UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
IGCE - Instituto de Geociências e Ciências Exatas
Câmpus de Rio Claro - SP

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

APRENDIZADO DE MÁQUINA PARA DETECÇÃO DE SPAM

UM ESTUDO COMPARATIVO DE ALGORITMOS DE MINERAÇÃO DE TEXTO E CLASSIFICADORES.

Rio Claro - SP
2020

THIAGO GIROTO MILANI

APRENDIZADO DE MÁQUINA PARA DETECÇÃO DE SPAM

UM ESTUDO COMPARATIVO DE ALGORITMOS DE MINERAÇÃO DE TEXTO E CLASSIFICADORES.

Dissertação apresentada como parte do requisito para obtenção do título de Mestre em Ciência da Computação junto ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Geociências e Ciências Exatas, da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de Rio Claro.

Prof. Dr. Fabricio Aparecido Breve
Orientador

Rio Claro - SP
2020

M637a

Milani, Thiago Giroto
Aprendizado de Maquina para detecção de Spam : Um Estudo Comparativo de Algoritmos de Mineração de Texto e Classificadores / Thiago Giroto Milani. -- Rio Claro, 2020
58 p. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Geociências e Ciências Exatas, Rio Claro
Orientador: Fabricio Aparecido Breve

1. Segurança Computacional. 2. Reconhecimento de Padrão. 3. Extração de Características. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Geociências e Ciências Exatas, Rio Claro. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Thiago Giroto Milani

Aprendizado de Máquina para Detecção de Spam: um estudo comparativo de algoritmos de mineração de texto e classificadores

Dissertação apresentada como parte do requisito para a obtenção do título de Mestre em Ciência da Computação junto ao programa de Pós-Graduação em Ciência da Computação do Instituto de Geociências e Ciências Exatas, da Universidade Estadual Paulista "Julho de Mesquita Filho", Campus de Rio Claro.

Conceito: Aprovado.

Banca Examinadora

Prof. Dr. Fabricio Aparecido Breve (Orientador)
Departamento de Estatistica, Matemática Aplicada e Computação
Universidade Estadual Paulista - UNESP

Prof. Dr. Verônica Oliveira de Carvalho
Departamento de Estatistica, Matemática Aplicada e Computação
Universidade Estadual Paulista - UNESP

Prof. Dr. João Roberto Bertini Junior
Faculdade de Tecnologia
Universidade de Campinas - UNICAMP

Rio Claro, 28 de Janeiro de 2020.

Agradecimentos

Agradeço primeiramente à Deus, meus pais e amigos que me insentivaram a enfrentar essa dificil jornada em busca do conhecimento. Aos Professores que não me deixaram desistir, e a todos que de uma forma direta e indireta sempre me apoiaram, foram compreencivos, e insentivadores para que eu conseguice passar por mais essa etapa da minha vida, principalmente em tempos tão difíceis nos quais vivemos em virtude da pandemia de COVID-19.

"A mente que se abre para algo novo nunca mais será a mesma"
(Albert Einstein)

Resumo

Com o grande crescimento da área de informática e inovação tecnológica (era digital), cresce cada vez mais a necessidade de dispositivos e algoritmos capazes de aprender e reconhecer padrões. A segurança computacional se torna cada vez mais essencial com toda essa evolução, pois os incidentes de segurança estão se tornando cada vez mais comum. Um exemplo são as mensagens de *spam*, podendo trazer conteúdos impróprios ou indesejados e causando diversos problemas ou até mesmo roubo de informação. Baseado nisso se torna cada vez mais necessário o estudo dessas duas áreas em conjunto. Aprendizado de máquina e segurança computacional, o que possibilita a criação de novos dispositivos e ferramentas capazes de reconhecer padrões de incidentes de segurança através da inteligência computacional. Assim, é proposto neste trabalho efetuar a extração de características (vetorização de texto), que tem a finalidade de efetuar a extração dos termos mais relevantes, e posteriormente combiná-los com algoritmos de aprendizado de máquina semi-supervisionados, como o objetivo de estudar qual combinação é mais viável para a detecção de *spam*.

Palavras-chave: Segurança Computacional. Reconhecimento de Padrão. Extração de Características.

Abstract

The boom of technological innovation (digital era) has imposed the need for devices and algorithms that learn and recognize patterns. Driven by such evolution, computer security has become an essential, once incidents regarding computer security have been increasing even faster than technology itself. An example are spam messages, which may display inappropriate content, or even cause damage or data theft. Therefore, it is important to integrate both machine learning and computer security to create new devices and tools that are able to recognize patters of computer security incidents by using computer intelligence. To do so, this study aims to carry out a feature extraction process (text vectorization) of features that extract relevant terms and combine them as semi-supervised machine learning algorithms, analyzing which combination is the most viable to detect spam.

Keywords: Computer Security, Pattern Recognition, Feature Extraction.

Lista de Figuras

Nomenclature

IG *Information Gain*

LABELPROP Label Propagation

LLGC Learning with Local and Global Consistency

LNP Linear Neighborhood Propagation

MD Mineração de Dados

MLP *Multilayer Perceptron*

MT Mineração de Texto

NB *Naive Bayes*

PCC Competição e Cooperação entre Partículas

PNL Processamento de Linguagem Natural

SVD *Singular Value Decomposition*

SVM *Supprt Vector Machine*

TF *Term Frequency*

TF-IDF *Term Frequency - Inverse Document Frequency*

UCE - *Unsolicited Commercial e-mail*

Sumário

1 Introdução

Cotidianamente vemos um grande avanço na utilização de algoritmos e modelos computacionais para resolver problemas envolvendo tecnologia. Isso, influenciou o crescimento de pesquisas nas áreas de aprendizado de máquina e inteligência artificial, sendo cada vez mais frequente a criação de grupos e eventos internacionais e nacionais ligados ao assunto.

Cada vez mais os usuários de computador têm se deparado com problemas mais sérios de segurança computacional e redes. Com o crescimento da internet e cada vez mais serviços e aplicações sendo executados em nuvem, os usuários se tornam mais dependentes dos serviços de comunicação *on-line* como, *e-mail* e aplicativos de mensagens instantâneas. Com isto, cria-se também um grande nicho para pessoas maliciosas, as quais utilizam de diversas técnicas para envio de mensagens de *spam* (mensagem recebida contendo informações genéricas, com propagandas ou informações com o objetivo de enganar o leitor para extrair informações [Cert.br, 2016]) e *phishing* (técnica de persuasão utilizada para convencer a pessa a fornecer informações, geralmente sesíveis, como dados bancários, senhas, endereços entre outros [Vechia, 2014]), com o intuito de enganar o usuário e obter informações sigilosas, como contas bancárias, cartões de crédito, fotos íntimas entre muitos outros dados pessoais.

Existem várias ferramentas de detecção no mercado, porém poucas incorporam técnicas de aprendizado de máquina para o aprimoramento automático. A maioria das ferramentas e dispositivos em geral, apresentam problemas de desempenho e são principalmente baseados em comparação de assinaturas em banco de dados já existentes ou gerados pelo sistema [Cert.br, 2016]. Para se obter um maior desempenho é preciso um maior conjunto de dados de assinaturas, porém, com um alto custo em eficiência. Outro grande problema com essas ferramentas, é o grande número de falsos positivos gerados [Moll, 2010].

1.1 Objetivos

Este projeto tem como objetivo principal, comparar diversas combinações de extratores de características e classificadores, com o propósito de analisar as características extraídas das mensagens de *spam* através de algoritmos de aprendizado de máquina, a fim, de classificá-

las. Além disso, busca-se analisar a possibilidade da utilização de menor quantidade de dados rotulados, mantendo a mesma acurácia e tempo de execução, visando reduzir o esforço humano para rotular os dados manualmente para o treinamento.

Foram utilizados dois algoritmos de extração de características combinados com quatro algoritmos de aprendizado semi-supervisionado, onde, será feita a análise do desempenho de cada combinação pela acurácia e tempo de execução.

1.2 Organização do Documento

A organização dessa dissertação será descrita a seguir. O Capítulo 2 irá mostrar uma visão geral sobre segurança computacional e o tipo de dado a ser trabalhado, trazendo o conceito de *Phishing* e mensagens de *spam*, assim como, uma breve descrição de seu funcionamento. No Capítulo 3 é explicado sobre a mineração de texto, processamento de linguagem natural, introduzido o conceito de extração de características com a sua importância para a detecção de mensagens de *spam* e os algoritmos de extração de características que serão utilizados. O 4º Capítulo traz uma introdução sobre o tema de aprendizado de máquina, explicando os objetivos desta área, assim como o seus conceitos principais. Nos subcapítulos serão apresentadas as abordagens do aprendizado não-supervisionado e semi-supervisionado, e os algoritmos que serão utilizados na fase de classificação. Posteriormente, no Capítulo 5 será apresentada uma revisão bibliográfica de estudos sobre detecção de *spam* com aprendizado de máquina contidos na literatura, no Capítulo 6 é explicada qual a metodologia utilizada, uma breve explanação sobre os conjuntos de dados (*dataset*) que foram utilizadas e no Capítulo 7, resultados obtidos. E por fim no Capítulo 8 temos as conclusões obtidas com a pesquisa e suas possibilidades de futuras pesquisas.

2 Segurança Computacional

De acordo com a Cert.br [2016], é essencial a preocupação com a segurança, pois os computadores podem ser utilizados para diversos fins, que necessitam de maior controle e sigilo. Os computadores são amplamente utilizados para transações financeiras, comunicação e armazenamento de dados confidenciais, entre muitas outras tarefas, e sempre existe a possibilidade de novos ataques e invasões que podem ocasionar em:

- roubo de informações sigilosas como, senhas e números de cartões de crédito ou débito;
- disseminação de vírus;
- acessos não autorizados à internet;
- envio de falsas mensagens de *e-mail*;
- acesso remoto não autorizado para propagação ou coordenação de ataque descentralizado.

Muitas das comunicações realizadas em redes e internet são relacionadas à transações bancárias, comércio eletrônico ou a quaisquer operações financeiras e comerciais. Com esse grande crescimento do uso de computadores e redes, também cresce a necessidade de se aumentar a segurança computacional, porém, isso não deve ser feito apenas em ambiente local e doméstico. Grandes empresas e instituições necessitam de ferramentas e infraestrutura mais robustas para lidar com possíveis ameaças de segurança e disseminação em massa de mensagens de *spam*.

2.1 *Phishing*

O *Phishing* pode ser facilmente confundido com *spam* e está cada vez mais frequente. Trata-se da "arte de enganar" [Milani, 2016]. Segundo Vechia [2014], essas são técnicas de persuasão utilizadas para convencer a pessoa a fornecer informações, geralmente sensíveis, como dados bancários, senhas, endereços, entre outros. Antigamente esse método de golpe

era realizado utilizando-se o telefone, porém, com a era digital é muito mais fácil e prático enviar milhares de mensagens eletrônicas para vários destinatários em pouco tempo.

Também são conhecidos como *phishing* os golpes envolvendo a tentativa de roubos financeiros. Por exemplo, um e-mail pode ser enviado com um formulário a várias pessoas, alegando ser de um banco e que é necessário uma atualização dos dados cadastrais, entre eles a senha (Figura 1). Uma vez preenchido o formulário e enviado, o criminoso pode utilizar esses dados como quiser, para fazer movimentações bancárias por exemplo [Milani, 2016].

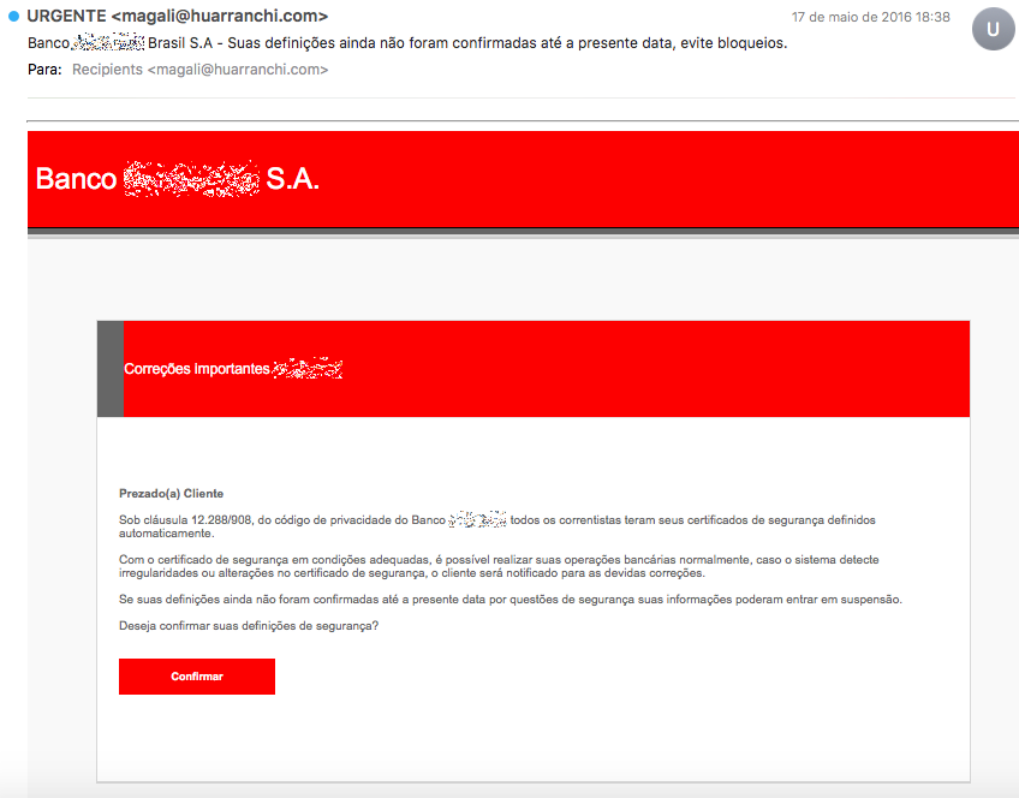

Figura 1: Captura de tela de um *e-mail* com uma mensagem de *phishing*, com a intenção de enganar o usuário para conseguir informações bancária.
Autoria Própria (17 de Maio de 2016)

2.2 Mensagens de *Spam*

Uma mensagem de *spam* pode ser de vários tipos, por exemplo, um *e-mail* falso contendo informações genéricas, com a intenção de enganar o leitor a ponto de extrair informações, ou propagandas não autorizadas. Uma definição mais formal diz que *spam* é o termo

usado para se referir aos *e-mails* não solicitados, que geralmente são enviados para um grande número de pessoas quando este tipo de mensagem possui conteúdo exclusivamente comercial. Também é referenciado como UCE (*Unsolicited Commercial E-mail*) [Vechia, 2014].

Uma grande maioria dos ataques computacionais, roubo de informação e invasões de sistemas e redes se originam de mensagens de *spam* não detectadas ou de pouco conhecimento por parte dos usuários para sua detecção manual. Milani [2011]

A técnica de mensagens de *spam* também se utiliza de engenharia social para enganar o usuário, podendo agir de diversas formas diferentes quando executada, conforme mencionado em [Cert.br, 2016], ocasionando:

- **Perda de mensagens importantes:** devido ao grande número de mensagens de *spam* recebida, ocorre o risco do usuário não ler mensagens importantes, ler com atraso ou apagá-las por engano.

- **Conteúdo impróprio ou ofensivo:** como grande parte das mensagens de *spam* são enviadas para conjuntos aleatórios de endereços de *e-mail*, é bastante comum que o usuário receba mensagens cujo conteúdo considere impróprio ou ofensivo.

- **Gasto desnecessário de tempo:** para cada mensagem de *spam* recebida, é necessário que o usuário gaste um tempo para ler, identificar e remover da sua caixa postal, o que pode resultar em gasto desnecessário de tempo e em perda de produtividade.

- **Não recebimento de *e-mails*:** caso o número de mensagens de *spam* recebidas seja grande e o usuário utilize um serviço de *e-mail* com limite de tamanho de caixa postal, corre-se o risco de lotar a caixa de *e-mail* e, até que se consiga liberar o espaço, ficará impedido de receber novas mensagens.

- **Classificação errada de mensagens:** caso sejam utilizados sistemas de filtragem com regras *anti-spam* ineficientes, há o risco de mensagens legítimas serem classificadas como *spam* e que, de acordo com as suas configurações, podem ser apagadas, movidas para quarentena ou redirecionadas para outras pastas de *e-mail*.

2.3 Problemas relacionado a *spam*

Independente do tipo de acesso à internet usado, é o endereço de destino do *spam* quem paga pelo envio da mensagem. Os provedores, para tentar minimizar os problemas de *spam*, disponibilizam mais recursos computacionais e os custos derivados acabam sendo transferidos e incorporados ao valor mensal que os usuários pagam.

Alguns dos problemas relacionados a *spam* que provedores e empresas costumam enfrentar, segundo a [Cert.br, 2016] são:

- **Impacto na banda:** o volume de tráfego gerado pelos *spams* faz com que seja necessário aumentar a capacidade dos *links* de conexão com a internet.
- **Má utilização dos servidores:** boa parte dos recursos dos servidores de *e-mail* como, tempo de processamento e espaço em disco, são consumidos no tratamento de mensagens não solicitadas.
- **Inclusão em listas de bloqueio:** um provedor que tenha usuários envolvidos em casos de envio de *spam*, pode ter a rede incluída em listas de bloqueio (*blacklist*), o que pode prejudicar o envio de *e-mails* por parte dos demais usuários e resultar em perda de clientes.
- **Investimento extra em recursos:** os problemas gerados pelos *spams* fazem com que seja necessário aumentar os investimentos para a aquisição de equipamentos e sistemas de filtragem e para a contratação de mais técnicos especializados na sua operação.

2.4 *Spammer*

Os *spammers* utilizam diversas técnicas para coletar endereços de *e-mail*, desde a compra de bancos de dados até a produção de suas próprias listas, geradas a partir de:

- **Ataques de dicionário:** consistem em formar endereços de *e-mail* a partir de listas de nomes de pessoas, de palavras presentes em dicionários e/ou da combinação de caracteres alfanuméricos.

- **Códigos maliciosos:** muitos códigos maliciosos, são projetados para varrer o computador infectado em busca de endereços de *e-mail* que, posteriormente, são repassados para os *spammers*.

- ***Harvesting*:** consiste em coletar endereços de *e-mail* por meio de varreduras em páginas *Web* e arquivos de listas de discussão, entre outros. Para tentar combater esta técnica, muitas páginas *Web* e listas de discussão apresentam os endereços de forma ofuscada (por exemplo, substituindo o "@" por "(at)" e os pontos pela palavra "dot"). Infelizmente, tais substituições são previstas por vários dos programas que implementam esta técnica.

Após efetuarem a coleta, os *Spammer* procuram confirmar a existência dos endereços de *e-mail* e, para isto, costumam se utilizar de artifícios, como:

- enviar mensagens para os endereços coletados e, com base nas respostas recebidas dos servidores de *e-mail*, identificar quais endereços são válidos e quais não são;

- incluir no *spam* um suposto mecanismo para a remoção da lista de *e-mails*, como um *link* ou um endereço de *e-mail* (quando o usuário solicita a remoção, na verdade está confirmando para o *Spammer* que aquele endereço de *e-mail* é válido e realmente utilizado);

- incluir no *spam* uma imagem do tipo *Web bug*, projetada para monitorar o acesso a uma página *Web* ou *e-mail* (quando o usuário abre o *spam*, o *Web bug* é acessado e o *Spammer* recebe a confirmação que aquele endereço de *e-mail* é válido).

Com todas essas características, variações e técnicas envolvendo a disseminação de mensagens de *spam*, torna-se cada vez mais difícil que sistemas e ferramentas de detecção tradicionais consigam identificar e bloquear essas ameaças. Por isso, novas ferramentas surgem utilizando o aprendizado de máquina e inteligência artificial, para acompanhar a evolução dessas ameaças.

3 Mineração de Texto

A mineração de texto, consiste em extrair as características mais relevantes em uma quantidade de texto não estruturado. Segundo Weiss [2010], as técnicas utilizadas na mineração de textos são semelhantes às utilizadas na mineração de dados, ou seja, fazem o uso dos mesmos métodos de aprendizagem, independente se uma técnica utiliza-se de dados textuais e a outra, de dados numéricos.

As duas técnicas podem ser diferenciadas a partir de dois conceitos, a mineração de dados é caracterizada por extrair informações implícitas, anteriormente desconhecidas, porém potencialmente úteis. Já na mineração de texto os dados extraídos são claros, sendo mais explícitos nos textos, mas as informações não são passíveis de processamento automático [Witten and Frank, 2011], ou seja, precisam, passar por um pré-processamento, onde serão transformadas em uma matriz numérica.

É fato que estamos vivendo um acelerado crescimento de informações não estruturadas (*posts* em redes sociais, *twitters*, fotos, comentários, blogs, foruns de discução, entre outros) e, com isso, a mineração de texto ganha destaque não apenas no meio acadêmico mas também no mundo dos negócios [Brito, 2016].

Este trabalho irá abordar a técnica de mineração de texto, na qual será feita a extração das características mais relevantes das mensagens de *spam* que, posteriormente, foram processadas e classificadas.

3.1 Processamento de Linguagem Natural (PNL)

O Processamento de Linguagem Natural, consite no desenvolvimento de modelos computacionais para realizar tarefas dependentes de informação em linguagem natural.

Segundo Covington et al. [1997], a PNL está focada em três grandes aspectos da comunicação em linguagem natural:

- **som:** fonologia;
- **estrutura:** morfologia e sintaxe;
- **significado:** semântica e pragmática.

A fonologia está relacionado ao reconhecimento dos sons que compões uma palavra. Já a morfologia faz o reconhecimento das palavras através das unidades primitivas que as compõem (ex. caçou, caç + ou). A sintaxe estuda a estrutura da frase, baseada na forma como as palavras se relacionam nessa frase, como mostrado na Figura 2. Por sua vez, a semântica faz a associação de uma estrutura sintática em termos dos significados das palavras que estão compondo essa estrutura. (ex. na estrutura da Figura 2 pode-se associar o significado "*um animal perseguiu / capturou outro animal*"). E, por fim, a pragmática faz a verificação do significado associado a uma estrutura sintática conferindo se ela é realmente mais adequada ao contexto. (ex. usando o contexto predador-presa, "*perseguiu/capturou*" $\rightarrow$ "*comeu*").

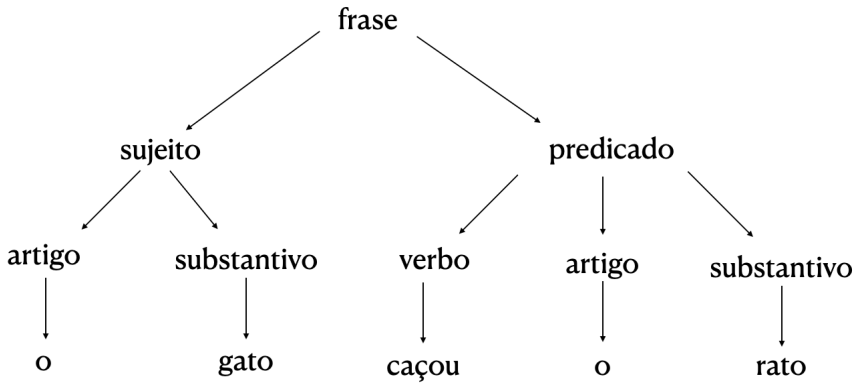

Figura 2: Exemplo de uma árvore sintática
Autoria Própria

A PNL é uma área de estudo fundamental para a mineração de texto e extração de características, uma vez que, será necessário extrair informações de textos em linguagem natural e os transformar em dados, para serem processados de forma mais rápida e facil.

3.2 Extração de Características

Quando utilizamos corpus de texto extraídos de mensagens é natural que se tenha dados de alta dimensionalidade. Por isso, se faz necessário o uso de filtros, que selecionam os termos mais relevantes no conjunto de características [Mitchell, 1997].

A extração de características, engloba simplificar o conjunto de dados requeridos, para descrever um grande conjunto com mais precisão. Também é um termo genérico para os

métodos de construção e combinação de valores, transformando uma entrada de dados grande, em um vetor simplificado, onde seu processamento é mais rápido.

3.3 *Term Frequency* (TF)

O Algoritmo TF (*Term Frequency*), como o próprio nome diz (frequência dos termos), tem a finalidade de atribuir pontuação para cada termo, de acordo com a frequência que esse termo aparece. As principais aplicações são voltadas para classificação, clusterização e regressão. Cada documento é uma amostra e, cada palavra, um atributo. O resultado será uma matriz de termos-frequências, com dimensões: (quantidade de documentos, quantidade de palavras), (2 X 13) como mostrado na Figura 3.

Documento 01: "Não trabalho para ter clientes, tenho clientes para poder trabalhar."

Documento 02: "Não se pode forçar a inteligência a trabalhar."

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

Figura 3: Exemplo de uma Matriz Termos-Frequências
Autoria Própria

Cada documento pode estar associado a um y, que pode ser categórico ou contínuo ("livro XYZ", "2,57", "português", "autor ABC", etc), assim como os conjuntos de dados que serão utilizados neste projeto.

É dificil fazer uma comparação utilizando documentos com tamanhos diferentes, como por exemplo: Texto 1 tem dois parágrafos, enquanto Texto 2 tem aproximadamente 400 páginas. Para isso é necessário fazer a normalização do vetor, onde será feita a divisão de cada um dos seus elementos pelo comprimento do mesmo e a frequência relativa.

TF:

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

TF normalizado:

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
0,27	0,27	0,53	0,27	0,53	0,27	0,27	0,27	0	0	0	0	0
0,32	0	0	0	0	0	0	0,32	0,32	0,32	0,32	0,63	0,32

Figura 4: Exemplo de uma Matriz Termos-Frequências Normalizada
Autoria Própria

A matriz de Termos-Frequências dá igual peso a todas as palavras, ou seja, as palavras como "o", "a", "de", "para", etc, tem o mesmo valor de "trabalho", "inteligência", "criatividade", etc. Dependendo da finalidade esperada, as palavras mais comuns (trabalho, inteligência) deveriam ter um peso maior e esse problema acaba atrapalhando. Por isso é necessário fazer a remoção das chamadas *stopwords* (artigos, pronomes e preposições). É uma solução comum e vários pacotes de mineração de texto já vêm com listas de *stopwords*.

3.4 *Term Frequency - Inverse Document Frequency* (TF-IDF)

Assim como o *Term Frequency* (TF), o TF-IDF também é utilizado para a mineração de texto e, sua principal utilidade é a descoberta de palavras importantes em um texto não estruturado ou semi-estruturado [Lima and Castro, 2012]. Ele faz isso atribuindo um peso a cada termo, considerando sua frequência no texto, e em todos os documentos do conjunto de dados indicando sua importância. Na fórmula abaixo o TF, representado por TF(i,j) é o número de vezes que o termo i aparece no documento j. Já o IDF, representado por IDF (i,k), é o logaritmo do total de documentos (k) dividido pelo número de documentos que contém o termo i.

$$TF - IDF_{(i,k,j)} = TF_{(i,j)} \times IDF_{(i,k)} \tag{1}$$

Esta técnica tem por objetivo filtrar todo o texto e classificar as palavras com valores que representam um grau de afinidade com o texto [Ramos, 2003]. Termos que são únicos ou pouco vistos em um ou em grupos pequenos de documentos tendem a ter um peso mais elevado no TF-IDF.

No IDF(i,k), se o número de documentos que contem o termo i for igual ou próximo ao número de documentos, seu valor será próximo de 1, e o resultado do TF-IDF será um número pequeno, que, representa um termo comum com uma pequena importância Ramos [2003]. Um bom exemplo para este caso seria o termo "Jesus" em uma análise do Novo Testamento [Choi et al., 2013], ou seja, este é um dos termos mais relevantes na Bíblia.

Documento 01: "Não trabalho para ter clientes, tenho clientes para poder trabalhar."
Documento 02: "Não se pode forçar a inteligência a trabalhar."

TF:

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

TF normalizado:

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
0,27	0,27	0,53	0,27	0,53	0,27	0,27	0,27	0	0	0	0	0
0,32	0	0	0	0	0	0	0,32	0,32	0,32	0,32	0,63	0,32

Figura 5: Comparação da pontuação de pesos do TF com o TF-IDF
Autoria Própria

Como podemos observar na Figura 5, o TF-IDF determinou um maior peso para os termos mais relevantes nos documentos apresentados.

4 Aprendizado de Máquina

A utilização de máquinas inteligentes para realizar diversas tarefas cotidianas ainda é algo muito recente. No entanto, nos últimos tempos isso vem se tornando mais comum, com o aprendizado de máquina na área de inteligência artificial, as máquinas vêm conseguindo executar várias funções que, até então, apenas humanos eram capazes. A ideia inicial do aprendizado de máquina é a criação de mecanismos que, sejam capazes de imitar, não apenas a mente humana [de Fernandes Teixeira, 2014], mas também a de diversos outros seres vivos em determinadas tarefas. Essa técnica está presente em diversas áreas de estudo. Na maioria das aplicações, o foco principal é o reconhecimento de padrões, onde os seres humanos são muito bons. Uma das vantagens na utilização de métodos computacionais para o reconhecimento de padrões está na precisão que pode ser obtida com essa tarefa, além da capacidade de processar uma grande quantidade de informação em um curto período de tempo.

O termo "aprendizado de máquina" define-se como, o desenvolvimento de algoritmos que podem aprender e se adaptar. Diferentemente dos programas convencionais, isto traz a capacidade de que o algoritmo melhore seu desempenho através do aprendizado de experiências anteriores [Blum, 2007]. Com isso, e com evolução dos métodos de aprendizado de máquina, os dados disponibilizados na internet para testes aumentaram consideravelmente [Dumais et al., 1998].

4.1 Reconhecimento de Padrões

Quando falamos em reconhecimento de padrões o ser humano é muito bom. É possível ver a complexidade de um algoritmo de reconhecimento de padrões quando pensamos na facilidade em que o ser humano reconhece um rosto, uma voz, um cheiro, entende palavras escritas ou ouvidas, reconhece objetos pelo tato, e muito mais. O reconhecimento de padrões tem sido fundamental para a sobrevivência do ser humano.

A capacidade de uma máquina reconhecer padrões como os seres humanos já está acontecendo. No decorrer da evolução da tecnologia e do aprendizado de máquina, elas já conseguiram grandes avanços, tais como: o reconhecimento de impressões digitais,

caracteres impressos, reconhecimento de imagens de satélite, reconhecimento facial, de voz, entre muitos outros. Entretanto, a maioria dessas aplicações é bem específica e com pouca ou nenhuma capacidade de generalização [Gonzalez and Woods, 2000].

Na maioria dos casos, a maneira como as aplicações resolvem o problema de reconhecimento de padrões é influenciada diretamente pelo conhecimento de como estes problemas são resolvidos na natureza, tanto nos algoritmos empregados, quanto no *hardware* [Breve, 2005].

O objetivo principal do reconhecimento de padrões é a classificação dos objetos de interesse dentro das categorias, que podem ser chamados de "padrões". Por exemplo: em um sistema, para reconhecer algarismos numéricos arábicos escritos à mão, teríamos dez classes, uma para cada algarismo (de 0 a 9), e cada número seria um padrão a ser classificado como pertencente a uma das dez classes [Breve, 2005].

Para que possa ser feita a classificação dos objetos, é necessário encontrar uma ou mais características para diferenciá-los. Por exemplo: em um sistema de reconhecimento facial, podemos utilizar várias características, tais como: formato do rosto, cor dos olhos, distâncias entre os olhos, formato da boca, simetria do queixo, entre muitas outras. Estas características são chamadas de atributos. Ao escolher os atributos é preciso ter o cuidado de selecionar as principais características que mais se diferenciam entre os objetos de uma classe a outra. Após isso, é preciso decidir qual modelo de classificação (ou classificador) será utilizado.

Após o treinamento, pode ser testada a êficiencia do algoritmo para classificar padrões ainda desconhecidos por ele. Assim, pode-se tomar um conjunto de padrões já rotulados e apresentá-los ao algoritmo sem os rótulos, para que ele mesmo possa rotula-los. Então, comparando os rótulos que já existiam com os exibidos pelo algoritmo é possivel estimar qual será o desempenho desse classificador, ou seja, qual a taxa de acertos e erros que se espera dele.

4.2 Aprendizado Semi-Supervisionado

O aprendizado semi-supervisionado reune características do aprendizado supervisionado e do não-supervisionado, utilizando das vantagens de ambas as categorias, uma vez que o processo de rotular dados frequentemente é caro, demorado e requer o trabalho de especialistas humanos, em contrapartida comumente temos uma abundância de dados não rotulados. Por isso essa abordagem híbrida se torna tão eficaz em determinadas aplicações.

O aprendizado supervisionado requer entradas de dados rotuladas, com o objetivo de ensinar ao algoritmo o padrão de cada classe. Dessa forma, se espera que, quando um dado é fornecido sem nenhum rótulo ou marcação atrelado, o método utilize o conhecimento adquirido durante o treinamento para identificar e classificar a entrada em sua categoria. Esse treinamento pode ser feito com vários tipos de dados, como imagens, textos, vídeos e outros. Com isso, identificar, reconhecer e separar uma grande quantidade de dados se torna mais viável. Como os métodos supervisionados necessitam de dados rotulados por especialistas, acaba sendo muito demorado e custoso obter esses dados rotulados, tendo então como alternativa os métodos não-supervisionados [Zhu, 2011].

Diferente da abordagem supervisionada, a abordagem não-supervisionada não recebe dados rotulados, portanto, os métodos desta categoria tentam encontrar padrões nos atributos dos dados de entrada. Uma das tarefas mais comuns no aprendizado não supervsionado é o agrupamento de dados (clusterização). Neste caso, assume-se que dados que pertençam a um mesmo grupo possuem atributos semelhantes enquanto que dados que pertencem a diferentes grupos possuem atributos diferentes. [Murty et al., 1999] Porém, esta hipótese nem sempre é verdadeira. [Carneloz, 2019].

Os métodos de aprendizado semi-supervisionado fazem o uso tanto dos dados não-rotulados quanto dos dados rotulados [Zhu, 2011], ou seja, contém características do aprendizado supervisionado e do não-supervisionado [Chapelle et al., 2009]

As principais vantagens dessa abordagem são a redução da necessidade de dados rotulados e o melhor aproveitamento da quantidade de dados disponíveis. Em consequencia obtem-se a economia de tempo e recursos. A Figura 6 ilustra um exemplo de aplicação

de aprendizado semi-supervisionado.

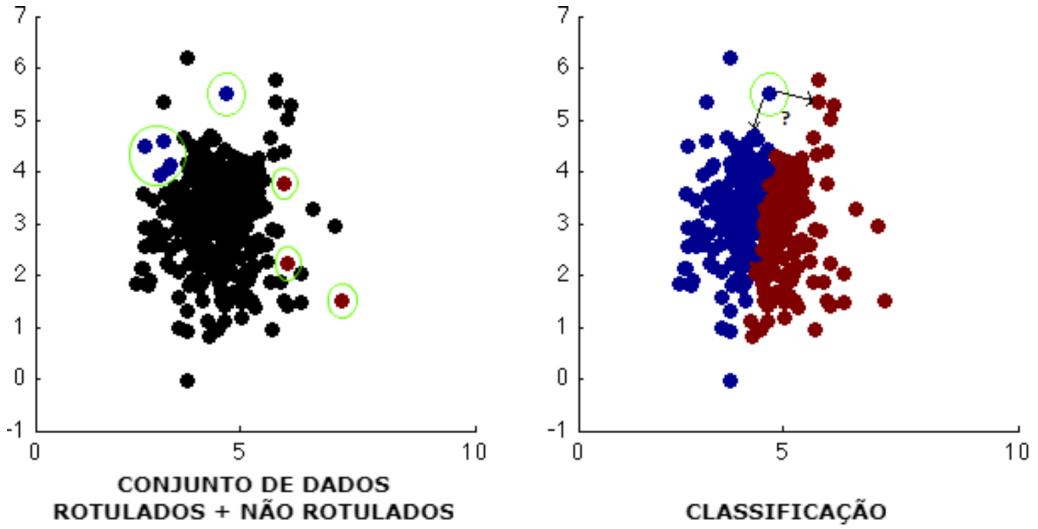

Figura 6: Exemplo de conjunto de dados rotulados e não-rotulados e sua relevância na classificação
Adaptado de: matlab-cookbook.com/recipes/0100_Statistics/kmeans_groups_uni.png

4.2.1 Métodos Baseados em Grafos

Os métodos baseados em grafos são baseados na estrutura analítica de um grafo, a fim de realizarem a classificação dos dados. Suas características semi-supervisionadas são apresentadas a seguir.

Os grafos utilizados nesses métodos representam instâncias que podem ser rotuladas ou não-rotuladas [Carneloz, 2019]. Cada instância é representada por um vértice, e as arestas do grafo representam a similaridade entre eles. A similaridade entre os nós do grafo é medida pelo peso dado a cada aresta que interliga as instâncias distintas. Esse peso é feito a cada iteração, até que seja alcançada a convergencia [Breve, 2010]. Também é possível definir a similaridade simplesmente com as arestas (sem pesos) apenas entre os n vizinhos mais próximos de cada nó. A maioria dos métodos funcionam iterativamente e, desta maneira, a principal função dos nós já rotulados é propagar a informação de rótulo para os demais nós.

Mesmo que os métodos baseados em grafo sejam uma das áreas mais estudadas em aprendizado semi-supervisionado [Chapelle et al., 2009], a maioria dos algoritimos ainda tem a complexidade computacional muito alta ($O(n^3)$), tornando inviável o uso de bases

de dados muito grandes [Carneloz, 2019]. O algoritimo de competição e cooperação entre partículas, que é um dos algoritmos utilizado nessa pesquisa e será explicado posteriormente, se destaca exatamente por conseguir manter uma complexidade linear de (O(n)) em seu laço principal [Breve et al., 2012].

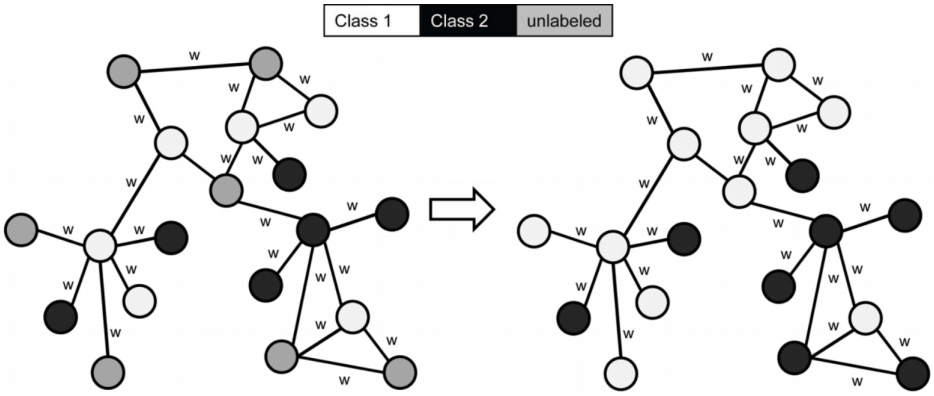

Figura 7: Demonstração da convergência de um método semi-supervisionado, baseado em grafos. A letra w representa a existência de um peso na aresta.
Park et al. [2014]

Através da Figura 7, podemos observar os modelos baseados em grafo. Inicialmente se tem o conjunto de três tipos de dados diferentes, sendo rotulados pertencentes à classe branca (*class* 1), rotulados pertencente à classe preta (*class*2) e não rotulados. O grafo da esquerda representa o estado inicial e o da direita oa convergência, sendo assim, o da direita demonstra a completa rotulagem dos dados que não possuíam classe [Park et al., 2014].

4.2.2 *Linear Neighborhood Propagation* (LNP)

O modelo *Linear Neighborhood Propagation* (LNP) [Wang and Zhang, 2008] é uma técnica de aprendizado de máquina semi-supervisionado baseado em grafos. Esse modelo, ao contrário dos outros, não constroi o grafo por inteiro, ou seja, possui arestas conectadas a todos os pontos. A idéia do LNP é que, cada um dos n elementos do conjunto de dados, podem ser reescritos como uma combinação linear de seus k vizinhos mais próximos [Romoni et al., 2013]. Assim, cada nó x_i possui arestas que ligam ele aos k nós mais

próximos, onde o peso dessas arestas w_{ij} correspondem a valores reais positivos.

$$min_{wij} \sum_{j,k:x_j x_k \epsilon N(x_i)} \omega_{ij} G_i^{jk} \omega_{ik} \tag{2}$$

Dado que $\Sigma_i w_{ij} = 1$ e $w_{ij} \geq 0$, onde G_{jk}^i é o elemento $G^i(j,k) = (x_i ... x_j)^T (x_i ... x_k)$ da matriz *Gram* G_i do nó x_i. Uma vez resolvido os problemas da minimização, é criada a matriz esparsa $W(i,j) = w_{ij}$, que representa as similaridades entre os nós do grafo.

O algoritmo LNP permite retirar ruídos do conjunto de dados em um passo de pré-processamento que remove as arestas. Posteriormente, aproxima todo o grafo com uma série de sobreposições da vizinhança, e os pesos das arestas em cada caminho [Negretto, 2016]. Após isso, todos os pesos das arestas são agregados de modo a formar a matriz de pesos de todo o grafo.

Essa propagação de rótulos através de vizinhanças lineares pode ser escrita por meio da iteração abaixo, onde F e Y são matrizes de dimensões x x c, sendo n o tamanho total do grafo e c o número de classes. A matriz F_i^m, corresponde aos valores de pertinência do nó x_i para cada classe, na iteração m. A matriz Y contém as pertinências para os dados já rotulados, ou seja, $Y_{ij} = 1$ se o exemplar i pertence à classe j, e 0 caso contrário. Já o parâmetro α, indica a importância dada aos rótulos dos vizinhos, e o complemento $(1 - \alpha)$ é a importância dada aos rótulos dos dados já classificados inicialmente. Após a convergência, a classe y_i de cada nó x_i recebe o valor $y_i = argmax_{1<j<c} F_{ij}$ [Romoni et al., 2013]. Dessa forma, o algoritmo de *Linear Neighborhood Propagation* tem uma complexidade computacional de O(kn) [Negretto, 2016].

$$F^{m+1} = \alpha W F^m + (1 - \alpha) Y \tag{3}$$

4.2.3 *Label Propagation* (LABELPROP)

O modelo de *label propagation* foi proposto por [Xiaojin and Zoubin, 2002], com a finalidade de propagar informações de dados rotulados para dados não rotulados, fazendo assim o maior uso possivel de dados não-rotulados, para treinamento de modelos semi-supervisionados [Ribeiro, 2015]. Esse modelo vem mostrando bons resultados em diversas

aplicações e sendo testado constantemente [Araújo, 2016].

A complexidade desse modelo é próxima a linear O(m), (onde m é o número de arestas do grafo) onde o torna um método de execução rápida. Devido aos resultados significativos em diversas aplicações distintas e seu tempo de execução reduzido, ele foi escolhido para compor esse trabalho.

Esse modelo de aprendizado semi-supervisionado, utiliza poucos dados rotulados, também chamados de *sementes*, para categorizar um grande número de dados não-rotulados [Araújo, 2016]. Além dos dados *semente* também é utilizada uma relação entre os dados. Existem dois requisitos para essa relação: devem ser transitivos e conter alguma similaridade entre os dados [Araújo, 2016].

Essa relação entre os dados é representada por um grafo, onde cada nó do grafo representa um dado, e as arestas representam a similaridade entre os nós. Além disso, existe um peso asociado a cada nó. No caso de nós *semente* esse peso é fixo, e para os demais nós esse peso é derivado [Araújo, 2016]. A Figura 8 demonstra o fluxograma do modelo *LABELPROP*.

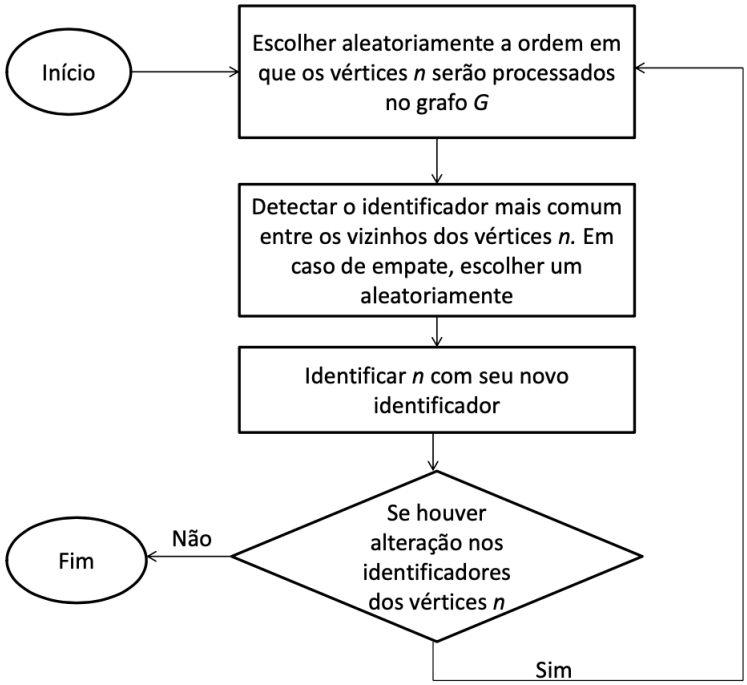

Figura 8: Fluxograma do Algoritmo *label propagation.*
Araújo [2016]

O modelo *labelprop* utiliza de uma abordagem simples em que todos os vértices do grafo G são marcados com uma identificação diferente. Todos os vértices inicialmente possuem uma identificação própria, e a cada iteração é atribuído uma nova identificação, de modo que cada nó possua a identificação de maior frequência entre seus vizinhos, sendo executado de maneira síncrona esse procedimento [Araújo, 2016]. Em caso de empate, é escolhido um identificador aleatório entre os empates. O modelo só para quando o rótulo de cada vértice for uma das etiquetas mais frequentes de sua vizinhança.

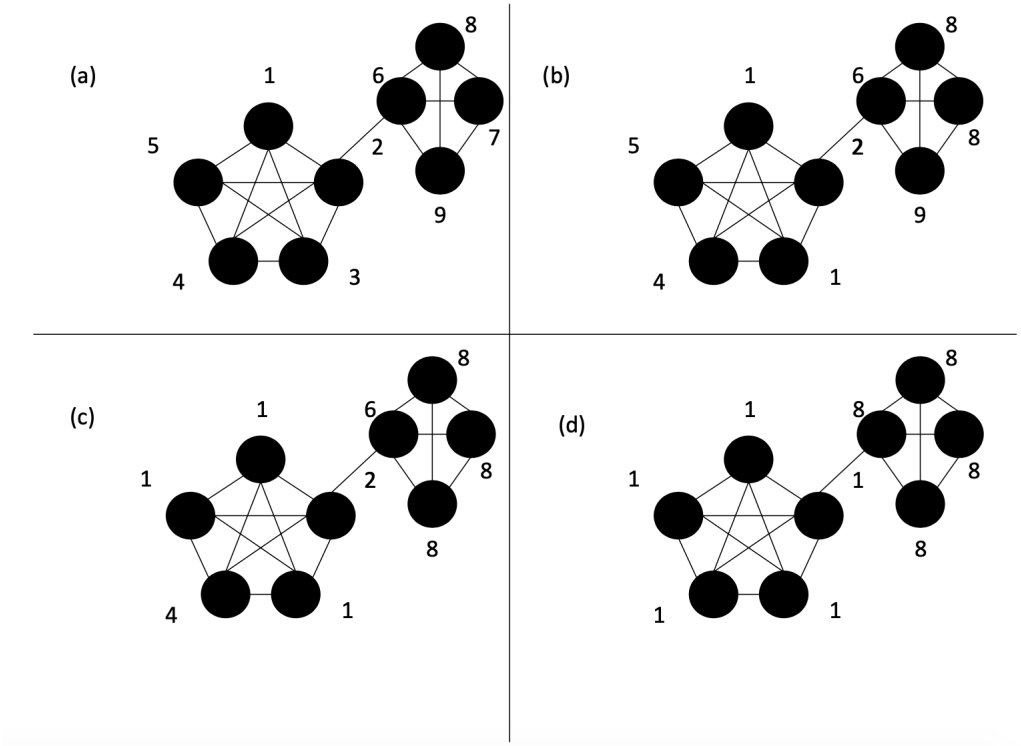

Figura 9: Exemplo de execução do *label propagation*
Adaptado de Raghavan et al. [2007]

A Figura 9 demonstra a execução com um grafo formado por 9 vértices. Na Figura 9(a) o algoritmo passa por sua execução inicial e finaliza na Figura 9(d). O grafo foi dividido em dois grupos com os índices de 1 e 8 ao final devido ao seu número elevado de arestas em comum entre os membros desses dois grupos [Araújo, 2016].

Considere um grafo G=(V,E) onde V são os nós e E são as arestas. O modelo *label propagation* tem o objetivo de minimizar a função quadrática de energia onde y_i e y_j são respectivamente os pesos associadas a i e j[Ribeiro, 2015]. Como mostrado na Equação 4.

$$\varepsilon = \frac{1}{2} \sum_{(i,j)\epsilon E} \omega_{ij}(y_i - y_j)^2 \tag{4}$$

Para derivar os pesos de y_i, é setado $\frac{\varrho\varepsilon}{\varrho y_i} = 0$ a fim de obter a equação de atualização.

$$y^i = \frac{\sum_{i,j\epsilon E} w_{ij} y_j}{\sum_{(i,j)\epsilon E} w_{ij}} \tag{5}$$

Uma matriz estocástica de transição T é derivada da normalização das linhas de W

onde W se trata de uma matriz n x n, onde $n = |V|$ e $W = [w_{ij}]$. A matriz T apresentada na equação 6, pode ser vista como a probabilidade de transição do nó j até o nó i [Ribeiro, 2015].

$$T_{ij} = P(j \rightarrow i) = \frac{w_{ij}}{\sum_{k=1}^{n} w_{kj}} \tag{6}$$

4.2.4 *Learning with Local and Global Consistency*(LLGC)

O modelo *Learning with Local and Global Consistency* (LLGC)[Zhou et al., 2004] é uma técnica de aprendizdo de máquina semi-supervisionado baseado em propagação de rótulos, onde faz a propagação da informação de dados rotulados para dados não rotulados, para efetuar a propagação de rótulos no grafo homogêneo, onde iterativamente os rótulos atribuídos aos vértices do grafo são propagados de maneira à minimizar a Equação 7 [Faleiros, 2016].

$$Q(G) = \frac{1}{2} \sum_{ej,i \epsilon \xi} f_{j,i} \Omega(R_j, R_i) + \mu \sum_{v_i \epsilon \gamma^1} \Omega^1(R_i, Y_i) \tag{7}$$

Para explicar tal função, considere l o número de classes e Y_i um vetor l-dimensional associado a um objeto rotulado $v_i \, \epsilon \, \gamma$. Denota-se Y_i, k como a k-ésima dimensão do vetor Y_i. Suponha que c_k $(0 \leq k \leq l)$ é o rótulo da classe associado a v_i, então Y_i, $k = 1$ e Y_i, $r = 0$ para todo $r \neq k$. Considerando que seja fornecido um pequeno conjunto de documentos rotulados γ l , será descrito a função geral de regularização como onde R_i é um vetor l dimensional associado a cada vértices $v_i \, \epsilon \, \gamma$, esse vetor é o rótulo multidimensional que contém as informações de classes – cada dimensão de R_i corresponde ao grau de filiação do vértice v_i para uma classe. As funções Ω e Ω' são métricas que retornam a similaridade entre objetos representado pelo grafo G.

A função objetivo da Equação 7 se baseia em duas premissas. A primeira delas afirma que os valores das informações de classes entre vértices vizinhos e altamente relacionados devem ser próximos. E a segunda requer que as informações atribuídas durante o processo de classificação devem ser o mais próximo possível da real informação de classe já fornecida. O parâmetro μ controla a influência da segunda premissa, i.e. o quanto os

objetos já rotulados devem manter suas informações de classes [Faleiros, 2016].

O Modelo LLGC é descrito a seguir:

1. A partir da matriz de similaridade, definida por:

$$w_{ij} = \begin{cases} exp(-\frac{dist(x_i,x_j)^2}{2\sigma^2}, sei \neq j;) \\ 0, sei = j \end{cases} \tag{8}$$

2. Constrói-se a matriz: $W = D^{\frac{-1}{2}} W D^{\frac{-1}{2}}$

3. Itera-se $F(t+1) = \alpha W F(t) + (1-\alpha)Y$ até convergência, $\alpha \epsilon (0,1)$.

4. Seja F^* o limite da sequência $\{F(t)\}$. Rotule cada x_i como $f_i = arg\ max_{1 \leq j \leq c} F^*_{ij}$.

No primeiro passo é definido o grafo G formado por todos os n pontos, e é construída a sua matriz de adjacência W através do núcleo Gaussiano. A distância entre dois pontos x_i e x_j, $dist(x_i, x_j)$ é comumente considerada a distância euclidiana, e a métrica é a do cosseno para problemas envolvendo classificação de textos. O parâmetro σ é um parâmetro livre no intervalo [0,1].

No segundo passo constrói-se a matriz $W = D^{1/2}\ WD^{1/2}$, onde $D = diag(d)$ é uma matriz diagonal com entradas $d_i = \sum_i w_i j$, e $W = (w_i j)$ é a matriz de pesos.

No terceiro passo temos a iteração $F(t+1) = \alpha W F(t) + (1\alpha)Y$ até a convergência, onde $\alpha \in (0,1)$.

E no quarto passo se F^* é o limite da sequência $F(t)$, então podemos rotular cada x_i com $f_i = arg\ max_1 jc F^*_i j$.

Conforme Zhou et al. [2004] prova-se que a sequência $F(t)$ converge para $F^* = (I\alpha W)^{1Y}$.

Vale ressaltar que o auto-reforço é evitado já que os elementos da diagonal de W são ajustados para zero no primeiro passo. Além disso, a informação é espalhada simetricamente desde que W seja uma matriz simétrica. Finalmente, o rótulo de cada ponto não rótulado é definido para ser a classe da qual ele recebeu mais informações durante o processo de iterações [Nhassengo, 2019]

4.2.5 Competição e Cooperação entre Partículas (PCC)

O modelo de competição e cooperação entre partículas é um algoritmo semi-supervisionado baseado em grafos, especificamente em busca por território em grafo.

Essa abordagem é uma combinação do modelo de competição de partículas (Quiles et al. [2008]), que adota que as partículas caminham no grafo e competem entre si de modo que cada uma conquiste o maior território possível (Figura 10), e do conceito de cooperação entre partículas, onde as partículas da mesma classe se movimentam de forma cooperativa para buscar o território [Breve et al., 2012].

A partir dos conceitos abordados, podemos observar um novo modelo de classificador semi-supervisionado, o qual se utiliza do estudo de grafos para detecção de comunidades em redes utilizando dados rotulados para inferir sobre dados não-rotulados [Carneloz, 2019].

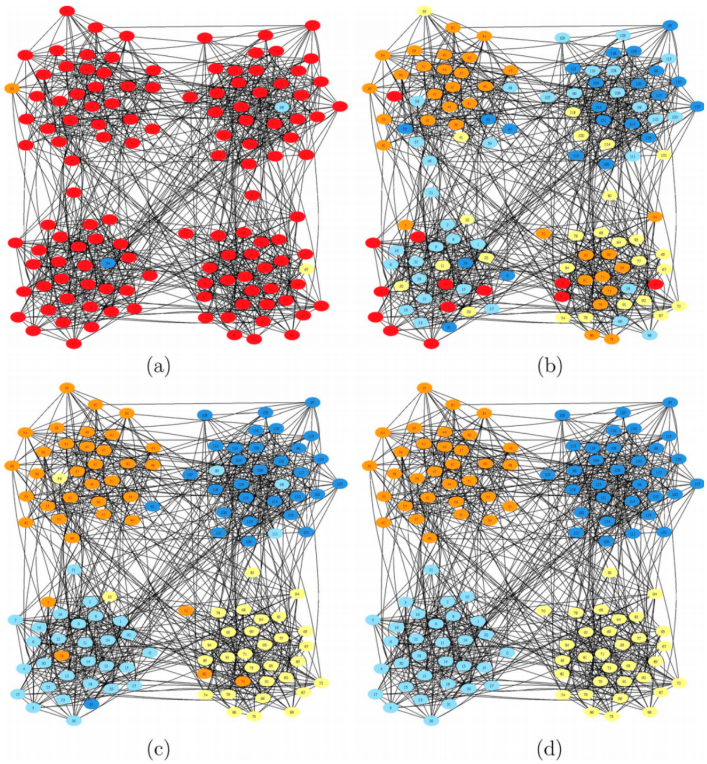

Figura 10: Detecção de 4 comunidades distintas em uma rede com 128 amostras através de competição de partículas. A Figura 10a ilustra a rede em seu estado inicial. As ilustrações seguintes (b), (c) e (d) demonstram o estado da rede após 250, 3500 e 7000 iterações, respectivamente. A cor vermelha simboliza amostras não-rotuladas, enquanto o restante das cores correspondem a cada uma das 4 comunidades.
Quiles et al. [2008]

Como já mencionado anteriormente, a maioria dos modelos baseados em grafos tem sua complexidade muito alta, (O(n^3)), [Zhu, 2005], já o modelo de competição e cooperação entre partículas tem a vantagem sobre os demais modelos por ter sua complexidade linear em (O(n)) no laço principal [Breve et al., 2012]. Isso acontece pelo fato da propagação dos rótulos acontecer localmente, ou seja, apenas um nó é alcançado a cada movimento, bem diferente dos demais modelos em quem a propagação ocorre de modo global [Carneloz, 2019].

De princípio é necessário que o conjunto de dados a ser utilizado seja transformado em uma rede (grafo). A conversão se dá da seguinte forma: Cada amostra do conjunto de dados se transforma em um nó. Dessa forma cada nó pode ser rotulado ou não-rotulado. Posteriormente é criada a relação inicial entre todas as amostras, sendo assim, criada as ligações dos nós entre arestas não direcionadas que não possuem pesos conforme Figura 11.

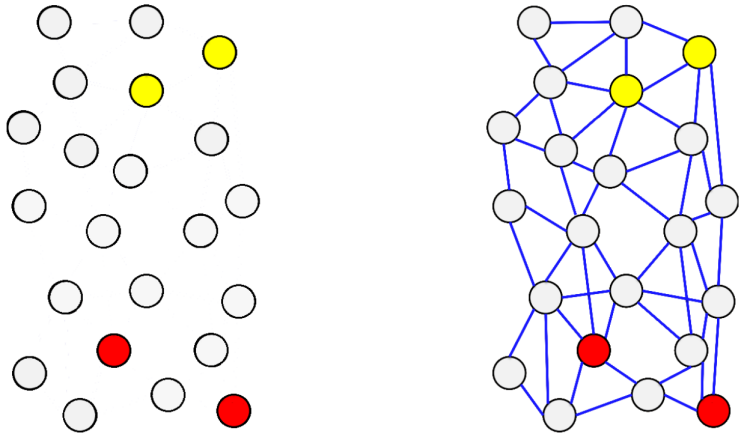

Figura 11: Ilustração do grafo no modelo de partícula. Nós em branco representam as amostras não-rotuladas. Nós amarelos e vermelhos as amostras rotuladas.

Carneloz [2019]

Existem várias formas de fazer a ligação dos nós, uma das formas é a ligação entre os nós com distância menos que um determinado limiar, ou seja, uma aresta E_{ij}, que liga os nós v_i e v_j, somente existirá se a amostra v_i estiver em uma distância menor ou igual a α da amostra v_j. Outra forma de um nó se conectar a k vizinhos mais próximos, assim cada nó terá no mínimo k arestas para k nós diferentes.

Após os nós estarem devidamente conectados entre sí, são criadas as partículas que irão caminhar pela rede, conforme demonstrado na Figura 12. Cada nó do grafo que já está rotulado receberá uma partícula. A partir daí esse nó já rotulado será denomidado *casa* pela partícula ali gerada [Carneloz, 2019]. Os nós *casa* contém propriedades diferentes do restante dos nós.

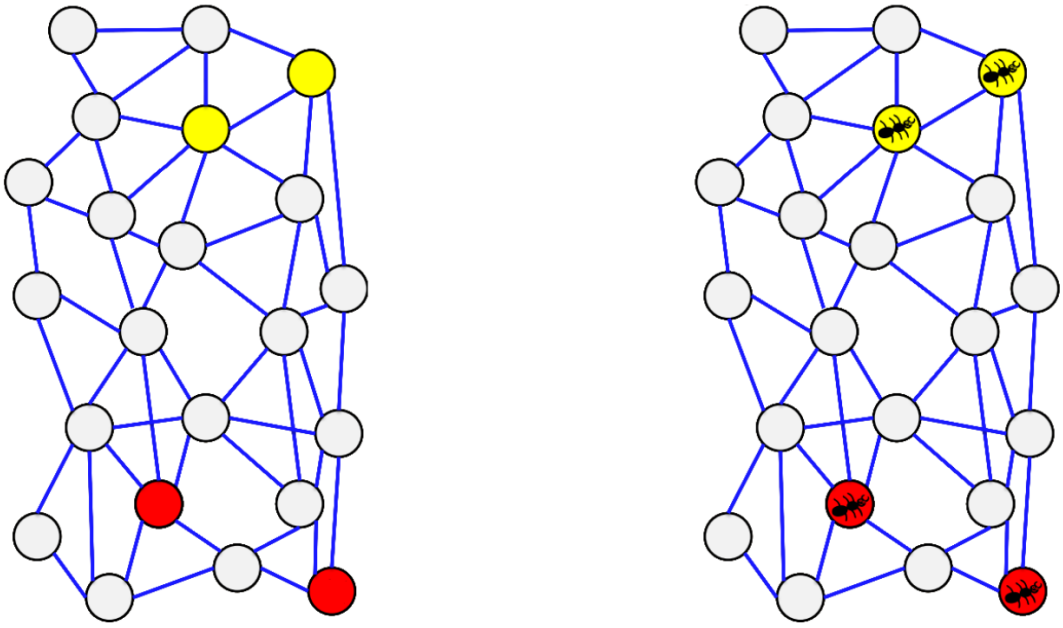

Figura 12: Ilustração do surgimento de partículas no grafo
Carneloz [2019]

A estrutura de domínio dos nós é chamada de "vetores de domínio". Eles indicam o quanto aquele nó está dominado por uma determinada classe. Exemplificando, com quatro classes, um vetor de domínio não-rotulado, no início estaria dominado igualitariamente por todas as classes, sendo assim, cada classe teria 25% do território, como mostrado na Figura 13B. Porém, vetores de domínio rotulados teriam o domínio de 100% pela classe de seu rótulo, como mostrado na Figura 13A. Com os valores iniciais pré-definidos, as partículas irão caminhar pelo grafo com o objetivo de aumetar o domínio de sua classe em todos os nós [Carneloz, 2019]. A cada visita de uma partícula em um nó não-rotulado ela aumenta o nível de domínio do seu rótulo, e diminui o nível de domínio dos demais rótulos.

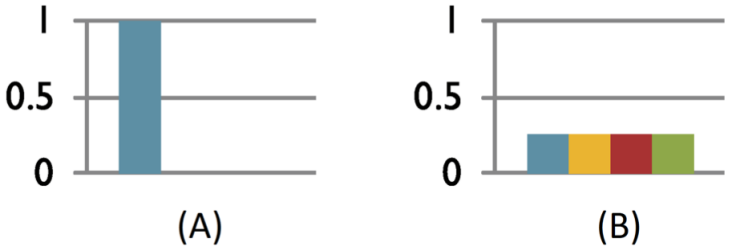

Figura 13: Comparação entre vetores de domínio. Esquerda: vertor de domínio de um nó rotulado. Direita: vetor de domínio de um nó não-rotulado.
Breve et al. [2012]

Segundo [Carneloz, 2019], considere um conjunto de rótulos L onde $y_i \in L$ configura uma amostra rotulada e $y_i = \emptyset$ uma amostra não rotulada. Sendo $y_j^w(t)$ a força de uma partícula p_j com rótulo p_j^f em uma iteração t, cada nó v_i tem seu vetor de domínio $v_i^{w\ell}(t)$ atualizado da seguinte forma:

$$v_i^{wl}(t+1) = \begin{cases} max\left\{0, v_i^{wl}(t) - \frac{\Delta_v \rho_j^w(t)}{c-1}\right\} \\ se\ y_i = \theta\ e\ l \neq p_j^f \\ v_i^{wl}(t) + \sum_{q \neq l} v_i^{wq}(t) - v_i^{wq}(t+1) \\ se\ y_i = \theta\ e\ l = p_j^f \\ v_i^{wl}\ se\ y_i \epsilon L \end{cases} \tag{9}$$

Onde ℓ representa uma determinada classe, e Δ_v é um parâmetro responsável por controlar a taxa de alteração nos níveis de domínio, e c representa a quantidade de classes. Sempre que o vetor de domínio é atualizado, a força da partícula é alterada seguindo a Equação 10.

$$p_j^w(t+1) = v_i^{w\ell}(t+1) \tag{10}$$

Com isso, é possível observar que partículas que caminham em terreno adversário vão perdendo força gradativamente. Com o objetivo de evitar essa situação, o grafo é atualizado automaticamente a cada passo, fazendo assim com que a partícula saiba o quão longe do seu nó *casa* está. Assim cada partícula prioriza proteger seu vizinho mais próximo. A força da partícula de acordo com o território é demonstrada nas Figuras 14

B e 14 C.

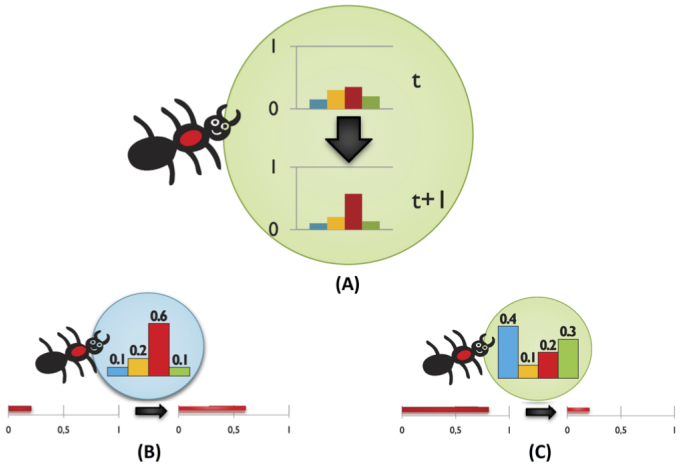

Figura 14: (A) Ilustração de uma partícula aplicando seu domínioa um nó (antes (t) e depois (t+1)).(B) e (C) mostra a força de uma partícula antes e depois de visitar um determinad terreno.

Breve et al. [2012]

As partículas podem se movimentar aleatoriamente, onde poderá escolher o próximo nó sem basear-se nos níveis de domínio, ou as partículas podem se movimentar de maneira gulosa, onde a preferência das partículas é por nós onde já existe um domínio do seu rótulo. As duas formas de movimentação ocorrem, para que assim possam equilibrar seu comportamento de exploração e defesa ao mesmo tempo, conforme mostrado na Figura 15.

No movimento guloso, com p_j^{di} representando o inverso da distância entre o nó v_i e o nó *casa* da partícula (v_j), se tem:

$$p(v_i \mid p_j) = \frac{W_{qi} v_i^{wl} (1 + p_j^{di})^{-2}}{\sum_{\mu=1}^{n} W_{q\mu} v_\mu^{wl} (1 + p_j^{d\mu})^{-2}} \tag{11}$$

Já para o movimento aleatório, sendo q o índice do nó atual da partícula, a probabilidade do nó v_i receber a visita pela partícula p_j é descrito pela equação:

$$p(v_i \mid p_j) = \frac{w_{qi}}{\sum_{\mu-1}^{n} W_{q\mu}} \tag{12}$$

Dessa forma a cada iteração, cada partícula tem a probabilidade P_{grd} de realizar o movimento guloso. Sendo assim, a partícula tem $1 - P_{grd}$ de possibilidade de realizar o

movimento aleatório ($0 \leq P_{grd} \leq 1$).

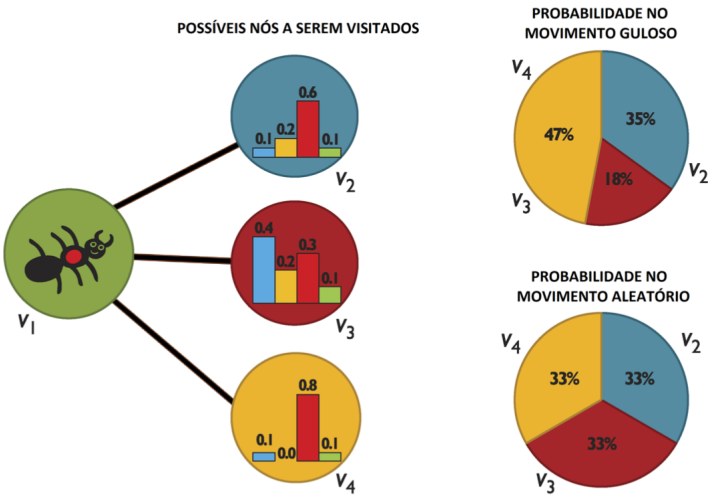

Figura 15: Exemplificação da possobilidade de movimentação para os nós azul, laranja e vermelho quando adotado a estratégia de guloso e aleatório.
Breve et al. [2012]

Desse modo, as partículas caminham pelo grafo até atingir algum critério de parada adotado. Um dos critérios de parada leva em consideração o nível que a rede está tendo de mudanças, e caso seja mínimo, é entendido que as classes já foram definidas. Com isso, os nós ainda não rotulados recebem o rótulo da classe com o maior domínio.

5 Trabalhos Relacionados

Existem diversos trabalhos e pesquisas relacionadas a detecção de *spam* na atualidade. Porém em áreas de atuação distintas, como os trabalhos de Sedhai [2017] (*Semi-Supervised Spam Detection in Twitter Stream.*), que é focado em detecção de *spam* em *twitters* utilizando aprendizado de máquina semi-supervisionado e lista-negra, e o trabalho de Wu et al. [2017] (*Twitter spam detection based on deep learning.*), que propõe uma técnica de aprendizado profundo e o modelo *WordVector*.

Também temos trabalhos que utilizam de API e modelos tradicionais de aprendizado supervisionado para a classificação como *Naive bayes* [Marinho, 2019] (Detecção de *spam* através de aprendizado de máquina.) e *Perceptron* [Tiboli, 2019] (Detecção de *spam* em mensagens de SMS utilizando aprendizado de máquina).

Yafeng and j. Donghong [2017] exploram empiricamente uma rede neural, modelo para aprender a representação em nível de documento para detectar *spam* de opinião enganosa. Primeiro, o modelo aprende a representação de sentenças com rede neural convolucional. Em seguida, as representações de sentenças são combinadas usando uma rede neural recorrente com portas, que pode modelar as informações do discurso e produzir um vetor de documento. Finalmente, as representações de documentos são usadas diretamente como recursos para identificar *spam* de opinião enganosa.

Faris et al. [2018], propôem um sistema de detecção inteligente que é baseado em *Genetic Algorithm* (GA) e *Random Weight Network* (RWN), é proposto para lidar com tarefas de detecção de spam em e-mail. Além disso, uma capacidade de identificação automática também está embutida no sistema proposto para detectar os recursos mais relevantes durante o processo de detecção.

Em *Optimizing Semantic LSTM for Spam Detection*, de Gauri et al. [2019], é implementado um algoritmo com a técnica de aprendizado profundo. Uma arquitetura especial conhecida como *Long Short Term Memory* (LSTM), uma variante da Rede Neural Recursiva (RNN), é usada para classificação de spam. Ele tem a capacidade de aprender recursos abstratos. Antes de usar o LSTM para a tarefa de classificação, o texto é convertido em vetores de palavras semânticos com a ajuda de *word2vec*, *WordNet* e ConceptNet.

6 Metodologia de Desenvolvimento

Esta sessão irá descrever a metodologia adotada, os conjuntos de dados que foram utilizados, as ferramentas e as atividades realizadas durante este projeto.

Como ja mencionado na sessão 1.1, o objetivo, é realizar um estudo comparativo entre extratores de características e classificadores. Com isso, inicialmente foi feito um estudo sobre os extratores de características escolhidos (TF, TF-IDF) e já mencionados na sessão 3.

Cada extrator de característica foi implementado junto a um algoritmo auxiliar, no qual irá fazer a chamada dos arquivos do conjunto de dados e das funções necessárias de cada extrator. Durante o pré processamento do conjunto de dados é feita a retirada das *stopwords* através de uma biblioteca própria do modelo, no caso do TF-IDF a biblioteca *TfidfVectorizer* e no caso do TF a biblioteca *CountVectorizer*. Posteriormente foram extraídas as matrizes resultantes. Após gerar todos os arquivos de cada extrator, foram utilizados quatro algoritmos semi-supervisionados, onde todos são modelos similares e baseados em grafo.

Para uma maior confiança nos resultados, cada algoritmo de classificação foi executado cem vezes variando o valor de k (k vizinhos mais próximos) nos modelos PCC e LNP, e o valor de *sigma* (largura do kernel gaussiano) nos modelos LABELPROP e LLGC em múltiplos de dois até 30 (onde a performace de todos os algoritmos começou a cair), os demais parâmetos de cada algoritmo seguiram as recomendações dos seus autores. Posteriormente, foi feita a média dos valores dos resultados, sendo considerado apenas o melhor resultado dessa combinação, no qual em cada uma das execuções eram selecionados aleatoriamente a quantidade de mensagens de *spam* e não *spam*. Essa metodologia foi repetida com cada um dos 5 conjuntos de dados gerados após suas características extraídas, computando um total de 7.500 execuções com cada algoritmo, possibilitando um comparativo mais confiável entre os resultados.

Foi utilizada a técnica de *hold-out* como critério de avaliação, onde 10% do conjunto de dados é utilizado para treinamento e 90% do conjunto de dados para testes, e posteriormente o inverso, 90% do conjunto de dados para treinamento e 10% para testes, e em

ambos os critérios a porcentagem de dados de treinamento seguiu a metodologia de usar mensagens aleatórias dentro do conjunto de dados onde pelo menos 10% fosse mensagens de *spam*.

A amostragem de 10 para 90 e 90 para 10 por cento foi escolhida devido ao grande crecimento de envio e recebimento de *e-mail's* e mensagens de texto, portanto em um ambiente real, a idéia é ter um algoritmo que seja bom desde o início da sua execução onde teria 90% de um conjunto de dados para treinamento até o ponto que teria 10% ou menos de todo o montante para treinamento.

6.1 Conjunto de Dados (*Dataset*)

Para possibilitar este projeto e também proporcionar uma maior confiança nos resultados foram utilizados conjuntos de dados de *spam* já conhecidas na área de segurança da informação.

Foi utilizado o conjunto de dados *Spambase* como referência, já está com suas características extraídas. Onde para a extração de suas características cada mensagens é dividida em atributos, sendo eles as palavras e caractéres especiais, utilizando os seguintes critérios:

- **palavras:** porcentagem de palavras na mensagem que correspondem ao atributo, ou seja, cem vezes (número de vezes que a palavra aparece na mensagem) dividido pelo número total de palavras da mensagem;

- **cacactéres:** porcentagem de caractéres na mensagem que correspondem ao atributo, ou seja, cem vezes (número de vezes que aparece o caracter na mensagem) dividido pelo número total de caracteres na mensagem.

Ela tem duas categorias, *spam* e *non-spam*, contém 4.601 mensagens e 51 termos distintos, todos em inglês [Mark et al., 1999].

O conjunto de dados *SMS_spam_collection*, que também foi utilizado no projeto, é composto de mensagens de *spam* não tratadas, ou seja, em linguagem natural, no idioma inglês, à qual se torna totalmente compatível para o projeto. Ela tem 5.574 mensagens,

divididas em *spam* e *non-spam*, e não tem sua lista de termos pronta como o conjunto de dados *Spambase* [Tiago, 2012].

Ainda foi utilizado o conjunto de dados *spam_ham_dataset*, na qual contém 5.171 mensagens, também categorizadas como *spam* e *non-spam* nas mesmas características da base *sms_spam_collection*, também em inglês. [Song and Malcolm, 2017].

Os conjuntos de dados *Spambase* e *sms_spam_collection* foram retiradas do repositório *UCI Machine Learning Repository* e são categorizadas para classificação e clusterização, já o conjunto de dados *spam_ham_dataset* foi retirado do repositório *Kaggle*, e também é categorizado para classificação e clusterização tornando-as mais propícias para esse projeto. Todos os três conjuntos de dados foram submetidas a todos os classificadores nos termos descritos anteriormente, proporcionando resultados mais confiáveis como mostrado a seguir.

7 Resultados Obtidos

Nesta seção serão apresentados os resultados obtidos a partir da execução da metodologia já mencionada na seção 5.

Conforme podemos observar nas Tabelas 1 e 2, onde respectivamente são demonstradas a porcentagem da acurácia de cada modelo e seu tempo de execução no qual todos os conjuntos de dados criados com os algoritmos TF e TF-IDF contém 10% de treinamento.

Tabela 1: **10% de Treinamento e 90% de Teste (% Acurácia)**

Algoritmo	**SPAM**	**SMS_TF**	**SMS_TFIDF**	**HAM_TF**	**HAM_TFIDF**
PCC	94,91%	95,32%	95,33%	95,23%	95,05%
LLGC	91,58%	92,29%	91,99%	91,89%	91,69%
LABELPROP	91,05%	91,08%	91,16%	91,03%	91,25%
LNP	83,29%	83,34%	84,47%	85,13%	85,17%
MÉDIA	**90,21%**	**90,51%**	**90,74%**	**90,82%**	**90,79%**

Tabela 2: 10% de Treinamento e 90% de Teste (Tempo em Segundos)

Algoritmo	**SPAM**	**SMS_TF**	**SMS_TFIDF**	**HAM_TF**	**HAM_TFIDF**
PCC	8,440	7,680	7,620	7,140	7,640
LLGC	0,012	0,012	0,017	0,013	0,017
LABELPROP	0,010	0,016	0,009	0,011	0,009
LNP	0,269	0,270	0,273	0,303	0,306
MÉDIA	**2,910**	**2,659**	**2,639**	**2,489**	**2,657**

Podemos observar também que nas Tabelas 3 e 4, onde também são demonstradas respectivamente a porcentagem da acurácia e o tempo de execução de cada modelo no qual todas as bases de dados criadas com os agloritmos TF e TF-IDF com 90% de treinamento.

Tabela 3: **90% de Treinamento e 10% de Teste (% Acurácia)**

Algoritmo	**SPAM**	**SMS_TF**	**SMS_TFIDF**	**HAM_TF**	**HAM_TFIDF**
PCC	97,70%	98,11%	97,58%	97,62%	97,57%
LLGC	96,00%	96,04%	95,74%	96,32%	95,94%
LABELPROP	95,11%	95,45%	95,26%	95,38%	95,40%
LNP	94,58%	95,13%	95,45%	95,42%	94,55%
MÉDIA	**95,85%**	**96,23%**	**96,01%**	**96,19%**	**95,87%**

Tabela 4: **90% de Treinamento e 10% de Teste (Tempo em Segundos)**

Algoritmo	SPAM	SMS_TF	SMS_TFIDF	HAM_TF	HAM_TFIDF
PCC	6,340	8,440	6,160	6,210	6,220
LLGC	0,007	0,021	0,006	0,006	0,010
LABELPROP	0,005	0,004	0,016	0,005	0,007
LNP	0,327	0,288	0,290	0,344	0,340
MÉDIA	**2,226**	**2,918**	**6,472**	**3,188**	**2,192**

Posteriormente serão discutidos detalhadamente os resultados obtidos em cada conjunto de dados utilizado durante o experimento.

7.1 Conjunto de Dados *spambase*

Como já mencionado na sessão 5.1 o conjunto de dados *spambase* é composto por 4.601 mensagens classificadas como *spam* e *nom-spam*, no qual já tinha suas características extraídas.

O modelo de Competição e Cooperação entre Partículas se mostrou ser o mais eficiente tanto com apenas 10% da base para treinamento (Figura 16) como com 90% da base para treinamento (Figura 17).

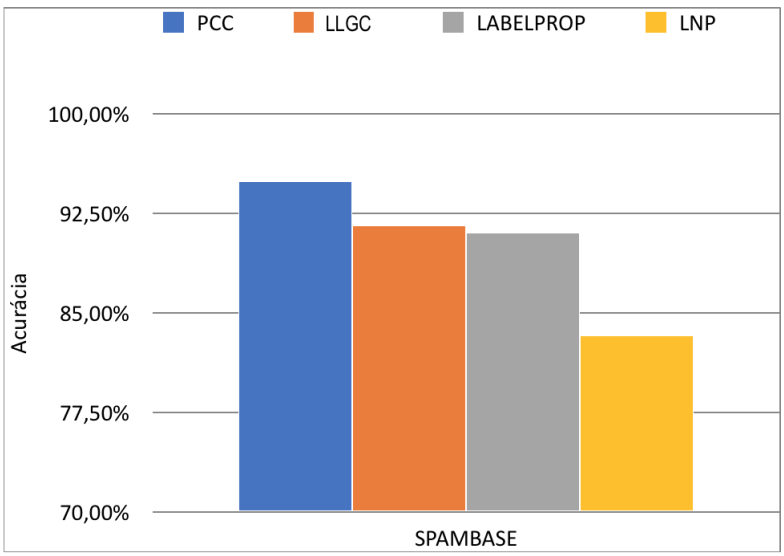

Figura 16: Modelos semi-supervisionados (10% de treinamento) com a base *spambase*

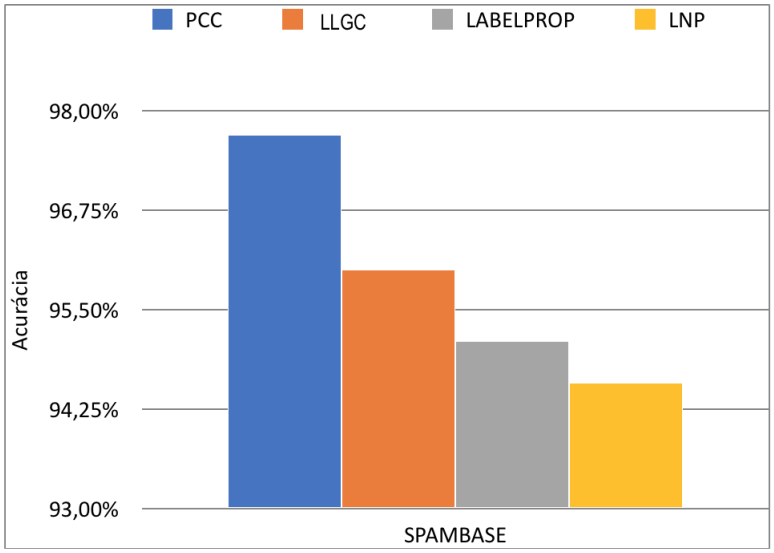

Figura 17: Modelos semi-supervisionados (90% de treinamento) com a base *spambase*

7.2 Conjunto de Dados *smscollection*

O conjunto de dados *smscollectin* foi submetido a extração de características pelos algoritmos TF e TF-IDF, assim gerando dois novos conjuntos de dados nomeados de *smscollection_tf* para as extrações de características com o algoitmo TF, e *smscollection_tfidf* para as extrações de características com o algoritmo TF-IDF foram criadas.

Entre os modelos apresentados, com o conjunto de dados *smscollection_tf* o modelo de competição e cooperação entre partículas se mostrou ser o mais eficiênte atingindo 95,32% de acurácia (Figura 18) e 8 segundos com apenas 10% da base para treinamento e 98,11% de acurácia (Figura 19) e 8 segundos com 90% da base para treinamento.

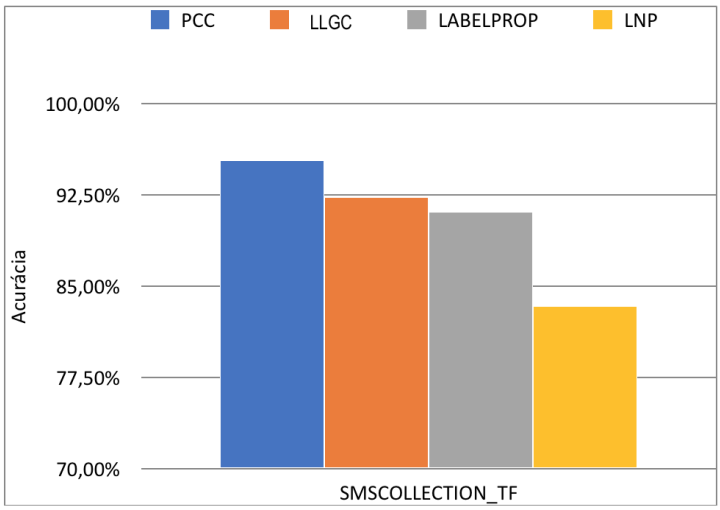

Figura 18: Modelos semi-supervisionados (10% de treinamento) com o conjunto de dados *smscollection_tf*

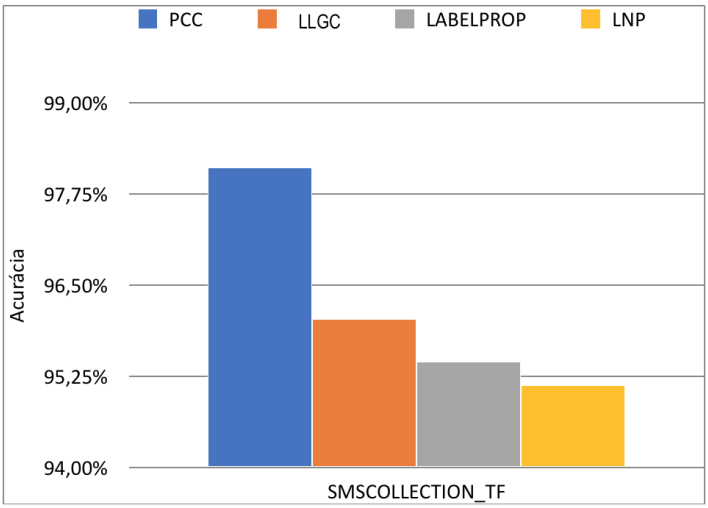

Figura 19: Modelos semi-supervisionados (90% de treinamento) com o conjunto de dados *smscollection_tf*

Com o conjunto de dados *smscollection_tfidf* o modelo que mais se destacou também foi o de competição e cooperação entre partículas obtendo 95,33% de acurácia (Figura 20) em 8 segundos com 10% da base para treinamento, e 97,58% de acurácia (Figura 21) com 90% de treinamento, em 6 segundos.

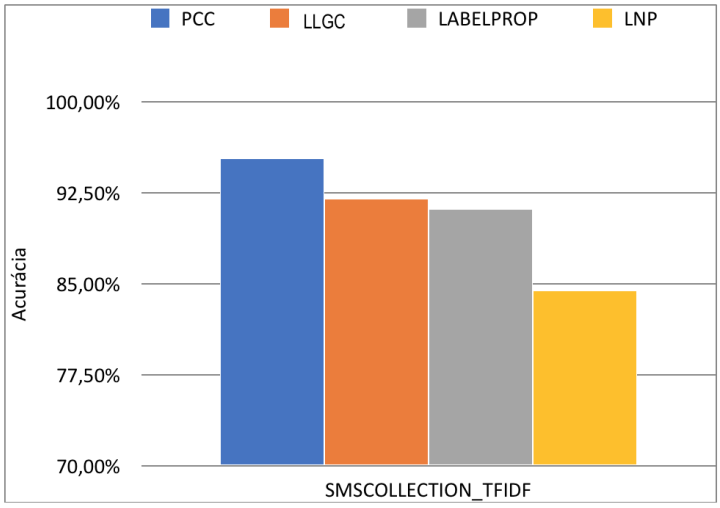

Figura 20: Modelos semi-supervisionados (10% de treinamento) com o conjunto de dados *smscollection_tfidf*

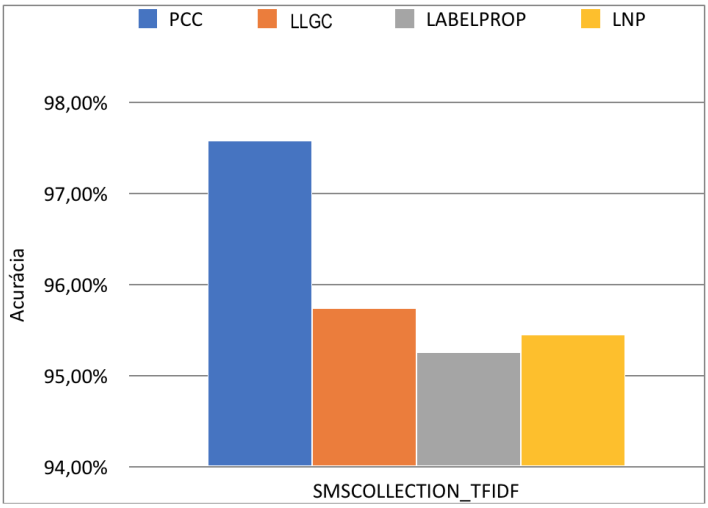

Figura 21: Modelos semi-supervisionados (90% de treinamento) com o conjunto de dados *smscollection_tfidf*

7.3 Conjunto de Dados *spam_ham_dataset*

Assim como o conjunto de dados anteriormente explicado na sessão 6.2 o *spam_ham_dataset* também foi submetido aos algoritmos TF e TF-IDF para a extração de características, e assim gerando duas bases de dados nomeadas de *spam_ham_dataset_tf* para as extrações de características com o algoritmo TF, e *spam_ham_dataset_tfidf* para as extrações de características com o algoritmos TFIDF.

Neste caso também o modelo de competição e cooperação entre partículas, se mostrou ser o mais eficiente, obtendo uma acurácia de 95,23% (Figura 22) em 7 segundos com 10% de treinamento, e uma acurácia de 97,62% 23) em 6 segundos com 90% de treinamento, com o conjunto de dados *spam_ham_dataser_tf*.

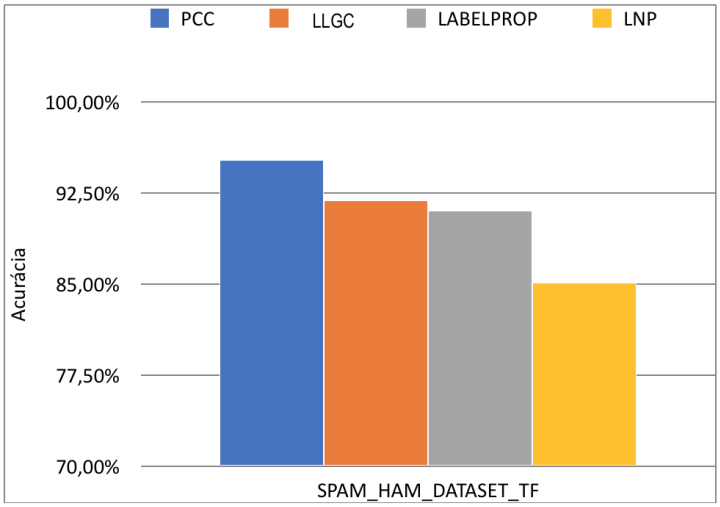

Figura 22: Modelos semi-supervisionados (10% de treinamento) com o conjunto de dados *spam_ham_dataset_tf*

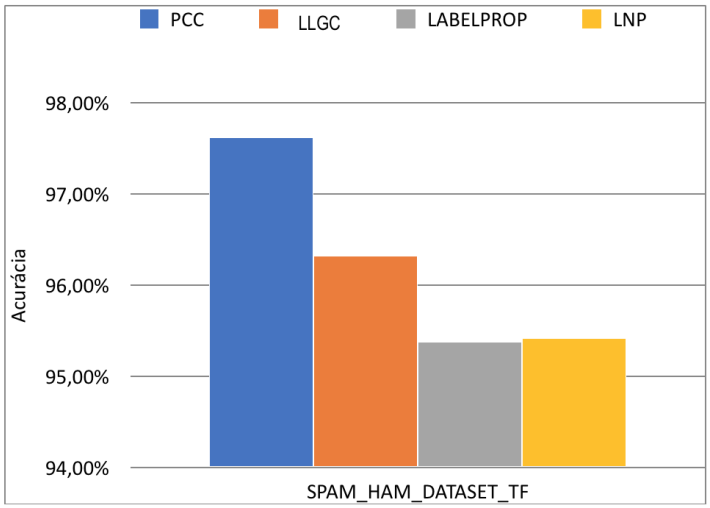

Figura 23: Modelos semi-supervisionados (90% de treinamento) com o conjunto de dados *spam_ham_dataset_tf*

Com o conjunto de dados *spam_ham_dataset_tfidf* o modelo de competição e cooperação entre partículas também se mostrou ser o mais eficiente entre os demais modelos de aprendizado semi-supervisionado, porém com a acurácia de 95,05% (Figura 24) em 7 segundos com 10% de treinamento, e acurácia de 97,57% (Figura 25) em 6 segundos com 90% da base de treinamento.

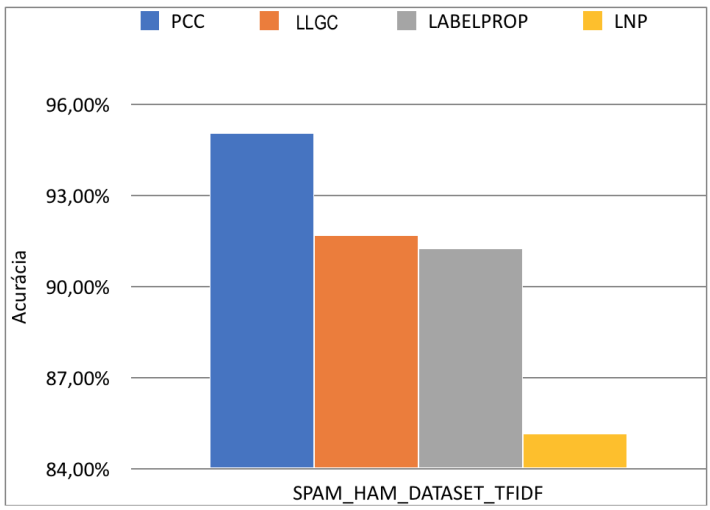

Figura 24: Modelos semi-supervisionados (10% de treinamento) com o conjunto de dados *spam_ham_dataset_tfidf*

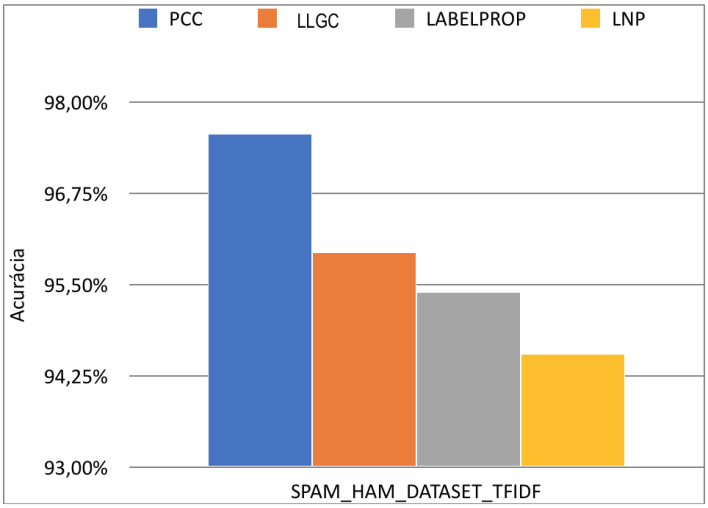

Figura 25: Modelos semi-supervisionados (90% de treinamento) com o conjunto de dados *spam_ham_dataset_tfidf*

Todos os testes do experimento foram realizados em um computador com a seguinte configuração: processador Intel Core i7 2600K 3.4Ghz, memória RAM de 32GB e Sistema Operacional *Windows 10 Professional 64bits*. Os algoritmos de extração de características foram executados em linguagem *Python* na sua versão 3.4 e, os algoritmos de aprendizado semi-supervisionado foram executados em *Matlab* na sua versão R2015a compilação 8.5.0.197613 de 64bits.

8 Considerações Finais e Trabalhos Futuros

Conforme foi possível observar a partir da análise detalhada de cada resultado, podemos constatar que entre todos os modelos o de competição e cooperação entre partículas se destaca em todos os conjunto de dados e nas duas modalidades (10% dos elementos do conjunto de dados de treinamento, e 90% dos elementos do conjunto de dados de treinamento) como mostra a (Figura 26) e (Figura 27).

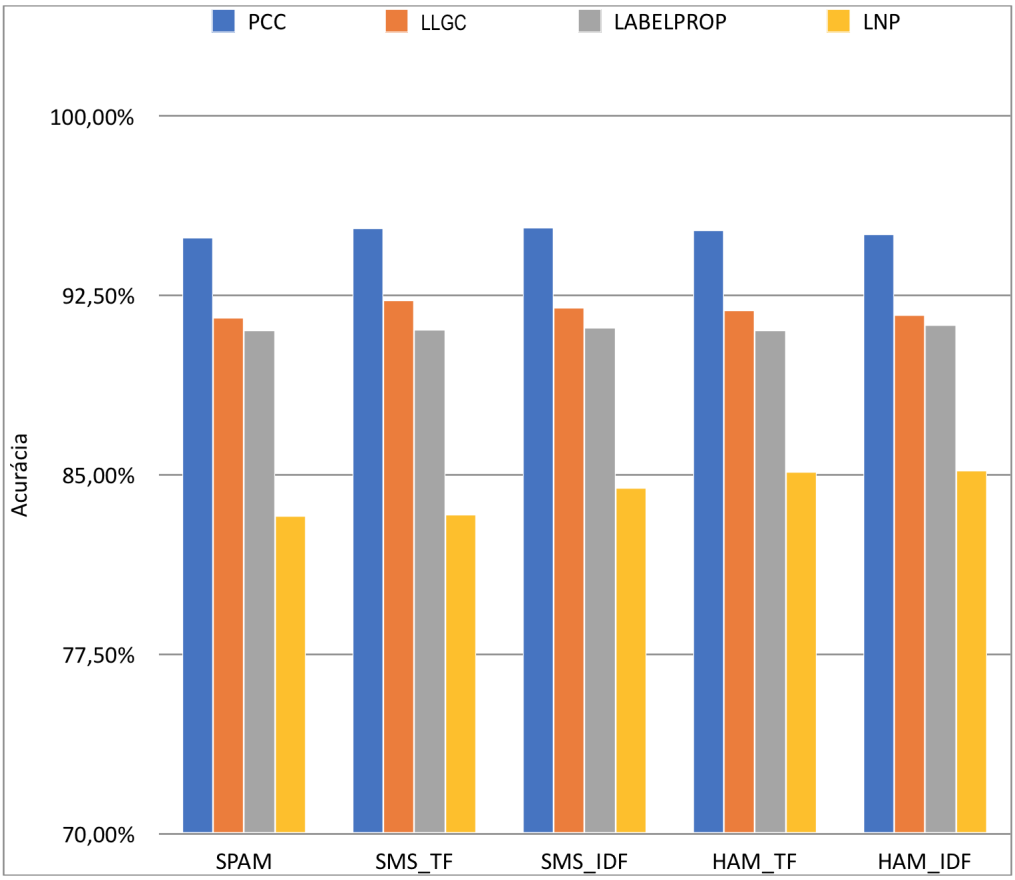

Figura 26: Comparativo dos modelos semi-supervisionados com todos os conjuntos de dados com 10% do conjunto para treinamento

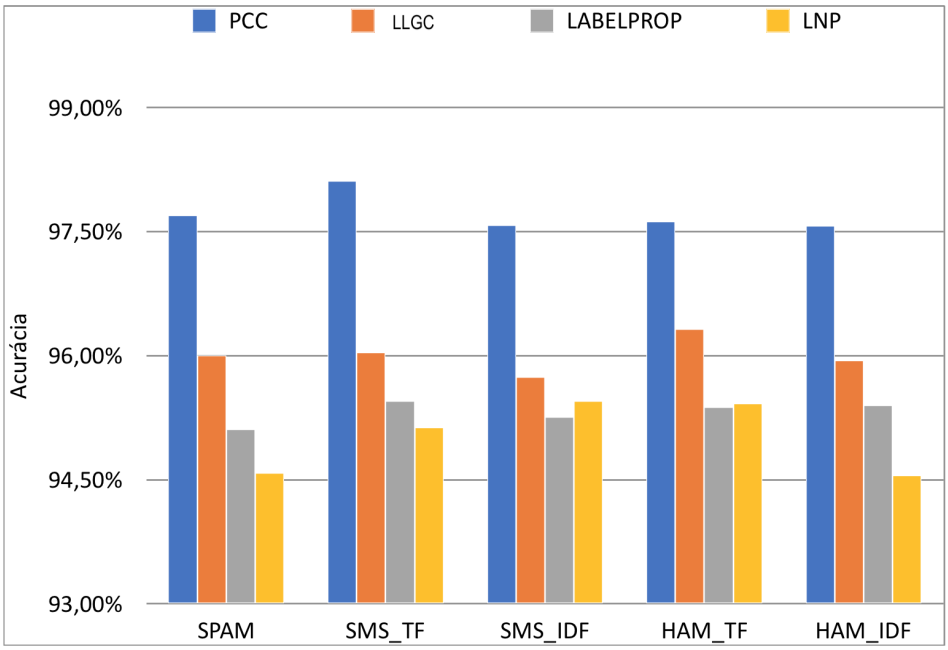

Figura 27: Comparativo dos modelos semi-supervisionados com todos os conjuntos de dados com 90% do conjunto para treinamento

O algoritmo de Competição e Cooperação entre Partículas ainda se mostrou mais eficiente com a técnica de extração de características TF-IDF onde tanto com 10% de treinamento como com 90% de treinamento, obteve acima de 95% de acurácia com menos de 8 segundos. E em relação a técnica de extração de características TF o PCC ainda se mostrou ser o mais eficiente entre os algoritmos testados.

Podemos concluir então que, o modelo de Competição e Cooperação entre Partículas (PCC) mostra ser até o momento a solução mais adequada dentre as testadas em relação a eficiência se adequando mais a técnica de extração de características *Term Frequency - Inverse Document Frequency* (TF-IDF) independente da quantidade de dados para treinamento.

Para trabalhos futuros, fica a necessidade de explorar novos algoritmos de extração de caracerísticas como o *Word2vec* [Aguiar, 2016], novos modelos de aprendizado de máquina como o OPF (*Optimum-Path Forest*)[Papa et al., 2012], e diferentes combinações, a fim de conquistar ainda melhores resultados dos que já foram obtidos com essa pesquisa na tentativa de melhor solucionar o problema de detecção de mensagens de *spam*.

Referências

E. M. Aguiar. Aplicação do word2vec e do gradiente descencente estocástico com tradução automática, 2016. Tese de Mestrado da Fundação Getulio Vargas, Escola de Matemática Aplicada. Rio de Janeiro.

E. J. Araújo. Método hibrido com detecção de regiões promissoras baseado em dencidade para o problema de localização de rótulos cartográficos, 2016. Tese de Mestrado, Instituto de Ciências e Tecnologia - UNIFESP.

A. Blum. Machine learning theory. *Carnegie Melon Universit, School of Computer Science*, page 26, 2007.

F. A. Breve. Classificação de imagens tomográficas de ciência dos solos utilizando redes neurais e combinação de classificadores., 2005. Monografia (Mestrado em Ciências da Computação), UFSCar - Centro de Ciências Exatas e da Terra - Departamento de Computação, São Carlos - SP.

F. A. Breve. Aprendizado de máquina em rede complexa, 2010. Tese de Douturado, Institudo de Ciências Matemáticas e de Computação, Universidade de São Paulo.

F. A. Breve, L. Zhao, and M. G. Quiles. Particle competition and cooperation for semi-supervisioned learning with label noise. *Neurocomputing*, 160:63–72, 2012.

E. M. N. Brito. Mineração de textos: Detecção automática de sentimentos em comentários nas mídias sociais., 2016. Monografia (Mestrado em Sistemas de Informação e Gestão de Conhecimento), FUMEC - Faculdade de Ciências Empresariais, Belo Horizonte-MG.

C. O. Carneloz. Auxílio no diagnóstico da doença de alzheimer a partir de imágens de ressonância magnética utilizadno competição e cooperação entre partículas, 2019. Tese (Mestre Ciência da Computação, área de Inteligência Artificial), Universidade Estadual Paulista "Júlho de Mesquita Filho ", Rio Claro - SP.

Cert.br. Centro de estudos, resposta e tratamento de incidentes de segurança no brasil,

cartilha de segurança para internet. *Cartilha de Segurança para a Internet*, 2016, 2016. URL `https://cartilha.cert.br`.

O. Chapelle, B. Schölkopf, and A. Zian. Semi-supervisioned learning (chapelle, o. et. al., eds.;2006)[book reviews]. *IEEE Transaction on Neural Networks*, 20(3):542–542, 2009.

D. Choi, B. Ko, H. Kim, and P. Kim. Text analysis for detecting terrorism-related articles on the web., 2013. ISSN 1084-8045. Journal of Network and Computer Applications.

M. Covington, D. Nute, and A. Vellino. *Prolog Programming in Depth.* Prentice-Hal, 1997.

J. de Fernandes Teixeira. *Inteligência artificial*. Pia Sociedade de São Paulo-Editora Paulus, 2014.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM, 1998.

T. P. Faleiros. Propagação em grafos bipartidos para extração de tópicos em fluxo de documentos textuais, 2016. Tese de Mestrado do Institudo de Ciências Matemáticas e Computação ICMC-USP, São Carlos.

H. Faris, A. M. Al-Zoubi, A. A. Heidari, I. Aljarah, M. Mafarj, M. A. Hassonah, and H. Fujita. An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks, 2018. Information Fusion.

J. Gauri, S. Manisha, and A. Basant. Optimizing semantic lstm for spam detection, 2019. International Journal of Information Technology.

R. F. Gonzalez and R. E. Woods. Processamento de imagens digitaisn, 2000. São Paulo: Edgar Blücher LTDA.

A. C. E. S. Lima and L. N. Castro. Automatic sentiment analysis of twitter messages., 2012. Computational Aspects of Social Networks. Fourth International Conference.

D. P. Marinho. Detecção de spam através de aprendizado de máquina, 2019. Trabalho de Conclusão de Curso - Faculdade de Tecnologia Alcides Maya - AMTEC, Porto Alegre - SC.

H. Mark, R. Erik, F. George, and S. Jaap. Spambase data set, 1999. https://archive.ics.uci.edu/ml/datasets/Spambase.

T. G. Milani. Honeypot para detecção e contenção de invasões e Botnets, 2011. Monografia (Bacharel em Sistemas de Informação), UNICEP (Centro Universitário Central Paulista), São Carlos, Brasil.

T. G. Milani. Introdução a Análise Forense Computacional: Detectando Rootkits em Ambiente Windows, 2016. Monografia (Especialização MBA em Tecnologia da Informação com Ênfase em Segurança da Informação), UNIARARAS, Araras, Brasil.

T. M. Mitchell. Machine learning, 1997. [S.I.]: McGraw-Hill Boston, MA.

V. Moll. Detecção de intrusão uzando técnica de aprendizado de máquinas, 2010. Dissertação Mestrado Universidade Federal de Santa Catarina, Florianópolis - SC.

M. N. Murty, A. Jain, and P. Flynn. Data clustering: a review acm compt. surv. *ACM Computing Surveys*, 31(3), 1999.

D. H. Negretto. Algoritmo de aprendizado semi-supervisionado baseados em grafos aplicado na bioinformática, 2016. Tese de Mestrado da Universidade Estadual Paulista "Julho de Mesquita Filho"Programa Multi-Campus.

E. C. Nhassengo. Processamento de sinal em grafos: Teoria de amostragem e sua aplicação no aprendizado semi-supervisionado ativo, 2019. Tese de Mestrado do Institudo de Ciências Matemáticas e Computação ICMC-USP, São Carlos.

J. P. Papa, A. X. Falcão, V. H. C. Albuquerque, and J. M. R. S. Tavares. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, 45:512–520, 2012.

C. Park, J. Ahn, H. Kim, and S. Park. Integrative gene network construction to analyze cancer recurrence using semi supervisioned learning. *PloS one*, 9(1), 2014. e86309.

M. G. Quiles, L. Zhao, R. L. Alonso, and R. A. Romero. Particle competition for complex network community detection. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 18(3), 2008. e86309.

U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect comunnity structure in large-scale network. *Physical Review E*, 76(3), 2007. 036106.

J. Ramos. Using tf-idf to determine word relevance in document queries., 2003. Technical report, Department of Computer Science, Rutgers University, 23515 BPO Way, Piscataway, NJ, 08855e.

P. L. V. Ribeiro. Uma abordagem unificada para análise de sentimento de tweets com domínio específico., 2015. Tese de Mestrado, Instituto de Ciências Exatas, Departamento de Ciência da Computação, Universidade de Brasília.

L. A. S. Romoni, B. F. Amaral, R. R. V. Gonçalves, J. Z. Júnior, and E. P. M. Sousa. Aplicação de técnicas de classificação semissupervisionada para análise de séries multitemporais de imagens de satélite. *XVI Simpósio Brasileiro de Sensoriamento Remoto*, 2013.

S. Sedhai. Semi-supervised spam detection in twitter stream, 2017. IEEE Transactions on Computational Social Systems.

X. Song and A. K. A. Malcolm. Spam text message classification, 2017. https://www.kaggle.com/team-ai/spam-text-message-classification.

A. A. Tiago. Sms spam collection data set, 2012. https://archive.ics.uci.edu/ml/datasets/sms+spam+collection.

R. H. Tiboli. Detecção de spam em mensagens sms utilizando aprendizado de máquina, 2019. Trabalho de Conclusão de Curso - Universidade Federal Tecnológica do Paraná - UTFPR, Departamento Acadêmico de Computação, Medianeira - PR.

E. D. Vechia. *Perícia Digital: Da Investigação à Análise Forense.* Millenium, 2014.

F. Wang and C. Zhang. Label propagation through linear neighborhoods. *IEEE Trans. on Knowl. and Data Eng.*, 20(1):55–67, 2008.

S. M. E. A. Weiss. Text mining: predictive methods for analyzing unstructured information., 2010. [S.l.]: Springer Science Business Media.

I. H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques., 2011. [S.L.]:Morgan Kaufmann.

T. Wu, S. Liu, J. Zhang, and Y. Xiang. Twitter spam detection based on deep learning, 2017. Proceedings of the Australasian Computer Science Week Multiconference.

Z. Xiaojin and G. Zoubin. Learning from labaled and unlabaled data with label propagation. *Tecnical Report: Carnegio Mellon University*, 2002.

R. Yafeng and j. Donghong. Neural networks for deceptive opinion spam detection: An empirical study, 2017. IEEE Transactions on Computational Social Systems.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems 16.*, pages 321–328, 2004.

X. Zhu. Semi-supervised learning literature survey. *Pattern Recognition*, 2005.

X. Zhu. Semi-supervised learning. *Encyclopedia of machine learning*, pages 892–897, 2011.