kernel* polinomial que representa a semelhança de vetores sobre polinômios das variáveis originais, permitindo a aprendizagem não linear.

	Precisão	**F1**	**Recall**
Venda	89%	92%	96%
Aluguel	86%	74%	65%

Tabela 4.12 Métricas de desempenho do ensemble para tarefa 1

As métricas de desempenho do classificador para a tarefa 1, presentes na Tabela 4.12, mostram que a frequência de atribuição do rótulo correto para anúncios de aluguel é menor em relação a venda. Apesar de apresentar uma precisão bastante parecida percebemos, através da medida F1, que o classificador possui mais capacidade de identificar anúncios de venda do que de aluguel.

Para tarefa 2, que se refere aos rótulos Apartamento, Casa e Outros, o ensemble é composto por 5 classificadores e dois algoritmos diferentes: AdaBoost e SGD (*Stochastic Gradient Descent*).

O algoritmo SGD realiza atualização de um conjunto de parâmetros de forma iterativa para minimizar a função de erro. Para realizar esta atualização é utilizada apenas uma amostra do conjunto de treinamento. Devido a rápida convergência a função de erro pode não ser minimizada como no Gradiente Descentente (GD), que utiliza todo conjunto de treinamento para minimizar a função de erro. Apresenta bons resultados em problemas esparsos e de larga escala.

Na Tabela 4.13 o classificador de maior influencia na definição da classe de uma instancia é o AdaBoost. Neste ensemble nenhum dos classificadores aplicou pré-processamento. O conjunto dos classificadores do tipo AdaBoost apresentaram parâmetros bastante semelhantes apresentando varição apenas na profundidade da árvore de regressão, que variou de 4 a 9, e

Classificador	**Peso**	**Parâmetros**
AdaBoost	0.36	*n_estimators*: 82; *algorithm*: SAMME; *preprocessor*: no_preprocessing
SGD	0.28	*loss*: perceptron; *n_iter*: 154; *preprocessor*: no_preprocessing
AdaBoost	0.2	*n_estimators*: 133; *algorithm*: SAMME; *preprocessor*: no_preprocessing
AdaBoost	0.14	*n_estimators*: 166; *algorithm*: SAMME; *preprocessor*: no_preprocessing
AdaBoost	0.02	*n_estimators*: 411; *algorithm*: SAMME; *preprocessor*: no_preprocessing

Tabela 4.13 Detalhes do ensemble para tarefa 2

na quantidade de árvores, que ficou entre o intervalo de 82 a 411. O SGD utilizou uma função de perda denominada perceptron, utilizada para problemas linearmente separáveis.

	Precisão	**F1**	**Recall**
Casa	88%	92%	97%
Apartamento	95%	96%	97%
Outros	100%	9%	5%

Tabela 4.14 Métricas de desempenho do ensemble para tarefa 2

Através da analise dos resultados obtidos pelo segundo classificador percebemos que as categorias Casa e Apartamento apresentam altos valores para precisão e recall. Porém a categoria Outros obteve 100% de precisão (todos documentos rotulados como Outros foram preditos como outros), e baixo recall. O baixo recall se deve a baixa frequência com que o classificador encontra exemplos dessa classe. Como foi visto na Seção 4.1 a base é desbalanceada e possui poucos exemplos desta categoria fazendo com que o classificador não consiga generalizar muito os documentos da classe minoritária; na maioria das vezes é pode ser atribuído a ele o rótulo da categoria majoritária no treinamento.

Apesar da acurácia geral de 92.73% apresentada na Tabela 4.7 podemos perceber através da analise macro que não apresenta bons resultados para todas as classes.

CAPÍTULO 5

Conclusão

Neste trabalho foi apresentada uma solução para classificação de páginas *web* em português no setor imobiliário através do texto. Para tanto foi construído um classificador capaz de identificar duas características presentes nas páginas, sendo elas: tipo do anúncio (aluguel ou venda) e o tipo do imóvel que esta sendo anunciado (apartamento, casa e outros). O classificador apresentado como solução é um ensemble, construído com o auxilio da biblioteca auto-sklearn. Apresenta acurácia de 88.6% para classificação do tipo de anuncio e 92.73% para classificação do tipo de imóvel.

Durante o desenvolvimento da solução percebemos que muitos classificadores considerados tradicionais na área de Classificação de Texto, como o SVM ou Naive Bayes, mesmo após ajustes de parâmetros não obtiveram resultados melhores do que os classificadores desenvolvidos pelas bibliotecas XGBoost e auto-sklearn que vem apresentado bons resultados nos mais distintos conjuntos de dados em muitas competições de Aprendizado de Máquina.

O desbalanceamento das classes presente no conjunto de dados prejudicou a classificação das classes minoritárias. Como trabalho futuro o aumento da base de dados ou aplicação de outras técnicas de *oversampling*, além do SMOTE, poderia ser fundamental para aumentar o conjunto de rótulo e a capacidade de generalização para domínios distintos.

Outra proposta é a extração de outras *features*, além de URL e título, que podem ser extraídas das páginas web [35, 37]. A exploração de técnicas de seleção de *features* e analise do vocabulário através de visualização de dados pode ser fundamental para melhor compreensão das dimensões presentes no vetor que representam o documento.

Referências Bibliográficas

[1] Sebastiani, F.. *Machine learning in automated text categorization.*. ACM Computing Surveys, volume 34(1):pages 1–47, 2002.

[2] Aggarwal, C. C. *Data classification: algorithms and applications.*. Minneapolis, USA: CRC Press, 2014.

[3] Kodratoff, Y. *Introduction to machine learning*. Palo Alto, USA: Morgan Kaufmann, 2014.

[4] Zhu, Xiaojin, and Andrew B. Goldberg *Introduction to semi-supervised learning*. Synthesis lectures on artificial intelligence and machine learning 3.1 (2009): 1-130

[5] Mencia, Eneldo Loza, and Johannes Fürnkranz. *Efficient pairwise multilabel classification for large-scale problems in the legal domain*. Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin, Heidelberg, 2008.

[6] Powers, David Martin. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2011

[7] Dumais, Susan, et al. *Inductive learning algorithms and representations for text categorization*. Proceedings of the seventh international conference on Information and knowledge management. ACM, 1998.

[8] Yang, Yiming, and Jan O. Pedersen. *A comparative study on feature selection in text categorization. Icml*. Vol. 97. 1997.

[9] Joachims, Thorsten. *A Statistical Learning Model of Text Classification for SVMs*. Learning to Classify Text Using Support Vector Machines. Springer US, 2002. 45-74.

[10] Selvakuberan, K., M. Indradevi, and R. Rajaram. *Combined Feature Selection and classification–A novel approach for the categorization of web pages*. Journal of Information and Computing Science (2008).

[11] Salton, Gerard, and Michael J. McGill. *Introduction to modern information retrieval*. (1986)

[12] Viera, Angel Freddy Godoy, and Johnny Virgil *Uma revisão dos algoritmos de radicalização em língua portuguesa*. Information Research 12.3 (2007): 12-3.

[13] Monard, Maria Carolina, and José Augusto Baranauskas. "Conceitos sobre aprendizado de máquina."*Sistemas Inteligentes-Fundamentos e Aplicações* 1.1 (2003).

[14] Van der Aalst, Wil MP, et al. "Process mining: a two-step approach to balance between underfitting and overfitting."*Software & Systems Modeling* 9.1 (2010): 87.

[15] Bishop, Christopher M. *Pattern recognition and machine learning*. springer, 2006.

[16] Lorena, Ana Carolina, and André CPLF de Carvalho. "Estratégias para a combinaçao de classificadores binários em soluçoes multiclasses."*Revista de Informática Teórica e Aplicada* 15.2 (2008): 65-86.

[17] Rossi, Rafael Geraldeli. *Classificação automática de textos por meio de aprendizado de máquina baseado em redes*. Diss. Universidade de São Paulo, 2016.

[18] Wallach, Hanna M. "Topic modeling: beyond bag-of-words." *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006.

[19] Gabrilovich, Evgeniy, and Shaul Markovitch. "Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4. 5."*Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004.

[20] COELHO, Alexandre Ramos. "Stemming para a língua portuguesa: estudo, análise e melhoria do algoritmo RSLP. 2007. 69f."Monografia (Graduação em Ciência da Computação)-Universidade Federal do Rio Grande do Sul, Porto Alegre (2007).

[21] Gonçalves, Lea Silvia Martins. *Categorização em text mining*. Diss. Universidade de São Paulo, 2002

[22] Aggarwal, Charu C., and ChengXiang Zhai. "A survey of text classification algorithms." *Mining text data* (2012): 163-222.

[23] Kinto, Eduardo Akira, and Emílio Del Moral Hernandez. "Classificadores de Texto Reduzido Basedados em SVM."*Proceedings of the VII CBRN-Congresso Brasileiro de Redes Neurais, Natal*. 2005.

[24] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine."*Annals of statistics* (2001): 1189-1232.

[25] Kaggle: The Home of Data Science & Machine Learn, `https://www.kaggle.com`

[26] Ditterrich, T. G. "Machine learning research: four current direction."*Artificial Intelligence Magzine* 4 (1997): 97-136.

[27] Santana, Laura Emmanuella Alves dos Santos. "Otimização em comitês de classificadores: uma abordagem baseada em filtro para seleção de subconjuntos de atributos."(2012).

[28] Feurer, Matthias, et al. "Efficient and robust automated machine learning."*Advances in Neural Information Processing Systems*. 2015.

[29] Eriksson, Tobias. "Automatic web page categorization using text classification methods."(2013).

[30] Chakrabarti, Soumen, Byron Dom, and Piotr Indyk. "Enhanced hypertext categorization using hyperlinks."*ACM SIGMOD Record*. Vol. 27. No. 2. ACM, 1998.

[31] Selvakuberan, K., M. Indradevi, and R. Rajaram. "Combined Feature Selection and classification–A novel approach for the categorization of web pages."*Journal of Information and Computing Science* (2008).

[32] B. Andaluz, "Um Coletor Inteligente para Entidades Imobiliárias Estruturadas em Sites", 2016.

[33] Sun, Aixin, Ee-Peng Lim, and Wee-Keong Ng. "Web classification using support vector machine."*Proceedings of the 4th international workshop on Web information and data management*. ACM, 2002.

[34] Asirvatham, Arul Prakash, et al. "Web page classification based on document structure."*IEEE National Convention*. 2001.

[35] Weimin Xue, Weitong Huang, and Yuchang Lu. Application of svm in web page categorization. In *Grandular Computing, 2006 IEEE International Conference on*, pages 469-472.IEEE, May 2006.

[36] Kan, Min-Yen. "Web page classification without the web page."*Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004.

[37] Glover, Eric J., et al. "Using web structure for classifying and describing web pages."*Proceedings of the 11th international conference on World Wide Web*. ACM, 2002.

[38] DE BRITO, EDELEON MARCELO NUNES. "Mineração de Textos: detecção automática de sentimentos em co-mentários nas mídias sociais."*Projetos e Dissertações em Sistemas de Informação e Gestão do Conhecimento* 6.1 (2017).

[39] Schiavoni, André Spinelli. "Um estudo comparativo de métodos para balanceamento do conjunto de treinamento em aprendizado de redes neurais artificiais."

[40] Beautiful Soup
`https://www.crummy.com/software/BeautifulSoup/`

[41] Natural Language Toolkit (NLTK)
`http://www.nltk.org/`

[42] Scikit-learn: Machine Learning in Pyhton
`http://scikit-learn.org/stable/`

[43] XGBoost: Scalable and Flexible Gradient Boosting
`http://xgboost.readthedocs.io/en/latest/`

[44] Auto-sklearn
`https://automl.github.io/auto-sklearn/stable/`

[45] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification."*AAAI-98 workshop on learning for text categorization*. Vol. 752. 1998.

[46] Schapire, Robert E. "The boosting approach to machine learning: An overview."*Nonlinear estimation and classification*. Springer New York, 2003. 149-171.

[47] Altman, Edward I., Giancarlo Marco, and Franco Varetto. "Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience)."*Journal of banking finance* 18.3 (1994): 505-529.

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