UM ALGORITMO EVOLUCIONÁRIO PARA MINERAÇÃO DE FLUXO DE DADOS EM MICROBLOGS

JULIANA OLIVEIRA FERREIRA

UM ALGORITMO EVOLUCIONÁRIO PARA MINERAÇÃO DE FLUXO DE DADOS EM MICROBLOGS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADORA: GISELE LOBO PAPPA

Belo Horizonte

Março de 2013

Ferreira, Juliana Oliveira
F383a Um algoritmo evolucionário para mineração de fluxo de dados em Microblogs / Juliana Oliveira Ferreira. — Belo Horizonte, 2013
xxii, 62 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Gisele Lobo Pappa

1. Computação - Teses. 2. Mineração de Dados - Teses. 3. Algoritmos Evolucionários - Teses. 4. Algoritmos Genéticos - Teses. I. Título.

CDU 519.6*73 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Um algoritmo evolucionário para mineração de fluxo de dados em microblogs

JULIANA OLIVEIRA FERREIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROFA. GISELE LOBO PAPPA - Orientadora
Departamento de Ciência da Computação - UFMG

PROF. LUIZ HENRIQUE DE CAMPOS MERSCHMANN
Departamento de Computação - UFOP

PROF. OMAR PARANAIBA VILELA NETO
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 01 de julho de 2013.

Agradecimentos

A Deus por permitir a concretização deste trabalho e pela força dada nos momentos mais difíceis desta caminhada.

Aos meus queridos pais, Arildo Barbosa Ferreira e Cássia Junqueira Oliveira Ferreira, sempre presentes nessa longa caminhada. Obrigado por tudo o que fizeram e fazem por mim até hoje.

As minhas irmãs, Adriana Oliveira Ferreira e Patrícia Oliveira Ferreira, aos meus avôs e tios pela amizade e apoio contínuo.

Ao meu namorado, Saulo Araújo Naves, por simplesmente tornar meus dias mais alegres e especiais.

A minha madrinha, Neiva Barbosa Ferreira, pelo apoio e suas palavras de carinho.

A minha orientadora, Prof.Gisele Lobo Pappa, pelos conhecimentos transmitidos, amizade e acima de tudo pela oportunidade e paciência ao longo deste trabalho.

A Universidade Federal de Minas Gerais (UFMG) e aos vários Laboratório que participei, que foram os alicerces para esta realização.

Aos meus colegas, pela amizade e companheirismo. A todos aqueles que de alguma forma ajudaram na realização deste trabalho. Muito Obrigada!

"Existe uma coisa que uma longa existência me ensinou: toda a nossa ciência, comparada à realidade, é primitiva e inocente; e portanto, é o que temos de mais valioso"
(Albert Einstein)

Resumo

A amplitude e a velocidade com que a informação passou a ser gerada através dos diversos recursos disponibilizados na Web, entre eles as redes sociais, provocaram a necessidade de mudanças de paradigma na maneira como os sistemas baseados em dados trabalham, gerando sistemas em que o fluxo de dados é contínuo e o conhecimento não é estacionário. Entretanto, os algoritmos de mineração antes utilizados para extrair padrões e informações desses dados não são ideais para tratar desses novos tipos de problemas, sendo os principais desafios, relacionados as seguintes perguntas: (i) Quais dados devem ser mantidos e quais devem ser descartados durante o processo de aprendizagem? (ii) Quando se deve atualizar o modelo de classificação? (iii) Como atualizar o modelo?

Visando lidar com esses desafios, este trabalho propõe a utilização de um algoritmo evolucionário (AE) para aprendizado em fluxos de dados que seja capaz de explorar a evolução dos classificadores justamente com a evolução dos dados. Uma das principais razões de utilizarmos AE é que este possui uma população de possíveis soluções para o problema que tende a evoluir ao longo do tempo através da seleção dos indivíduos mais aptos e operações de cruzamento e mutação. Essa característica pode ser aproveitada de forma que, com o passar do tempo, tanto os modelos quanto os dados evoluam simultaneamente.

Considerando os principais desafios apresentados anteriormente, a técnica basicamente ataca cada um desses desafios da seguinte forma: a técnica utiliza um método baseado em repositório, que nada mais é do que uma estrutura onde armazenamos um conjunto de exemplos que sejam capazes de representar o fluxo. Para saber quando o modelo deve ser atualizado foi utilizado o teste estatístico de *Page-Hinkley* (PH), que detecta mudanças no desempenho dos classificadores. O modelo é atualizado aproveitando os operadores de evolução dos Algoritmos Genéticos. O vocabulário evolui juntamente com a população, baseado em sua frequência e entropia ao longo do tempo.

O foco da pesquisa são bases de dados com texto curto e vocabulário extenso,

como na rede social *Twitter*. A técnica foi validada utilizando 4 bases de dados de eventos distintos. Os resultados foram comparados a dois algoritmos estado da arte na literatura, e a técnica foi melhor em uma base e perdeu nas três outras. Entretanto, nos casos onde a técnica obteve melhor resultado, a diferença foi maior que nos casos onde perdeu.

Palavras-chave: Algoritmos Evolucionários, Mineração de Dados, Fluxo de dados contínuos, Data Streams, Algoritmos Genéticos, Classificador.

Abstract

The outgrowing number of information posted by users in social network, together with other resources provided by the Web 2.0, asked for a paradigm shift in the way data-based systems work. A few well-behaved data instances were replaced by a continuous and non-stationary data flow. Hence, traditional mining algorithms used to extract patterns from data had to be adapted for dealing with these new reallity. Given the nature of data flows, algorithms had to learn how to deal with at least three challenges: (i) What data should be kept and which should be discarded during the learning process? (ii) When should the classification model be updated? (iii) How should it be updated?

In this direction, this paper proposes an evolutionary algorithm (EA) for learning in data streams that able to explore the evolution of classifiers together with the evolution of the data. One of the main reasons we use AE is that it has a population of possible solutions to the problem which tends to evolve over time by selecting the fittest individuals and operations of crossover and mutation. This feature can be exploited so that, over time, both models as data evolve simultaneously.

The proposed algorithm works with a dynamic vocabulary, and tackles the three challenges aforementioned. It uses a method based on a data repository, which stores a predefined set of instances. The Page-Hinkley (PH) statistical test is used to detect changes in the performance of classifiers, signaling when the model should be retrained.The model is updated leveraging the evolution operators of the EA.

The method was tested in four datasets of short text and extensive vocabulary colleted from Twitter, each of them corresponding to a real-life event. The results were compared with two state of the art algorithms from the literature, and the technique was better on a base and the other tree lost. However, in cases where the technique had the best results, the difference was greater than in cases where lost.

Keywords: Evolutionary Algorithms, Learning, Data Streams, Classifiers.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AE	Algoritmo Evolucionário
AG	Algoritmo Genético
DP	Desvio Padrão
EM	Entropia Máxima
HT	Hoeffding Tree
JDA	Janela Deslizante Ativa
MOA	Massive Online Analysis
M	Média
NBM	Naive Bayes Multinomial
PH	Page-Hinkley
SGD	Stochastic Gradient Descent

Sumário

Capítulo 1

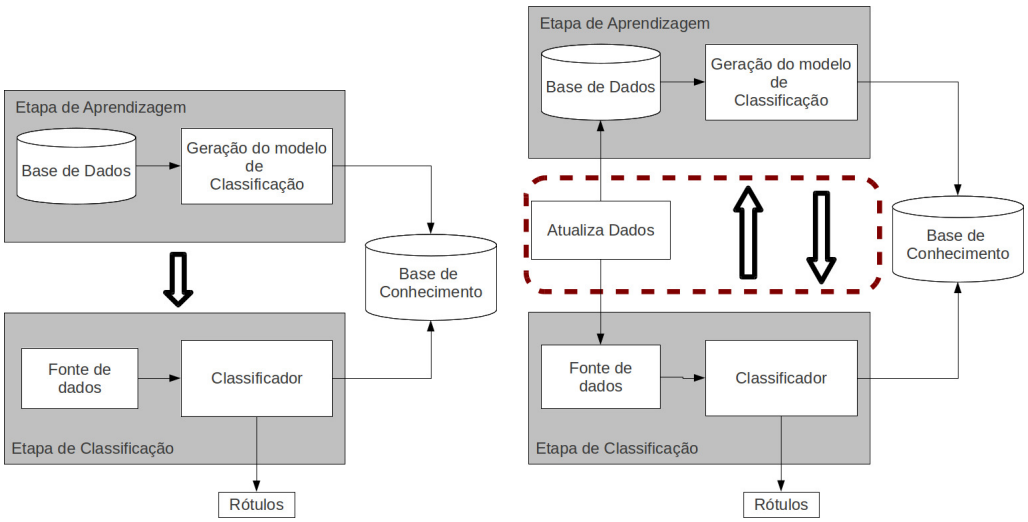
Introdução

Os diversos recursos disponibilizados na Web 2.0 têm possibilitado às pessoas desfrutarem de várias facilidades, podendo não apenas consumir informação a qualquer forma e de qualquer lugar, mas também gerar informação. Isso provocou uma mudança de paradigma e, consequentemente, mudou também a maneira como os sistemas baseados em dados trabalham. Muitos sistemas tradicionais baseados em dados são alimentados em intervalos de tempo fixos ou variáveis. Porém, essa nova maneira de gerar e consumir dados gerou sistemas em que o fluxo de dados é contínuo. Avaliando situações do nosso dia a dia, alguns exemplos desses fluxos são: transações de cartão de crédito, informações de redes sociais, controle de tráfego urbano, transações na bolsa de valores, informações de sensores de redes, entre outros.

Entretanto, os algoritmos de mineração de dados antes utilizados para extrair padrões e informações de dados provenientes de sistemas com um fluxo de dados "bem comportados" não são ideais para aplicações que trabalham com fluxo de dados contínuo. Por exemplo, uma das tarefas de mineração de dados mais estudadas é a classificação. Na classificação, cada instância (exemplo) da base contém um rótulo (classe a qual a instância pertence) e algumas características que descrevem esta instância (atributos). Os algoritmos utilizados para este problema procuram relações entre os atributos e o rótulo de cada instância em uma base de treinamento, tentando encontrar padrões comuns entre os exemplos, expressos através de um modelo. De posse desse modelo, torna-se possível classificar novas entradas diferentes das presentes no conjunto de treinamento, denominadas conjunto de teste [Rezende, 2003; Zhu & Goldberg, 2009].

Quando trabalhamos com dados contínuos, a distinção entre dados de treino e teste se torna difusa, uma vez que o conjunto de treinamento pode crescer indefini-

damente, os dados chegam em grande quantidade e em alta velocidade, a memória do computador é limitada - impossibilitando o processamento e armazenamento de todos os dados do fluxo e a distribuição dos dados pode alterar ao longo do tempo. Todos esses são considerados desafios para a área de mineração de dados.



(a) Modelo genérico para dados estacionários.

(b) Modelo genérico para dados não estacionários.

Figura 1.1. Diferença entre o funcionamento de algoritmos de mineração para dados estacionários 1.1(a) e não estacionários 1.1(b)

Os componentes básicos das técnicas de classificação são mostrados na Figura 1.1 [Bifet & Kirkby, 2009]. As técnicas baseadas na Figura 1.1(a) são utilizadas com sucesso para dados estacionários (que não sofrem mudanças ao longo do tempo). Algumas adaptações destes algoritmos foram desenvolvidas para tratar de dados criados em fluxo contínuo e em grande quantidade, tais como Algoritmo *Hoeffding Tree* (HT)[Domingos & Hulten, 2000] e *Naive Bayes Multinomial* (NBM)[Kibriya et al., 2004]. Nestes cenários, considera-se que a distribuição dos dados tende a mudar ao longo do fluxo, tornando-se necessário a existência de um processo constante de aprendizagem. A Figura 1.1(b) descreve os componentes e o funcionamento geral dos algoritmos de classificação para fluxos de dados contínuos.

Avaliando as imagens da Figura 1.1 podemos perceber que a grande diferença entre os dois modelos é que, enquanto o estacionário realiza a etapa de aprendizagem apenas uma vez, o não estacionário realiza esta etapa várias vezes. Além disso, devido a grande quantidade de dados que chegam constantemente e a impossibilidade de armazenar todos esses dados, o não estacionário possui um componente

para atualizar a base de dados utilizada na etapa de aprendizagem.

Baseado nas definições acima, os principais desafios da mineração em fluxo de dados estão relacionados as seguintes perguntas:

1. Quais dados devem ser mantidos e quais devem ser descartados durante o processo de aprendizagem?
2. Quando se deve atualizar o modelo de classificação?
3. Como atualizar o modelo?

Diferentes algoritmos para classificação em fluxo contínuo foram desenvolvidos nos últimos anos [Wang et al., 2003; Street & Kim, 2001; Folino et al., 2008; Folino & Papuzzo, 2010; Folino et al., 2007; Gama & Rodrigues, 2009; Bifet & Kirkby, 2009; Silva, 2012]. A maioria dos trabalhos encontrados na literatura por esta pesquisa são baseada em árvores de decisão [Gama & Rodrigues, 2009; Folino et al., 2008; Bifet & Kirkby, 2009; Folino et al., 2007; Kumar et al., 2011; Hashemi & Yang, 2009; Silva, 2012] e utilizam um comitê de classificação [Folino et al., 2007; Folino & Papuzzo, 2010; Folino et al., 2008; Street & Kim, 2001; Wang et al., 2003; Kumar et al., 2011].

Além disso, grande parte dos algoritmos atuais retreina o modelo a medida que as características dos dados do fluxo mudam. Considerando essas características, existe um tipo de algoritmo que poderia tratá-las naturalmente, levando os classificadores a evoluir com o tempo a medida que os dados também "evoluem": algoritmos evolucionários.

Algoritmos Evolucionários (AE) tem sido utilizados com sucesso para problemas de classificação estáticos [Pei et al., 1995; Tanwani & Farooq, 2009; Mola & Miele, 2006] e são baseados em populações. Cada população é composta por vários indivíduos que representam possíveis soluções para o problema. Nesse trabalho, cada indivíduo representa um classificador. Uma das principais características dos AE é que a população tende a evoluir ao longo do tempo através da seleção dos indivíduos mais aptos e operações de cruzamento e mutação. Essa característica pode ser aproveitada de forma que, com o passar do tempo, tanto os modelos quanto os dados evoluam simultaneamente. Além disso, AEs trabalham implicitamente com um comitê de indivíduos, podem representar árvore ou regras de decisão ou representações ainda mais simples, lidam bem com incertezas e são tolerantes a ruídos [Bäck, 2002].

Assim, este trabalho propõe um novo método para classificação de fluxos de dados baseado em algoritmos evolucionários, que trabalha com vocabulário dinâmico e sempre retreina o modelo considerando parte das soluções já encontradas

pelo AE. Focamos em um tipo especial de dados: textos curtos provenientes de microblogs como o Twitter. O Twitter vem se tornando uma fonte de dados interessante para os mais diversos fins, tais como: medir influência de usuários em redes sociais [Cha et al., 2010], avaliar tendências políticas [Diakopoulos & Shamma, 2010; Silva, 2012], vigilância epidemiológica [Gomide et al., 2012; Silva et al., 2011], entre outros. Na Universidade Federal de Minas Gerais (UFMG), por exemplo, foi desenvolvida uma ferramenta online dedicada ao monitoramento de importantes fatos, eventos e entidades na web em tempo real (Observatório da Web[1]).

O Observatório já foi utilizado para coletar *tweets* (nome dado às mensagens postadas no Twitter) relacionados a diferentes temas, entre eles Dengue [Gomide et al., 2012; Silva et al., 2011], futebol e campanha política [Silva, 2012]. Essas bases de dados foram utilizadas nesta pesquisa para avaliação do método proposto.

1.1 Objetivos

Tentando aproveitar a característica dos AE de evoluir indivíduos ao longo do tempo, este trabalho propõe o uso destes algoritmos para o desenvolvimento de um método para classificação de dados que chegam em um fluxo contínuo, focando em microblogs. Vale ressaltar aqui que, inicialmente, a técnica proposta neste trabalho é baseada em aprendizado supervisionado e classificação binário. A ideia principal consiste em estudar se é possível **gerar modelos que possam evoluir junto com os dados**. Para tal, os desafios da classificação de dados em fluxos contínuos, definidos pelas três perguntas anteriores (como, quando e com que dados o modelo deve ser atualizado), devem ser atacados para que o conhecimento adquirido seja preservado ao longo do tempo.

Outros objetivos mais específicos foram visados pelo presente trabalho, entre eles:

- Levantamento bibliográfico de métodos de classificação para fluxos de dados;
- Identificação dos métodos mais apropriados para comparação com o método proposto;
- Avaliar o método proposto para bases extraídas do Twitter, considerando suas características temporal (momento da chegada da mensagem);

[1] http://observatorio.inweb.org.br/

1.2 Principais Contribuições

Dados os objetivos apresentados anteriormente, as principais contribuições desta dissertação são:

- Um novo método baseado em Algoritmo Genético adaptado para trabalhar com fluxos de dados contínuos, focando em bases retiradas de microblogs como o *Twitter*.
- Desenvolvimento de um algoritmo de classificação sequencial que aproveita a característica de evolução dos algoritmos genéticos para adaptar os classificadores na medida em que os dados evoluem.

1.3 Organização da Dissertação

Este trabalho está organizado da seguinte forma: O Capítulo 2 apresenta alguns aspectos teóricos relacionados a mineração de dados em fluxo contínuo (*data streams*) e uma breve descrição de Algoritmos Evolucionários.

Já o Capítulo 3 apresenta o método proposto baseado em Algoritmo Genético para mineração em fluxo de dados contínuos proposto por esta pesquisa. O Capítulo 4 apresenta os resultados alcançados e comparações com outros algoritmos para classificação de *streams*. Finalmente, as conclusões gerais do projeto e trabalhos futuros encontram-se no Capítulo 5.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta algumas definições relacionadas às duas área de pesquisa que fundamentam a proposta apresentada neste trabalho: mineração de fluxo de dados contínuo e computação evolucionária. A Seção 2.1 introduz conceitos relacionados à mineração de fluxo de dados contínuos, enquanto a Seção 2.2 apresenta conceitos básicos sobre algoritmos evolucionários.

2.1 Mineração de Fluxos de Dados Contínuo

Com as facilidades proporcionadas pela utilização da Web ocorreu um aumento significativo no volume de informações disponibilizadas em diversas áreas do conhecimento. Este fato criou uma nova classe de aplicações baseadas em mineração de fluxos de dados contínuos. A modelagem para fluxos de dados contínuos considera que [Bifet & Kirkby, 2009]:

- Existe um fluxo contínuo de dados;
- A quantidade de dados que chega é grande e a velocidade de chegada é alta;
- Somente uma pequena parte dos dados pode ser processada e armazenada, sendo o restante dos dados descartado;
- A distribuição dos dados pode ser alterada ao longo do tempo.

Analisando as características acima, os principais desafios encontrados em mineração de fluxos de dados contínuos podem ser resumidos nas seguintes perguntas:

1. Quais dados devem ser armazenados e quais dados devem ser descartados?

2. Quando devemos atualizar o modelo?

3. Como devemos atualizar o modelo?

As próximas seções descrevem os principais métodos para tratar desses problemas. Em seguida, apresentamos os principais trabalhos relacionados a esta pesquisa. É importante ressaltar que, como esse trabalho pesquisa a tarefa de classificação de fluxo de dados contínuos, sempre o termo modelo refere-se a modelos de classificação.

2.1.1 Quais e quantos dados devem ser utilizados no processo de aprendizagem?

O termo *Concept Drift*, em aprendizado de máquina, significa que a distribuição dos dados de entrada muda ao longo do tempo de forma não previsível [Gama, 2010]. Como resultado, as predições passam a ser menos precisas a medida que o tempo passa. Para ilustrar esse conceito, tomemos como exemplo uma das bases de dados utilizadas nesse trabalho, relacionada a uma partida de futebol (uma descrição mais detalhada da base aparece na Seção 4.1). Esta base contém dados de um jogo polêmico entre Brasil e Holanda, no qual um jogador brasileiro, Felipe Melo, teve participação decisiva no jogo. O jogador foi fundamental para o primeiro gol da partida, mas logo após, este mesmo jogador marcou um gol contra e minutos depois foi expulso do jogo. A derrota neste jogo eliminou o Brasil da Copa do Mundo. Desta forma, os dados refletem a opinião dos torcedores em relação a este jogador (Sentimento positivo ou negativo). A Figura 2.1 mostra a mudança na distribuição do sentimento ao longo do jogo. Para avaliar estas alterações a base foi dividida em pequenos blocos e a forma de divisão da base foi determinado tentando criar blocos de dados que possibilitem a análise da mudança ao longo do tempo. No início do jogo os *tweets* apresentam sentimento positivos em relação ao jogador, e mudam ao longo do tempo da partida (após o gol contra e a expulsão do jogador).

Considerando esta tendência dos fluxos de dados, torna-se essencial retreinar o modelo para que este consiga se adaptar às novas informações da base, tanto capturando novos conhecimentos como eliminando os defasados. Entretanto, o computador possui memória limitada não sendo possível armazenar todos os dados do fluxo. Essa situação define o primeiro problema em fluxo de dados: quais e quantos dados devem ser utilizados no processo de aprendizagem?

A fim de tratar esse primeiro problema, diversos métodos foram desenvolvidos para limitar a sequência de eventos a serem tratados. Esses métodos devem não

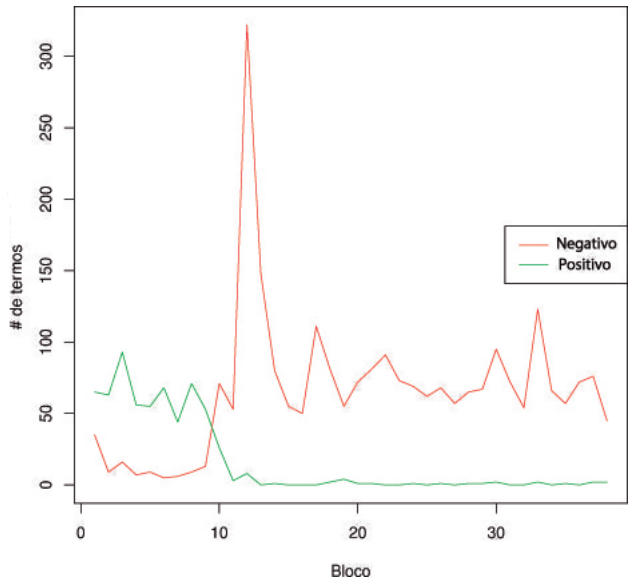


Figura 2.1. Concept Drift para base de dados referente ao sentimento da torcida em relação ao jogador Felipe Melo

apenas assimilar novos exemplos, mas também identificar e efetivamente remover os dados de treinamento que já não descrevem o fluxo corrente. Um dos métodos mais conhecidos para limitar a sequencia de eventos é o da janela de eventos [Chaudhry et al., 2005]. As janelas definem quais e quantos dados serão armazenados e, normalmente, são definidas em função de dois critérios:

- Tamanho da janela - pode ser baseado em tempo (tamanho definido por um intervalo de tempo) ou em evento (tamanho definido por uma quantidade de eventos estabelecidos de acordo com o domínio do problema);

- Escopo da janela - Pode ser móvel ou por marcação. Na janela móvel o tamanho é fixo, mas seus pontos de início e fim podem movimentar-se, substituindo eventos antigos por eventos mais novos. Na janela por marcação, um dos pontos (inicio ou fim) é mantido fixo enquanto o outro se movimenta, desta forma, a janela não possui tamanho fixo.

A técnica de janela é uma maneira simples de lidar com *concept drift*. A ideia é utilizar apenas as últimas mensagens recebidas ao invés do fluxo todo. O tamanho dessa janela deve ser configurado previamente pelo usuário, e geralmente a taxa de mudança dos fluxos de dados não é conhecida. Assim, uma janela pequena pode refletir bem o conceito atual do fluxo, mas não conter dados suficientes para que o classificador alcance uma eficácia esperada, e uma janela grande pode permitir ao

classificador alcançar um bom desempenho, mas demorar a detectar a ocorrência de uma mudança dos dados [Gama, 2010].

As abordagens que utilizam janelas consideram a ordem de chegada dos dados. Porém, alguns métodos trabalham com um reservatório de amostragem de dados do fluxo. O reservatório nada mais é que uma estrutura onde armazenamos um conjunto de exemplos que sejam capazes de representar o fluxo. Essa amostragem é realizada com uma função probabilística, que determina se um exemplo deve ser incluído ou retirado do reservatório, ou seja, essa função deve ser utilizada para determinar quais os dados serão utilizados no treinamento do modelo.

No trabalho de Al-Kateb et al. [2007], por exemplo, foram estudados reservatórios de amostragens em fluxos de dados com tamanho adaptativo sob duas perspectivas: tamanho do reservatório e uniformidade da amostra. Já em Silva [2012] foi proposto uma técnica de Janela Deslizante Ativa (JDA) que consiste em uma solução fundamentada na teoria do aprendizado ativo. Nesse trabalho, ao invés de escolher quais exemplos serão selecionados para entrar no conjunto de treinamento, o objetivo é permitir que o classificador escolha quais exemplos esquecer. Essa escolha é baseada em uma função que também foi utilizada em Veloso & Meira-Junior [2011], que considera que quanto maior a similaridade entre os exemplos e a idade do exemplo do treino, maior é a chance deste exemplo ser descartado. Assim, é possível prover ao classificador um maior ganho de informação com um viés temporal.

O *Naive Bayes Multinomial*, por sua vez, é um método incremental que supõe que todas as entradas são independentes e passa apenas uma vez por cada exemplo [McCallum & Nigam, 1998; Kibriya et al., 2004]. O método *Hoeffding tree* também é incremental, e assume que a distribuição de geração de exemplos não muda constantemente. Ele explora o fato de que uma pequena amostra pode ser suficiente para escolher um atributo com boa separação entre as classes. Esta ideia é suportada matematicamente pelo conceito de limite de Hoeffding (*Hoeffiding bound*), que quantifica o número de exemplos quando necessários para estimar o quão bom é um atributo [Domingos & Hulten, 2000]. Os dois últimos métodos (*Naive Bayes Multinomial* e *Hoeffiding Tree*) são bem adequados para lidar com fluxos de texto [Bifet & Kirkby, 2009], e serão utilizados como métodos de comparação para a técnica proposta no Capítulo 3.

2.1.2 Quando e como atualizar o modelo?

Já vimos que a distribuição dos dados pode mudar com o tempo e que o principal objetivo de algoritmos para classificação em fluxos de dados consiste em manter

um classificador atualizado. Dessa forma, o modelo gerado inicialmente deve ser retreinado para refletir essas mudanças. Existem duas formas principais de determinar quando retreinar o modelo [Gama, 2010]:

- Métodos cegos - métodos que adaptam o modelo em intervalos regulares, sem considerar se as mudanças realmente ocorreram. Alguns exemplos incluem métodos que utilizam janela de tamanho fixo e os métodos incrementais que atualizam sempre que chega um novo dado;
- Métodos Informados - métodos que modificam o modelo apenas quando uma mudança foi detectada.

Alguns trabalhos relacionados à primeira abordagem podem ser encontrados em Klinkenberg & Renz [1998]; Klinkenberg & Joachims [2000]; Lanquillon [2001]; Maloof & Michalski [2000]; Widmer & Kubat [1996b]. Esses métodos devem considerar que, como os algoritmos para fluxo trabalham com grandes quantidades de dados e as mudanças podem ocorrer em intervalos diferentes, a definição do tamanho do intervalo pode ter um efeito grande na eficiência e eficácia do método. Se determinarmos um intervalo consideravelmente pequeno podemos atualizar o modelo várias vezes sem ser necessário (isso pode ter um custo alto), porém conseguimos nos recuperar mais rápido caso ocorram mudanças. Se determinarmos um intervalo grande, evitando retreinar o modelo em excesso, podemos demorar a recuperar de mudanças ocorridas.

O método *Naive Bayes Multinomial* apresenta uma forma simples de atualizar o modelo, atualizando os contadores utilizados para estimar as probabilidades condicionais sempre que chega um novo dado. Ele considera as instâncias como um conjunto de termos (no caso de mineração de textos) e para cada classe determina a probabilidade de observar o termo t dado uma classe. O cálculo desta probabilidade é realizado através dos dados já apresentados ao método (dados de treinamento) simplesmente computando a frequência relativa para cada termo na coleção de instâncias de treinamento. Desta forma, o modelo é atualizado a cada nova instância. Já o método *Hoeffiding Tree* apresenta um modelo incremental, porém não o atualiza a todo momento. O classificador é uma árvore de decisão, e essa árvore só é atualizada quando um número n de exemplos (definido pelo limite de *Hoeffiding*) é recebido, para então escolher um atributo com boa separação entre as classes e atualizar o modelo de decisão.

Considerando métodos informados, independente da técnica utilizada, os sistemas de detecção consistem de três módulos, apresentados na Figura 2.1 [Bifet &

Kirkby, 2009]. Em geral, a entrada desses sistemas consiste em uma sequência de itens *x1,x2,..., xt,...* que variam sua distribuição de forma desconhecida ao longo do tempo. Como saída, esses sistemas retornam uma estimativa de alguns parâmetros importantes da distribuição dos dados de entrada, e um alarme caso tenham ocorrido mudanças nessa distribuição. Essa distribuição dos dados pode consistir na maneira como as instâncias estão divididas por classe, distribuição dos termos para cada classe, variação dos termos ao longo do tempo, entre outras informações.

A **memória** corresponde ao local onde o algoritmo armazena todos os dados que ele considera relevantes, no decorrer do tempo, para proporcionar informações das distribuições dos dados. O **estimador** é um algoritmo que retorna as estatísticas desejadas dos dados de entrada, podendo utilizar os dados da **memória**. O **detector de mudanças** possui como entrada a saída do **estimador** e pode ou não utilizar os dados da **memória**. A saída do detector é um alarme, acionado caso alguma mudança de conceito seja verificada.

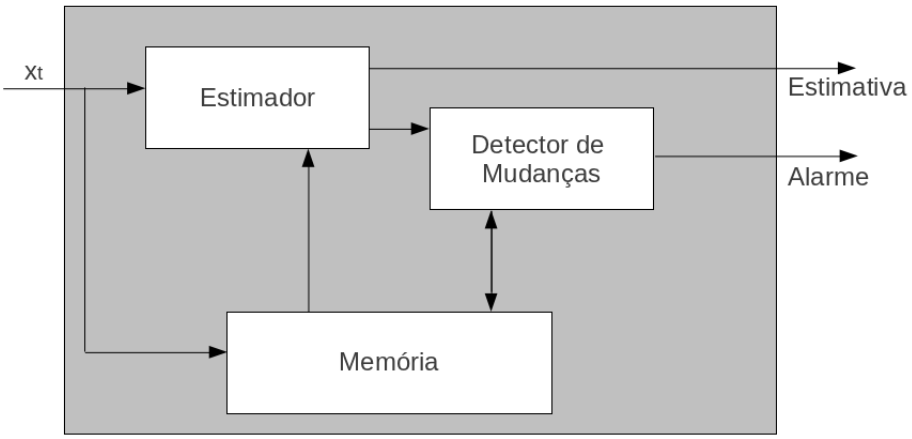


Figura 2.2. Modelo genérico de sistemas de detecção de mudanças de conceitos.

Quando utilizamos as técnicas de detecção de mudanças para atualização do modelo normalmente podemos seguir duas abordagens [Gama, 2010]:

- Acompanhar a evolução do desempenho dos classificadores. Considerando que a mudança de conceito (*concept drift*) inviabiliza o classificador manter um bom desempenho, uma vez que a distribuição dos novos dados não corresponde à distribuição dos dados iniciais [Bifet & Kirkby, 2009; Helmbold & Long, 1994; Widmer & Kubat, 1996b], o modelo pode ser retreinado quando seu desempenho (medido através de precisão, F1, ou qualquer outra medida convencional) piorar;

- Levar em conta a mudança na distribuição dos dados em duas janelas de dados distintas. A primeira janela, a de referência, resume informações passadas, enquanto a segunda janela gera informações sobre os exemplos mais recentes.

Relacionado a primeira abordagem de detecção, Klinkenberg & Renz [1998] propõe o monitoramento do valor de três indicadores de desempenho: precisão, revocação e precisão ao longo do tempo. Klinkenberg & Joachims [2000] apresentam um método teoricamente bem fundamentado para reconhecer e lidar com as mudanças de conceitos usando propriedades de *Support Vector Machine*. A ideia-chave é selecionar o tamanho da janela para que o erro de generalização estimado com novos exemplos seja minimizado.

Já relacionados a segunda abordagem, Kifer et al. [2004] apresentam um algoritmo (que usa um teste estatístico baseado em *Chernoff bound*) que examina amostras extraídas de dois conjuntos diferentes de dados para decidir se as distribuições são diferentes. Na mesma linha, o sistema VFDT, proposto em Gama & Pinto [2006], tem a capacidade de lidar com a mudança de conceito monitorando continuamente a diferença entre as distribuições de classes dos exemplos.

Li et al. [2007], por sua vez, introduzem um método de detecção de mudanças que assume que os pontos no fluxo de dados são gerados de forma independente, mas também assume a natureza do processo de geração dos mesmos. Esta abordagem utiliza uma função para determinar a distância entre dois exemplos, um valor limite para determinar se ocorreram mudanças e técnicas de amostragem para decidir quais pontos dos dados serão analisados.

2.1.3 Mineração de Fluxo de Dados no Twitter

Nos últimos anos a tarefa de analisar conteúdo disponíveis em redes sociais (Twitter, Facebook, entre outros) tem possibilitado a descoberta de conhecimento a partir do conteúdo expresso por milhares de usuários da Web oferecendo oportunidades estratégicas para diferentes áreas [Jansen et al., 2009; Diakopoulos & Shamma, 2010; Bifet & Frank, 2010].

No trabalho Jansen et al. [2009], por exemplo, foram analisados comentários sobre algumas marcas com o objetivo de verificar se micro-blogs podem ser considerados mecanismos para propaganda online, sendo verificado que estes micro-blogs podem ser utilizados tanto para divulgar como para obter informações dos clientes.

Diakopoulos & Shamma [2010] utilizaram uma base contendo comentários sobre debate político. Neste trabalho foram apresentados metodologias analíticas e re-

presentações visuais para compreender melhor a dinâmica temporal do sentimento em relação ao debate.

Bifet & Frank [2010], por sua vez, apresentam uma discussão sobre os desafios da mineração em fluxo de dados do Twitter. Nesta rede social, além dos desafios da mineração de fluxo geral distinguimos alguns outros problemas: incerteza nos dados, textos curtos e vocabulário extenso. Neste trabalho foram avaliados três algoritmos incrementais bem adequados para lidar com fluxo de textos para analise de sentimentos: *Multinomial Naive Bayes* (NBM), *Stochastic Gradient Descent* (SGD) e *Hoeffding Tree*. Em um primeiro experimento foi utilizado um conjunto de treino rotulado a partir dos símbolos que transmitem sentimentos contidos na mensagem (e.g. :) :( ), e um conjunto de teste rotulado manualmente. O algoritmo que mostrou maior eficácia nos experimentos foi o NBM. Já em um segundo experimento, foi utilizado somente um conjunto de dados rotulados a partir de símbolos que expressam sentimentos contidos nas mensagens. Nesse experimento o algoritmo que apresentou melhor resultado foi o SGD.

O trabalho [Silva et al., 2011; Silva, 2012] apresenta um classificador baseado em regras de associação com uma solução de esquecimento baseada em Janela de treino Deslizante Ativa (JDA). A abordagem proposta foi avaliada experimentalmente a partir da simulação da classificação em tempo real de mensagens enviadas através do Twitter, para quatro bases distintas. Destas bases, três foram utilizadas neste trabalho.

2.1.4 Avaliação de Classificadores para Fluxo de Dados

Um outro problema de mineração em fluxo de dados consiste na forma de avaliação do método. Os algoritmos estacionários apresentam uma separação bem definida de conjunto de treinamento (dados utilizados para treinar o modelo) e conjunto de teste (dados utilizados para avaliar o classificador encontrado), o que é mais difícil em fluxo de dados. Quanto a forma de avaliação de mineração em fluxo de dados, existem duas abordagens principais [Bifet & Kirkby, 2009]:

- *Holdout* - o desempenho é medido usando um conjunto simples de teste, ou seja, algumas instâncias são separadas e usadas apenas para avaliar a qualidade do classificador;

- *Interleaved Test-Then-Train ou Prequential* - cada exemplo é usado para testar o modelo antes de ser usado no conjunto de treinamento, e a precisão é atualizada de forma incremental. Ou seja, as instâncias que chegam são classifica-

das, avaliamos a classificação incrementando o resultado da nova instância, e em seguida esta instância pode ser adicionada ao conjunto de treinamento.

A segunda técnica é mais utilizada, dado que não existe uma definição clara de quais dados seriam conjunto de treinamento (dados utilizados para treinar o modelo) e conjunto de teste (dados utilizados para avaliar o modelo) [Bifet & Kirkby, 2009], e foi utilizada para avaliação dos classificadores nesse trabalho. Considerando esta técnica, existem duas formas diferentes de adicionar a instância no conjunto de treinamento:

- Através de aprendizado supervisionado - nesta técnica a instância é adicionada ao conjunto de treinamento considerando a classe real da instância.
- Através de aprendizado semi-supervisionado - aqui a instância é adicionada ao conjunto de treinamento considerando a classe que o classificador determinou para esta.

Este trabalho utiliza aprendizado supervisionado, sendo o aprendizado semi-supervisionado considerado como trabalhos futuros. As próximas seções descrevem os principais conceitos de Algoritmos Evolucionários.

2.2 Algoritmos Evolucionários

Os Algoritmos Evolucionários foram inicialmente propostos em 1950 e consistem em modelos computacionais inspirados na teoria da evolução [Back, 1996]. Eles operam sobre uma população de indivíduos, onde cada indivíduo representa uma possível solução para o problema. Eles se dividem em quatro grandes grupos: algoritmos genéticos, estratégias de evolução, programação evolucionária e programação genética [Freitas, 2002].

Este trabalho foca no uso de algoritmos genéticos (AG). Eles são algoritmos não determinísticos que constroem um conjunto de possíveis soluções para um dado problema, chamados indivíduos. Esses indivíduos evoluem ao longo do tempo através de operadores que combinam soluções distintas ou realizam pequenas modificações aleatórias em soluções já existentes, procurando uma solução adequada. Assim, uma das principais qualidades dos AGs é a característica de evolução, que pode ser combinada com a característica temporal de fluxo de dados como o Twitter para criar métodos mais dinâmicos e interessantes.

AGs têm se mostrado eficientes para a busca de soluções ótimas ou aproximadas do ótimo em uma grande variedade de problemas [Tanwani & Farooq, 2009;

Mola & Miele, 2006; Kumar et al., 2011]. Foram propostos inicialmente por **?** e são baseados nos mecanismos naturais de sobrevivência e reprodução das populações, onde os indivíduos mais adaptados ao seu ambiente sobrevivem e reproduzem a taxas mais altas do que indivíduos menos adaptados [Bäck, 2002].

Quando empregarmos os AGs em problemas do mundo real, cada indivíduo da população, também conhecido como cromossomo, corresponde a uma possível solução para o problema em questão. Esses indivíduos são tradicionalmente modelados como estruturas de dados fixas, definidas por sequências de bits, denominados genes, que representam a presença (1) ou ausência (0) de uma determinada característica. Na Figura 2.3 podemos observar um exemplo de cromossomo e gene.

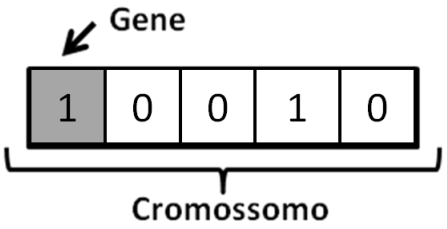


Figura 2.3. Exemplo de Indivíduo do Algoritmo Genético

Os AGs possuem um conjunto de passos distintos e bem definidos, sendo que cada passo possui muitas variações possíveis. O fluxograma básico do AG pode ser visto na Figura 2.2. O AG começa seu processamento com a criação de uma população inicial formada por um conjunto aleatório de indivíduos (possíveis soluções do problema). Após ser criada, a *fitness* dos indivíduos é calculada e a condição de parada é verificada. Se atendida, o melhor individuo é apresentado e o algoritmo é encerrado. Caso contrário, inicia-se o processo de evolução dos indivíduos, composto por uma série de iterações (gerações) executadas até que seja satisfeita essa condição de parada. As populações em AG possuem tamanho fixo, e depois de criada uma nova população, a antiga é descartada.

Após criada a população inicial, as gerações subsequentes consiste em:

- Passo 0 - Seleciona o tipo de operador a ser utilizado(Mutação ou Cruzamento). Em caso de mutação realizamos o passo 1, caso contrário, o passo 3.
- Passo 1 - Seleciona dois indivíduos da população existente;
- Passo 2 - Realiza cruzamento entre os dois indivíduos selecionados e adiciona os novos indivíduos gerados na população. Logo após, realizamos o passo 5.

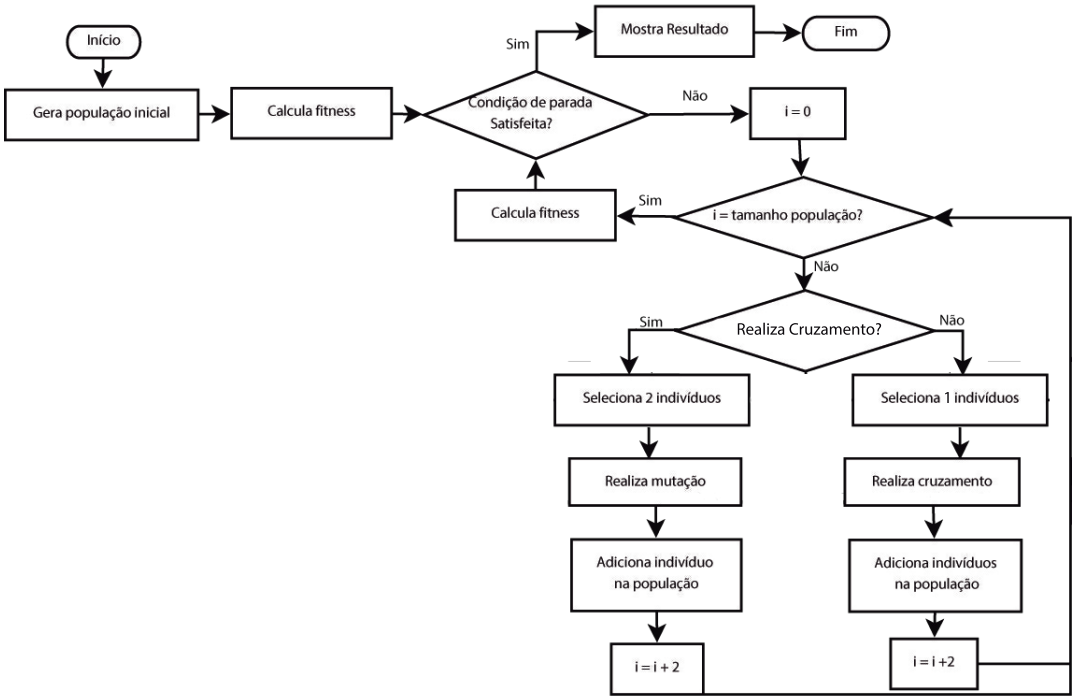


Figura 2.4. Fluxograma do Algoritmo Genético

- Passo 3 - Seleciona um indivíduo da população existente;
- Passo 4 - Realiza mutação nos indivíduos selecionados e adiciona os novos indivíduos na população. Logo após, realizamos o passo 5.
- Passo 5 - Avalia todos os indivíduos da população. Nessa fase, cada indivíduo é associado um valor de *fitness* (ou aptidão), ou seja, um valor que representa a capacidade do indivíduo em resolver um determinado problema.

Vale ressaltar que as probabilidades de mutação e cruzamento são parâmetros do algoritmo e devem ser determinadas pelo usuário do algoritmo. Além disso, a seleção é necessária para escolher os indivíduos sobre os quais serão aplicados os operadores genéticos. É realizada utilizando os valores de *fitness* (qualidade do classificador) e consiste em escolher aqueles que irão sofrer transformações e consequentemente compor a nova geração [Back, 1996]. Os principais métodos de seleção são:

- Seleção proporcional: Neste método, quanto maior for o *fitness* de um indivíduo, maior a chance de ser selecionado;

- Seleção por torneio: Esse método seleciona t indivíduos, sendo t o tamanho do torneio. O vencedor do torneio é aquele com maior valor de *fitness* entre os selecionados.

- Seleção Aleatória: A escolha dos pais ocorre de forma aleatória, sem a utilização de avaliação dos indivíduos.

Já a evolução da população em algoritmos evolucionários é obtida através da exploração de áreas mais promissoras do espaço. Esta exploração consiste na localização de indivíduos mais adaptados ao problema. A localização desses indivíduos ocorre através de operadores genéticos que causam modificações em indivíduos já conhecidos pela população. Os operadores mais conhecidos são:

- Mutação - A mutação consiste na criação de um novo indivíduo através de pequenas alterações aleatórias em indivíduos já existentes na população. Esta operação muda de forma aleatória alguns genes do indivíduo [Eiben & Smith, 2003];

- Cruzamento: Já no cruzamento selecionam-se dois indivíduos pais, e logo após, trocam-se genes entre esses pais. Existem vários operadores de cruzamentos disponíveis na literatura. No cruzamento uniforme avalia-se uma probabilidade de troca para cada gene dos indivíduo. No cruzamento de um ponto seleciona-se um ponto inicial e um ponto final do vetor e trocam-se esses sub-vetores entre os pais.

Um detalhe importante a ser tratado nestes algoritmos é a configuração de parâmetros. Os principais parâmetros de um AE são: tamanho da população, quantidade de gerações, taxa de cruzamento e taxa de mutação. Estes parâmetros normalmente são definidos através de gráficos que apresentam a evolução e convergência dos algoritmos ou através de testes experimentais.

2.2.1 Algoritmos Genéticos para Classificação

Como mencionado anteriormente, os AGs têm apresentado bons resultados para diversos problemas, incluindo a classificação [Vivekanandan & Nedunchezhian, 2011, 2010; Tanwani & Farooq, 2009; Mola & Miele, 2006; Kumar et al., 2011]. Esta seção foca nos diferentes tipos de representação de modelos utilizados por AGs, mais especificamente em representações por regras de decisão e binárias.

Freitas [2002] descreve uma representação onde cada gene de um indivíduo corresponde a uma regra de decisão do tipo se-então. Considere, por exemplo, uma

base de dados com características de pessoas (altura, peso, idade, comprimento do cabelo). Uma regra válida para este cenário pode ser definida como “se (altura > 1,75)"ou “(idade = 19)"então *classe 0*. Nesta abordagem, um indivíduo é composto por um conjunto de regras, e a classificação é realizada através dos resultados retornados para cada regra, após o seu processamento com os valores contidos na instância a ser classificada. Desta forma, após processado o indivíduo, a técnica retorna a qual classe a instância pertence.

Uma abordagem mais simples e com grande potencial para classificação de textos pode ser encontrada em Pietramala et al. [2008]. Eles utilizam uma representação semelhante a descrita na Figura 2.2, onde cada individuo é um vetor de bits. Cada bit representa uma característica (termo), e o valor 1 indica a presença desta característica e o valor 0 a ausência. Além disso, o vetor é dividido em duas partes, a primeira representa os atributos que devem aparecer na instância e a segunda parte os atributos que não devem aparecer na instância para que ela seja considerada de uma determinada classe.

Desta forma a classificação seria realizada por um modelo semelhante a: dado um documento *d*, este é da classe *c* se $(t_1 \in d \vee ... \vee t_n \in d) \wedge (t_{n+1} \notin d \wedge ... \wedge t_{n+m} \notin d)$ onde cada t_i é um termo presente na base de dados em questão.

O método proposto nesta pesquisa é baseado neste trabalho, porém com adaptações, visto que as bases de dados utilizadas por Pietramala et al. [2008] continham documentos com textos grandes e os dados do Twitter (foco desta dissertação) são textos curtos e com vocabulário extenso. Na próxima seção apresentamos alguns trabalhos com AE para fluxo de dados.

2.2.2 Algoritmos Evolucionários para mineração em fluxo de dados contínuos

Dentro do campo de pesquisa para fluxo de dados, alguns trabalhos foram encontrados utilizando Algoritmos Evolucionários. Folino et al. [2007] e Folino & Papuzzo [2010] apresentam algoritmos baseados em Programação Genética para mineração de fluxo de dados. Esta abordagem divide o conjunto de treino em vários subconjuntos, gera um classificador para cada subconjunto e em seguida constrói um modelo global, obtido pela agregação dos modelos locais provenientes de cada subconjunto. Desta forma, trabalham com um comitê de classificadores. Além disso, utilizam uma função baseada na dimensão fractal para determinar se ocorreram mudanças na distribuição dos dados, sendo assim, necessário atualizar os classificadores.

Já nos trabalhos de Vivekanandan & Nedunchezhian [2011, 2010] os classificadores são baseados em Algoritmos Genéticos. A proposta consiste em construir um modelo baseado em regras de forma incremental e utilizando poucos registros do conjunto de treinamento. Para isso, o conjunto de treinamento é dividido em n conjuntos distintos e para cada um desses n conjuntos é gerado um classificador, representado por um conjunto de regras. Tendo esses classificadores, combinam-se as regras geradas no primeiro passo para formar um conjunto de regras candidatas. As regras são então avaliadas no conjunto completo de treinamento, e aquelas cuja confiança atingir um limiar determinado pelo usuário, são determinadas como um conjunto de regras final.

Nestes trabalhos a atualização do modelo é realizada através de janela de tempo fixo, ou seja, a cada novo bloco de dados que chega o modelo é retreinado e as alterações nos indivíduos ocorre através de operadores padrões dos AGs. Os dados são armazenados de forma incremental, porém tomando cuidado com as classes que possuem poucos exemplos (eles controlam para que não sejam apagados todos os exemplos de uma determinada classe) e os dados são armazenados em janelas distintas.

Desta forma, as principais diferenças entre estes trabalhos e o proposto por esta pesquisa está na forma de representação dos dados e no fato que este trabalho retreina somente quando detectado alterações, enquanto Vivekanandan & Nedunchezhian [2011, 2010] retreinam o modelo em tempo fixo.

Até o presente momento foram apresentados alguns conceitos teóricos importantes para o entendimento da proposta de pesquisa, sendo o próximo capítulo dedicado a descrição da proposta.

Capítulo 3

Algoritmo Genético para Classificação em Fluxos de Dados Contínuos

As pesquisas de mineração em fluxo de dados contínuos têm se tornado atrativas devido a importância de suas aplicações e ao aumento significativo de fluxo de dados na Web. Nesta direção, este trabalho propõe a utilização de um algoritmo genético para classificar dados em fluxo contínuo. Como apresentado no Capítulo 2, o algoritmo deve ser modelado de forma a tentar resolver os três maiores desafios da literatura para fluxo de dados:

1. Quantos e quais os dados devem ser armazenados?
2. Quando devemos atualizar o classificador?
3. Como devemos realizar a atualização?

Uma das grandes motivações do uso de AG para este problema está relacionada a característica de evolução de soluções candidatas ao longo do tempo. Esta característica possibilita aos classificadores evoluir junto com os dados, ou seja, cada vez que o modelo é retreinado, as soluções criadas anteriormente são inseridas na nova população inicial e evoluídas juntamente com novas soluções, que podem inclusive ser representadas por vocabulários distintos.

Alguns elementos básicos dos algoritmos evolucionários devem ser definidos para o desenvolvimento desta pesquisa, entre eles: (i) como representar os indivíduos, (ii) como avaliar os indivíduos (função de *fitness*), e (iii) como adaptar os algoritmos genéticos para oferecer suporte a mineração em fluxo de dados.

Nesse capítulo, inicialmente apresentamos como o classificador é representado e uma visão geral do algoritmo proposto, e as seções seguintes descrevem a função de avaliação, e as adaptações do AG definidas nesse trabalho.

3.1 Representação dos indivíduos

Os classificadores poderiam ser representados por árvores de decisão, regras de classificação ou funções matemáticas [Bäck, 2002]. Este trabalho utiliza um modelo de regras com representação simplificada, onde o que importa é a presença ou não de um atributo na base. A ideia inicial do modelo foi proposta para classificação de textos em Pietramala et al. [2008], e aqui é utilizada com algumas alterações. A principal motivação para escolha dessa representação é sua simplicidade.

Figura 3.1. Indivíduo proposto.

Neste trabalho o indivíduo é representado por uma lista de termos dividida em 2 partes. A primeira é destinada aos atributos que devem aparecer na instância e a segunda aos atributos que não devem aparecer na instância para que esta seja considerada de uma determinada classe. A classificação é feita considerando a aparição de no mínimo x termos que devem estar presentes e no máximo y atributos do vocabulário que devem estar ausentes. Observando a Figura 3.1 podemos entender melhor como funciona um indivíduo. Imagine que nossa base contenha diversos *tweets* relacionados a um determinado jogador, e queremos classificar se o texto "fala mal" ou "fala bem". Considere que o indivíduo representado na Figura 3.1 é o nosso classificador, x é definido como 1 e y como 0 e a classe como "fala mal". Avaliando o dado que acaba de chegar com o seguinte texto "Este jogador tem talento" observamos que não temos nenhuma palavra do vetor *Termos Presentes* e uma palavra do vetor *Termos Ausentes*, logo, este texto é classificado como "*fala bem*", que é a classe complementar à fala mal. Note que a representação aqui apresentada funciona para problemas com duas classes (classificador binário). Além disso, os indivíduos não possuem termos iguais no seu vetor.

Para selecionarmos os termos que podem fazer parte do indivíduo, utilizamos o conceito de entropia [Tan et al., 2005; Freitas, 2002]. A entropia mede a variação ou diversidade na distribuição probabilística de um evento, ou seja, mede a quantidade

de informação de um determinado objeto. Quanto maior o grau da entropia maior é a desordem do conjunto e quanto menor, melhor a sua organização [Mitchell, 1997]. Considere um conjunto de entrada D (conjunto de treinamento) que pode ter c classes distintas e um atributo A presente neste conjunto de treinamento. A entropia de A considerando o conjunto D pode ser observada na equação 3.1.

$$Entropia(A) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{3.1}$$

onde p_i é a proporção de dados em D que possui o termo A e pertence a classe i. Vale ressaltar que quando um determinado termo aparece apenas em uma classe, atribuímos seu valor de entropia como 0, indicando que este pode ser um bom termo para compor o indivíduo. Desta forma, se um termo apareceu em 10 instâncias positiva e nenhuma negativa ele recebe o valor de entropia 0, da mesma forma que, se um termo apareceu em 40 instâncias positivas e nenhuma instância negativa ele recebe 0 de entropia. Apesar da intuição de que no segundo caso o termo seria melhor para compor o indivíduo, devido a uma maior probabilidade deste possuir maior organização, o primeiro exemplo pode indicar a aparição de um novo termo de qualidade no vocabulário. Desta forma, optou-se por deixar que o classificador realize essa escolha possibilitando que os dois termos disputem com igual probabilidade.

Após calcularmos a entropia para cada termo presente no conjunto de treinamento, realizamos uma seleção escolhendo apenas os termos com entropia menor que *maxEnt* (este deve ser definido pelo usuário na execução dos experimentos). Após selecionarmos todos estes termos, separamos os termos que aparecem mais na classe positiva dos que aparecem mais na classe negativa. Desta forma, criamos duas listas de vocabulário, a primeira contém os termos que podem compor o indivíduo na lista de termos que devem estar presentes e a segunda contém os termos que podem compor o indivíduo na lista de termos que devem estar ausentes.

Como o foco deste trabalho é a utilização de bases do Twitter, algumas adaptações e definições foram realizadas considerando que *tweets* tem tamanho máximo de 140 caracteres, mas o vocabulário das bases de dados é tão grande quanto os de documentos de texto em geral e mudam com frequência. O tamanho de cada lista de termos do indivíduo esta relacionado a quantidade de termos selecionados através da entropia, ou seja, a lista do indivíduo de termos presentes deve ser menor ou igual a quantidade de termos presentes selecionados e a lista de termos ausentes deve ser menor ou igual a quantidade de termos selecionadas para o vocabulário de termos ausentes. Além disso, sempre quando for detectada uma mudança a lista de vocabulários é atualizada, ou seja, a lista de palavras que podem compor um indi-

víduo é atualizada, considerando o conjunto de treinamento atual. Desta forma, é possível atualizar o vocabulário eliminando palavras que não são mais boas para a distinção e adicionar novas palavras.

3.2 Visão geral

A ideia principal do método proposto é aproveitar a característica de evolução dos algoritmos genéticos para evoluir o classificador junto com a evolução natural dos dados. Considerando os principais desafios apresentados na Seção 2, a técnica basicamente ataca cada um desses desafios da seguinte forma:

1. Quais dados devem ser armazenados e quais dados devem ser descartados? A técnica utiliza um método baseado em repositório.

2. Quando devemos atualizar o modelo? Foi utilizado o teste estatístico *Page-Hinkley* (PH) para detectar mudanças no desempenho dos classificadores.

3. Como devemos atualizar o modelo? O modelo é atualizado aproveitando os operadores de evolução dos Algoritmos Genéticos.

Algoritmo 1 Método Proposto

Entrada: TreinoInicial, $maxEnt$
Calcula entropia dos termos
Seleciona termos com entropia menor que $maxEnt$
Gera população inicial P
M = Evolui população P
enquanto chega nova instância x **faça**
    Classifica x
    Realiza manutenção do repositório de dados
    se atinge número máximo (m) de instâncias **então**
        Avalia desempenho do classificador
        se desempenho mudou de acordo com o teste de Page-Hinkley **então**
            Recalcula entropia dos termos
            Seleciona termos com entropia $< maxEnt$, atualizando o vocabulário
            Insere n indivíduos aleatórios em P
            M = Evolui população P
        fim se
    fim se
fim enquanto

No Algoritmo 1 apresentamos um pseudo-código com uma visão geral do método proposto. O algoritmo começa seu processamento recebendo um conjunto de treinamento inicial, do qual são armazenadas algumas estatísticas. As informações armazenadas referem-se a distribuição de atributos nas classes (neste trabalho, termos e atributos são usados como sinônimos), o grau de pureza para cada atributo

(nesta pesquisa foi utilizada a entropia para cada termo), média da quantidade de atributos por instância e distribuição das instâncias por classe. Após a análise dessas informações, os atributos mais relevantes são selecionados para poder fazer parte dos classificadores. Esta seleção é baseada nos termos com menor entropia, ou seja, que apresentam melhor separação entre as classes. Para isso o usuário deve informar um parâmetro $maxEnt$ e o método seleciona todos os termos com entropia menor que $maxEnt$. A principal motivação para esta seleção é que nem todos os atributos são relevantes para separar duas classes. Além disso, ela diminui o tamanho do espaço de busca por soluções dos algoritmos genéticos.

O próximo passo do método cria uma população inicial. Geralmente, nenhuma heurística é utilizada, sendo a criação conduzida de forma aleatória. Entretanto, nesta pesquisa, como conhecemos o valor de entropia para cada termo do conjunto de treinamento inicial, utilizamos esse dado para proporcionar maior probabilidade de selecionarmos um termo com menor valor de entropia para compor o indivíduo. Para isso, ao escolher um atributo para compor o indivíduo selecionamos x termos aleatórios, e o que possuir menor entropia é selecionado para fazer parte do indivíduo. Após criar a primeira população, esta passará por um processo de evolução como apresentado no pseudo-código.

Até este ponto o algoritmo é o mesmo utilizado em um processo de classificação estático. A diferença de um algoritmo não estacionário encontra-se nas próximas etapas. Vale ressaltar que os passos definidos abaixo são realizados durante todo o processo de chegada de informação. Assim, para cada nova instância:

1. Classifica a instância com o classificador já encontrado;

2. Atualiza o repositório de dados, ou seja, atualiza o conjunto de treinamento com a classe real da instância (mais detalhes na Seção 3.5);

3. Verifica se o número máximo de instâncias no repositório foi atingido. Em caso positivo, verificamos o desempenho do classificador através de um teste estatístico denominado Page-Hinkley (mais detalhes na seção 3.5). Se o sistema detectar uma alteração no desempenho do classificador os próximos passos são realizados.

4. Atualizamos o vocabulário;

5. A população evoluída no primeiro passo é incrementada com a inserção de novos indivíduos, e um novo processo de evolução se inicia, gerando um

novo classificador atualizado com os novos dados. Desta forma, o sistema esta pronto para classificar novas instâncias voltando ao passo 1.

Vale ressaltar que o classificador selecionado para realizar a classificação dos novos indivíduos é sempre o que possui a maior *fitness*.

Nesta seção foi apresentado o funcionamento geral da técnica proposta, sendo as próximas seções destinadas a apresentar mais detalhes de algumas partes essenciais do algoritmo.

3.3 Qualidade do indivíduo

Para avaliar o quão bom é um indivíduo, ou seja, cada classificador, é necessário escolher uma medida que reflita sua capacidade de classificação. Várias medidas podem ser utilizadas. Entre elas:

1. Precisão (P)- Porcentagem de instâncias que o modelo é capaz de classificar corretamente como positiva dentre todas as que foram classificadas como positivas.

2. Revocação (R) - Porcentagem de instâncias que o modelo é capaz de classificar corretamente como positiva dentre todas as que realmente são positivas.

3. F1 (F1)- Média harmônica entre Precisão e Revocação, conforme definido na Equação 3.2. Esta forma de avaliação é mais indicada em casos de bases com classes desbalanceadas, evitando um classificador que seja muito bom em classificar uma classe, mas não seja capaz de classificar a outra.

$$F1 = (2 \times R \times P)/(R + P) \tag{3.2}$$

No caso desta pesquisa, como na maior parte do tempo trabalhamos em bases com classes altamente desbalanceadas, optamos pela utilização do F1 (média do F1 em todas as classes).

3.4 Evolução da População

A evolução da população em AG é obtida através da localização de indivíduos mais adaptados ao problema. A localização desses indivíduos ocorre através da seleção, e se prolifera através do uso de operadores genéticos que causam modificações em indivíduos já conhecidos pela população. Vale ressaltar aqui que a

seleção dos indivíduos nesse trabalho é realizada através de torneio (Seção 2.2). Os operadores utilizados são:

- Cruzamento - No cruzamento selecionam-se dois indivíduos pais, trocam-se genes desses pais, formando-se assim dois filhos (indivíduos) com as características dos pais [Eiben & Smith, 2003]. Neste trabalho o cruzamento ocorre da seguinte forma: para cada gene (termo) do primeiro indivíduo selecionado, avaliamos uma probabilidade de realizarmos a troca deste. Desta forma, para cada gene, selecionamos um número aleatório entre 0 e 1. Caso o número seja maior que o valor fixo de cruzamento, determinado pelo usuário, selecionamos um gene aleatoriamente do outro indivíduo selecionado e trocamos esses genes entre os pais. Vale ressaltar que essas trocas ocorrem apenas entre os vetores semelhantes, ou seja, o vetor "*Termos Presentes*" do primeiro pai só realiza trocas com o vetor "*Termos Presentes*" do segundo pai e o vetor "*Termos Ausentes*" do primeiro pai só realiza trocas com o vetor "*Termos Ausentes*" do segundo pai. Além disso, não colocamos termos repetidos nos vetores.

- Mutação: Já a mutação em AG é a criação de um novo indivíduo através de pequenas alterações aleatórias em indivíduos já existentes na população. Existem várias técnicas de operadores de mutação disponíveis na literatura. Neste trabalho, para cada gene é selecionado um número aleatório entre 0 e 1, caso o número seja maior que o valor fixo de mutação determinado pelo usuário, este gene é alterado. Esta alteração é realizada removendo o atributo do indivíduo e selecionando um atributo novo para substituí-lo.

Para ambos operadores, em casos onde o indivíduo já possui o gene a ser trocado ou adicionado, este não é inserido novamente. Ou seja, os indivíduos não possuem termos iguais em seus vetores de termos.

3.5 Adaptações para tratar de fluxos de dados

Como apresentado anteriormente, a ideia principal desta pesquisa consiste em aproveitar a característica de evolução dos algoritmos genéticos para evoluir classificador junto com a evolução dos dados. Considerando os principais desafios apresentados na Seção 2, as próximas seções apresentam como cada problema é tratado pelo método.

3.5.1 Quais os dados devem ser armazenados e quais dados devem ser descartados?

Em cenários semelhantes ao Twitter, onde pessoas de várias regiões distintas são responsáveis pelo conteúdo disponibilizado, e os assuntos tratados são altamente dinâmicos, é comum ocorrer modificações no vocabulário ao longo do tempo [Bifet & Kirkby, 2009]. Desta forma, pretende-se trabalhar com um vocabulário dinâmico. Para isso, o método utiliza o conceito de repositório de dados (Seção 2.1.1). Assim, a cada novo dado que chega atualizamos o nosso conjunto de treinamento tentando capturar novos termos e esquecer termos mais antigos ou com informações desatualizadas (que não são mais capazes de realizar a diferenciação entre classes). Como definido na Seção 3.2, a seguinte abordagem foi proposta para manter os dados e o modelo atualizado: se o tamanho máximo do conjunto de treinamento (repositório) não foi atingindo, apenas inserimos a nova instância neste conjunto. Se o tamanho máximo foi atingido, verificamos se existem instâncias iguais a nova instância, ou seja, instâncias com os mesmos termos. Em caso positivo, esta é descartada e atualizamos o tempo de chegada da instância igual, presente no conjunto de treinamento, para o tempo atual. Em caso negativo, inserimos essa instância e removemos a instância mais antiga do conjunto de treinamento. Esta atualização é importante para que possamos manter termos mais recentes no vocabulário e esquecer os desatualizados. Entretanto, eliminamos apenas as instâncias idênticas visto que instâncias parecidas podem indicar que determinados termos são bons para a classificação. Vale ressaltar que, nessa primeira versão, o tamanho do repositório é definido pelo usuário do método.

3.5.2 Quando devemos atualizar o modelo?

Para determinarmos quando atualizar o modelo utilizamos uma abordagem baseado na qualidade do classificador (Seção 2.1.2). Para isso foi utilizado um teste estatístico, conhecido como Teste *Page-Hinkley* (PH) [Gama, 2010]. Este teste monitora a diferença entre duas variáveis m_t e M_t, onde a variável acumulativa m_t é definida como a diferença acumulada entre o valor observado (no nosso caso a *fitness* do indivíduo) e sua média até o presente momento t:

$$m_t = \sum_{t=1}^{T} (p_t - \bar{p_t} - \delta) \tag{3.3}$$

onde,

$$\bar{p_t} = \frac{1}{t}\sum_{i=1}^{t} p_i \tag{3.4}$$

o parâmetro δ corresponde a magnitude da mudança que não deve ocasionar em um alarme. A segunda variável M_t é definida como o valor mínimo observado de m_t.

$$M_t = min(m_i = 1...t) \tag{3.5}$$

A alteração da distribuição é informada quando a diferença das duas variáveis é superior a um dado limiar $\lambda : m_t - M_t > \lambda$. O valor do parâmetro λ determina a taxa de alarmes falsos e é informado pelo usuário. Um valor alto implica em menos alarmes falsos, porém algumas mudanças podem ser descartadas. Desta forma, em caso de mudanças na *fitness* do indivíduo (considerando a média das *fitness* anteriores e um limiar de aceitação), este ocasiona em um alarme informando que devemos realizar atualização no classificador. Um pseudo-algoritmo deste teste pode ser analisado no Algoritmo 2.

Algoritmo 2 Pseudo-código do Teste *Page-Hinkley*

Entrada: p_t, n_t
$media \leftarrow ((n_t - 1) * media + p_t)/n_t$
$m_t \leftarrow m_t + p_t - media - \delta$
se $m_t \leq M_t$ **então**
    $M_t \leftarrow m_t$
fim se
se $m_t - M_t > \lambda_w$ **então**
    $estado \leftarrow mudou$
senão
    $estado \leftarrow estacionario$
fim se
retorna $estado$

O algoritmo recebe como entrada a *fitness* atual (p_t) e a quantidade de vezes que o teste já foi verificado (n_t). Como passo inicial, calcula-se a média geral das *fitness* até o presente momento, e em seguida é calculada a diferença acumulada entre o valor observado e a média até o presente momento. No próximo passo, verificamos qual o valor mínimo observado de m_t e atribuímos a M_t. Como passo final, retornamos um sinal de mudança quando a diferença entre m_t e M_t é maior que o limiar λ, informado pelo usuário do sistema, ou um sinal que não ocorreu mudança caso contrário.

3.5.3 Como devemos atualizar o modelo?

Como mencionado no começo deste capítulo o modelo é atualizado aproveitando os operadores de evolução dos Algoritmos Genéticos. Desta forma, sempre que detectado uma mudança, recalculamos a entropia para cada termo, o método atualiza o vocabulário de termos que podem compor a lista de termos presentes e ausentes de um indivíduo e cria n classificadores aleatórios que são adicionados a população atual e esta evolui através dos operadores de evolução dos algoritmos genéticos (cruzamento e mutação). A criação desses novos indivíduos aleatórios ajuda a introduzir os novos termos descobertos e a diversificar a população já existente. Além disso, quando realizamos o operador de mutação aos indivíduos, eles também podem selecionar os termos que apareceram recentemente no conjunto de treino. Desta forma, a técnica consegue trabalhar com um vocabulário dinâmico acompanhando as alterações dos termos na base de dados.

Capítulo 4

Resultados Experimentais

Esse capítulo avalia a eficácia e eficiência do GA proposto quando comparado a outros métodos estado da arte na literatura.

4.1 Base de Dados

Nesta seção apresentamos uma pequena caracterização das bases de dados utilizadas para avaliação da técnica proposta. Foram realizados experimentos com quatro bases de dados do *Twitter*. Essas bases foram desenvolvidas pelo Observatório da Web na Universidade Federal de Minas Gerais (UFMG) e apresentam característica temporal. Através destas bases foi possível avaliar a técnica em cenários onde a distribuição dos dados muda de diferentes formas no decorrer do tempo (diferentes tipos de *concept drift*). Na Tabela 4.1 são apresentadas as seguintes características para cada base: número de instâncias, número de classes, número de termos, média de termos por instância (mt/i), número de blocos e média de termos por bloco (mt/b).

A seguir é detalhada cada coleção de dados e o evento a que ela se refere. Para cada uma das bases são apresentados gráficos que evidenciam a ocorrência de dife-

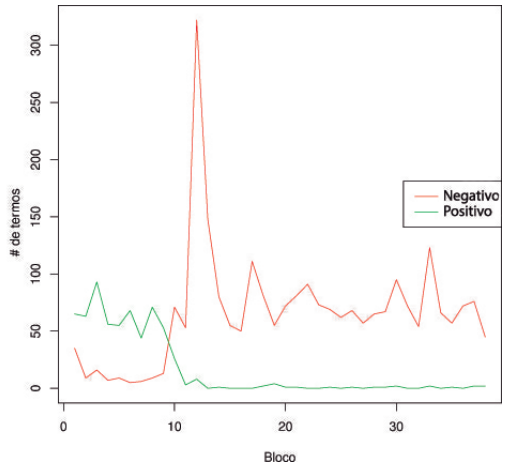
Tabela 4.1. Base de Dados do *Twitter*.

Base de Dados	*# Instâncias*	*# Classes*	*# Termos*	*mt/i*	*# Blocos*	*mt/b*
Futebol	3.127	2	4.261	10	38	322.60
Futebol Inglês	1.461	2	2.576	11	14	358.85
Eleições	66.626	2	36.245	8	100	1278.35
Dengue	119	2	604	7	5	165.8

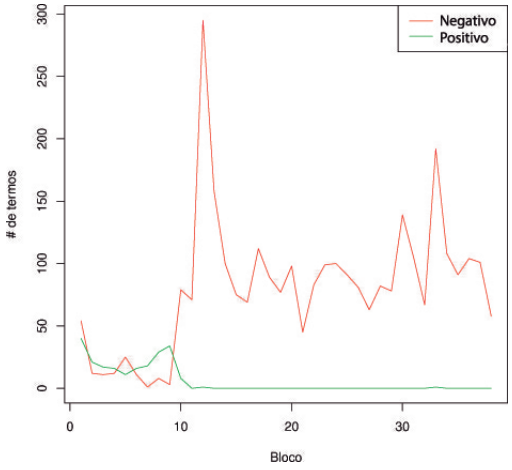
rentes mudanças na distribuição das classes nas coleções analisadas. Cada base foi dividida em vários blocos e apresentamos, respectivamente, um gráfico que apresenta a distribuição das classes para cada bloco de dados, quantidade de termos com entropia menor que 0.1 para cada bloco, número de termos distintos para cada bloco e quantidade de *tweets* com determinado número de termos. Essa divisão das bases foi determinada tentando criar blocos de dados que possibilitem a análise da mudança ao longo do tempo. Um fato importante a ser analisado nestes gráficos é a maneira como a seleção de termos utilizada para inicializar os indivíduos do AG, utilizando entropia, acompanha os gráficos de distribuição entre as classes. Vale ressaltar aqui, que as bases *Futebol*, *Futebol Inglês* e *Eleições* foram retiradas do trabalho Silva [2012] e a base *Dengue* do trabalho Gomide et al. [2012]. Além disso, foi realizado um pré-processamento na base onde foram desconsiderados os *stopwords* (palavras comuns sem significado relevante como preposições, pronomes, etc.)[Tan et al., 2005].

- **FUTEBOL e FUTEBOL INGLÊS** - Contém dados de um jogo polêmico entre Brasil e Holanda na Copa do Mundo de 2010 realizado no dia 02 de Julho. As mensagens estão relacionadas a um jogador específico, chamado Felipe Melo, que teve participação decisiva na partida. O jogador foi fundamental para o primeiro gol do Brasil na partida, mas logo após foi autor de um gol contra o Brasil e minutos depois expulso do jogo. Desta forma, os dados refletem a opinião dos torcedores em relação ao jogador durante o tempo da partida (Positivo ou Negativo). A base foi rotulada manualmente, e as Figuras 4.1 e 4.2, apresentam a distribuição das classes, distribuição dos termos selecionados com entropia menor que *0.1* para cada bloco, quantidade de termos por bloco e quantidade de *tweets* relacionados a um determinado número de termos. Para geração dos gráficos a base foi dividida em intervalos de 5 minutos, o que tornou possível observar a reação dos torcedores no momento do gol inicial aos 10 minutos do primeiro tempo, no momento do gol contra aos 8 minutos do segundo tempo e no momento em que o jogador foi expulso faltando 20 minutos para o fim da partida.

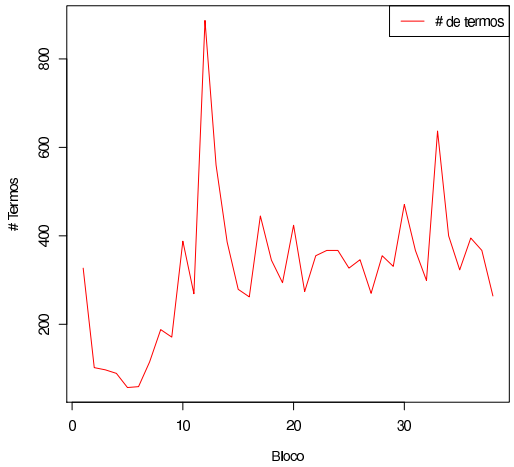
  Considerando as duas bases relacionadas ao jogador Felipe Melo (Português e Inglês) podemos observar uma diferença na forma em que ocorrem as mudanças na distribuição de classes (Figuras 4.1(a) e 4.2(a)). A base em português (de agora em diante referenciada como Fut1) apresentou picos de alterações mais atenuados durante o jogo. No começo é possível observar uma maior porcentagem de mensagens positivas em relação ao jogador uma vez que ele



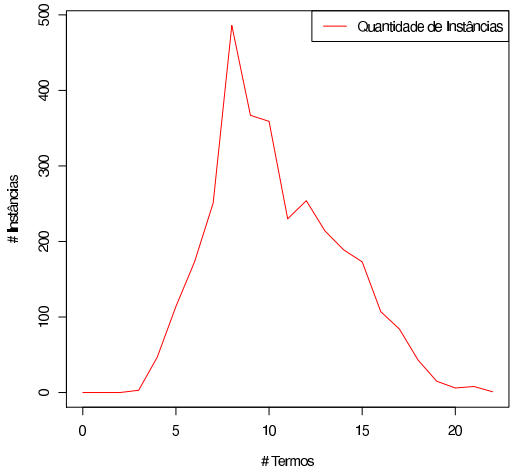
(a) Distribuição de Classes ao longo do tempo.



(b) Distribuição de termos selecionados com entropia menor que *0.1* ao longo do tempo.



(c) Número de Termos distintos para cada bloco.



(d) Quantidade de Tweets relacionados a um número de termos.

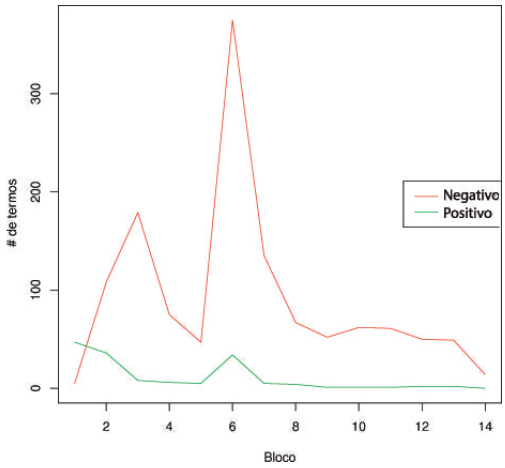
Figura 4.1. Características da base Futebol em Português

contribuiu para o primeiro gol a favor da seleção Brasileira. Contudo, após um tempo, houve um aumento de mensagens negativas. Essa mudança reflete o sentimento das pessoas a partir do momento em que o jogador faz um gol contra e posteriormente é expulso. O sentimento negativo tende a se intensificar no final da partida, isso porque o Brasil foi eliminado da copa nessa ocasião e a responsabilidade da derrota foi atribuída ao jogador. Já considerando a base em Inglês (de agora em diante referenciada como Fut2) é possível perceber que as alterações são mais suaves, ao mesmo tempo em que muitas pessoas se mostram felizes a respeito da derrota do Brasil, outras se declaram tristes, ou vice-versa. Isso pode ser explicado porque as pessoas que postam mensagem em inglês apresentam maior diversidade e não apresentam um sentimento definitivo em relação ao Brasil, desta forma, o sentimento pode variar de maneira inconsistente.

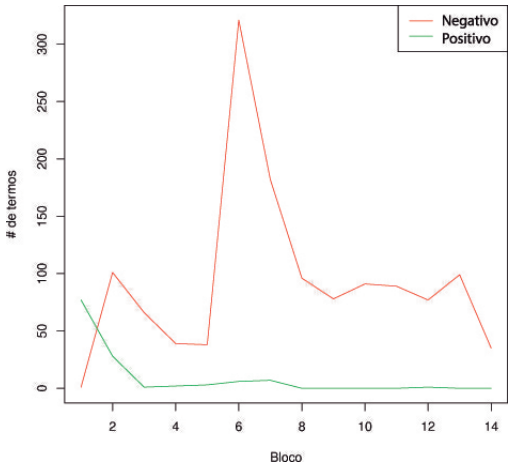
Estes sentimentos diferenciados em ambas as bases (Fut1 e Fut2) possibilitaram a análise da técnica para diferentes tipos de mudanças de conceitos.

- **ELEIÇÕES** - Esta base é referente a campanha eleitoral à presidência do Brasil no ano de 2010. Os dados foram coletados no período de Junho a Outubro. Nesta base, foram monitorados *tweets* relacionados a candidata Dilma Rousseff, que utilizou o *Twitter* como uma das principais fontes de informação para seus eleitores. Nestas eleições Dilma ganhou com 56 por cento dos votos. As mensagens foram classificadas como negativas ou positivas em função da candidata Dilma. Este sentimento de aprovação varia fortemente durante este período devido a várias polêmicas e ataques políticos. Podemos observar este comportamento no gráfico da Figura 4.3(a), onde é visível algumas alterações próximas aos blocos 5, 20,50, 70 e 90. Para esta base as mensagens no fluxo chegam em uma taxa de 0.02 mensagens por segundo. Como esta base não apresenta o horário exato da chegada de cada *tweet*, e sabemos que os dados estão em ordem cronológica, ela foi dividida em 100 blocos distintos de tamanhos iguais. Além disso, o gráfico da Figura 4.3(b) mostra que a seleção de termos com entropia menor que 0.1 acompanha o gráfico de distribuição das classes.

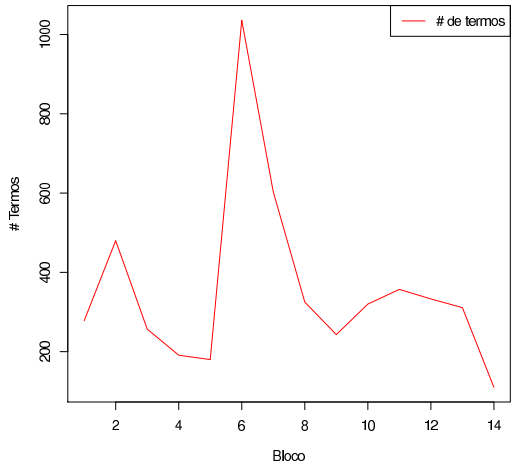
- **DENGUE** - Nesta base foram coletados *tweets* que apresentaram os termos *dengue* e *aedes aegypti*. Entre os *tweets* podem ser identificados temas como campanhas, ironia, experiência pessoal (tweets que informam que uma pessoa ou algum conhecido está com dengue), entre outros. Nesta pesquisa, a



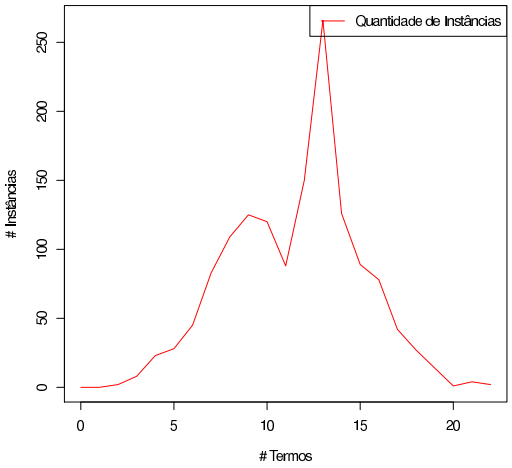
(a) Distribuição de Classes ao longo do tempo.



(b) Distribuição de termos selecionados com entropia menor que *0.1* ao longo do tempo.

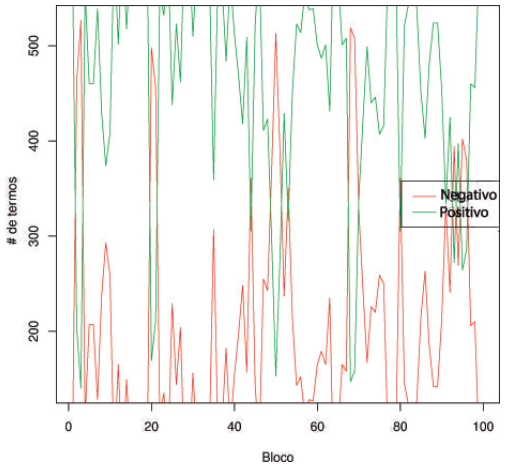


(c) Número de Termos distintos para cada bloco.

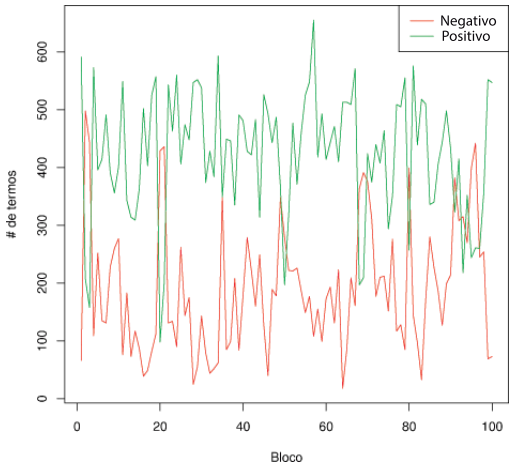


(d) Quantidade de Tweets relacionados a um número de termos.

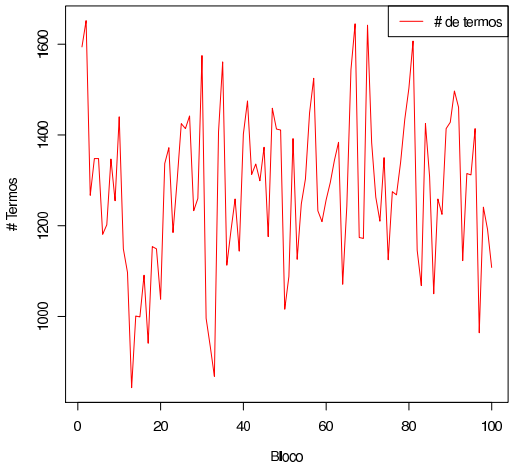
Figura 4.2. Características da base Futebol em Inglês



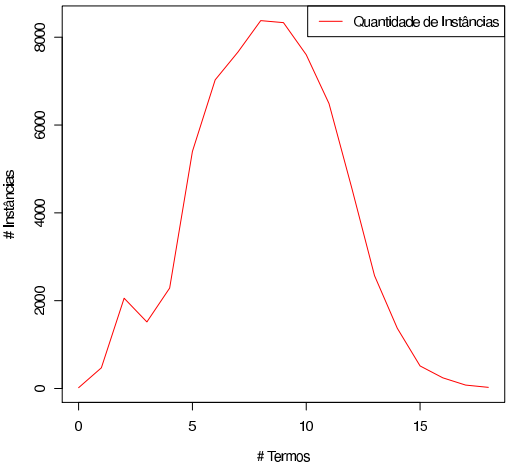
(a) Distribuição de Classes ao longo do tempo.



(b) Distribuição de termos selecionados com entropia menor que *0.1* ao longo do tempo.



(c) Número de Termos distintos para cada bloco.



(d) Quantidade de Tweets relacionados a um número de termos.

Figura 4.3. Características da base Eleições

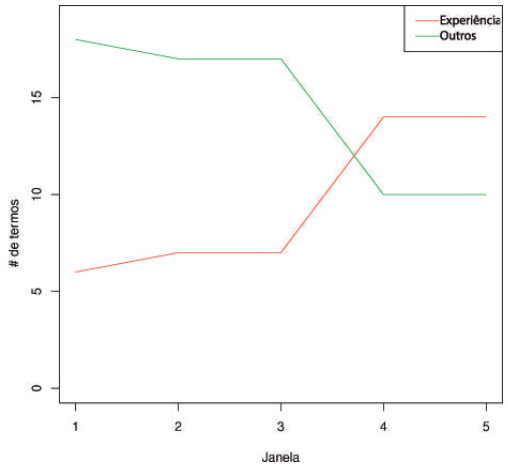
base foi utilizada para aprender um modelo de classificação que possibilite a caracterização de um novo *tweet* como experiência pessoal ou não. Essa distinção é importante porque permite, juntamente com a localização geográfica do usuário que postou a mensagem, detectar focos da doença e gerar modelos de propagação. Esta caracterização pode ajudar no combate a doença e planejamento de ações pelo governo e agentes de saúde. Para análise da base, ela foi dividida em 5 blocos de tamanhos iguais. A Figura 4.4 apresenta as características da base. Na Figura 4.4(a) foi possível visualizar uma alteração nos dados entre os blocos 3 e 4, onde houve um crescimento nos relatos de casos de dengue pelo *Twitter*. Além disso, uma característica interessante desta base pode ser observada no gráfico da Figura 4.4(b), onde não foi selecionado nenhum termo para o vetor de Termos Ausentes para todos os blocos. Isto demonstra uma maior organização nos termos que devem estar ausentes no *tweet*.

Esta seção apresentou algumas características das bases utilizadas nesta pesquisa, sendo a próxima seção destinada a apresentar a avaliação experimental realizada sobre estas bases.

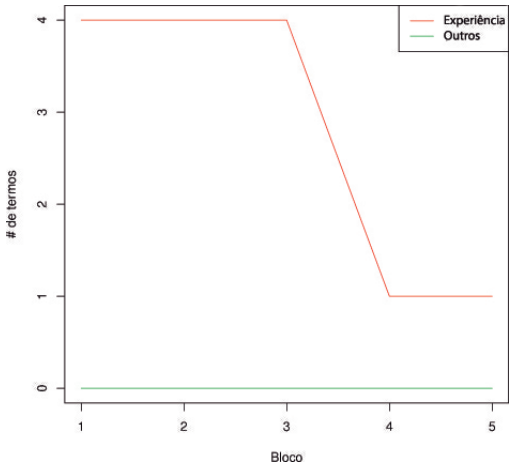
4.2 Avaliação Experimental

Para avaliar o desempenho do método proposto foram realizados experimentos sobre as 4 bases do Twitter apresentadas na Seção 4.1. Além disso, utilizamos dois algoritmos consolidados na literatura de Fluxos de Dados para compararmos os resultados. Os classificadores utilizados para comparação foram: *Multinomial Naive Bayes* (MNB) e *Hoeffding Tree* (HT). Esta escolha foi motivada pela adaptação destes dois métodos para trabalhar com bases textuais como no caso do *Twitter* e ao fato de ambos serem adaptados a fluxo de dados. Embora os trabalhos de Vivekanandan & Nedunchezhian [2011, 2010] sejam também um algoritmo de grande importância nessas comparações, eles serão considerados em trabalhos futuros, uma vez que as implementações não estão disponíveis.

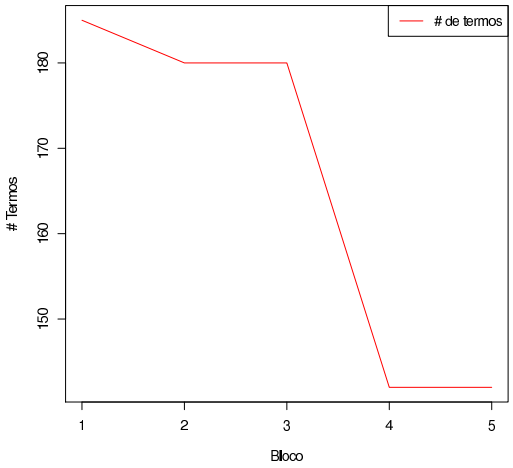
Para execução destes dois algoritmos utilizamos um ambiente de software disponibilizado na internet, denominado *Massive Online Analysis* (MOA) [Bifet & Kirkby, 2009]. Este ambiente é utilizado para implementação de algoritmos e execução de experimentos para aprendizagem em fluxos de dados e contém uma coleção de métodos online e offline já implementados. Algumas alterações na configuração dos algoritmos foram testadas, porém os melhores resultados foram alcançados com as configurações padrões da ferramenta. Vale ressaltar aqui que a técnica utilizada



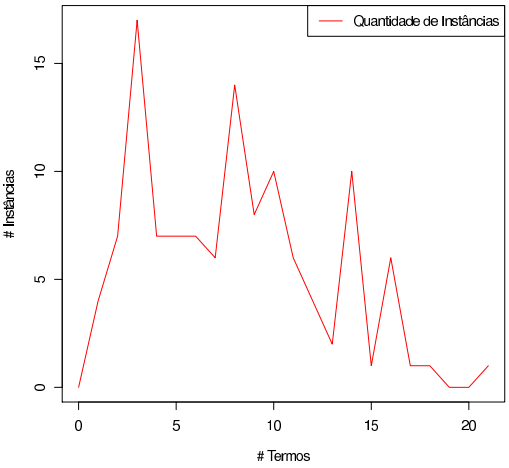
(a) Distribuição de Classes ao longo do tempo.



(b) Distribuição de termos selecionados com entropia menor que *0.1* ao longo do tempo.



(c) Número de Termos distintos para cada bloco.



(d) Quantidade de Tweets relacionados a um número de termos.

Figura 4.4. Características da base Dengue

para avaliar a qualidade de um classificador foi o F1 (Seção 3.3) devido ao grande desbalanceamento das bases de dados do *Twitter*.

Para regular os parâmetros tradicionais de AG e os relacionados a fluxo de dados foram realizados experimentos preliminares. Os resultados podem ser encontrados na Seção 4.2.1 e uma comparação com alguns algoritmos consolidados na literatura é realizada na Seção 4.2.2.

4.2.1 Análise de Parâmetros.

O algoritmo de classificação recebe como entrada, além da base de dados, um conjunto de parâmetros de configuração. Nesta seção, será feita uma análise sobre os principais parâmetros do algoritmo, tanto relacionados ao AG padrão como ao fluxo de dados, de modo a investigar a qualidade das soluções geradas de acordo com a existência de possíveis variações nos valores destes parâmetros. Os parâmetros relacionados ao AG padrão investigados foram:

- Tamanho da população (P) - Quantos indivíduos (possíveis soluções para o problema) formam a população.
- Quantidade de Gerações (G) - Quantidade de vezes que a população é evoluída.
- Probabilidade de Mutação (M) - Probabilidade com que a mutação é realizada.
- Probabilidade de Cruzamento (C) - Probabilidade com que o cruzamento é realizado.

Já os parâmetros relacionados ao fluxo de dados foram:

- Entropia máxima (EM) - determina o valor máximo de entropia que os termos devem possuir para poderem ser candidatos a participar de um indivíduo.
- Número termos presentes (TP) - quantos termos no mínimo a instância deve ter do vetor termos presentes.
- Número termos ausentes (TA) - quantos termos no máximo a instância deve ter do vetor termos ausentes.
- Tamanho treinamento (TT) - tamanho do treinamento no momento em que detectamos uma mudança.

- Número instâncias mínimas para verificar mudança (NIM) - quantidade mínima de instâncias de teste necessárias para verificar se ocorreram mudanças.
- λ - determina a taxa de alarmes falsos e é informado pelo usuário na detecção de mudanças.

A ordem de avaliação e os valores avaliados para cada parâmetro foram:

1. Quantidade de Gerações (G) - 50, 100, 200, 500, 1000, 2000
2. Tamanho da população (P) - 10, 50, 100, 200
3. Taxa de mutação (M) - 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
4. Taxa de cruzamento (C) - 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
5. Entropia máxima (EM) - 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
6. Número de termos presentes (TP) - 1, 2, 3, 4, 5,6
7. Número de termos ausentes (TA) - 0, 1, 2, 3, 4, 5
8. Tamanho conjunto de treinamento (TT) - 5, 10, 20, 50, 100
9. Número instâncias mínimas para verificar mudança (NIM)- 5, 10, 20, 30
10. λ - 0.0, 2.0, 5.0, 10.0

Os parâmetros foram investigados separadamente, ou seja, para cada parâmetro investigado foram fixados os valores dos outros parâmetros. Realizamos 15 execuções do algoritmo para cada configuração. Calculou-se então a média e o desvio padrão da qualidade das soluções geradas para podermos escolher a configuração mais adequada dos parâmetros. Para inicializarmos os experimentos, consideramos que um *tweet* somente será classificado em determinada classe se possuir pelo menos um termo do vetor Presentes (TP=1) e nenhum termo do vetor Ausentes (TA=0). Além disso, descartamos a pré-seleção dos termos através da entropia, desta forma, qualquer termo pode compor um indivíduo (EM = 1). Os valores iniciais dos outros parâmetros foram escolhidos com execuções preliminares do algoritmo, sendo um refinamento de cada parâmetro realizado ao longo dos experimentos.

Assim, definimos os parâmetros iniciais como:

1. Quantidade de Gerações = 50
2. Tamanho da população = 10

3. Taxa de mutação = 0.5
4. Taxa de cruzamento = 0.5
5. Entropia máxima = 1
6. Número mínimo de termos presentes = 1
7. Número máximo de termos ausentes = 0
8. Tamanho do conjunto de treinamento = 10
9. Número instâncias mínimas para verificar mudança = 5
10. λ = 0.0

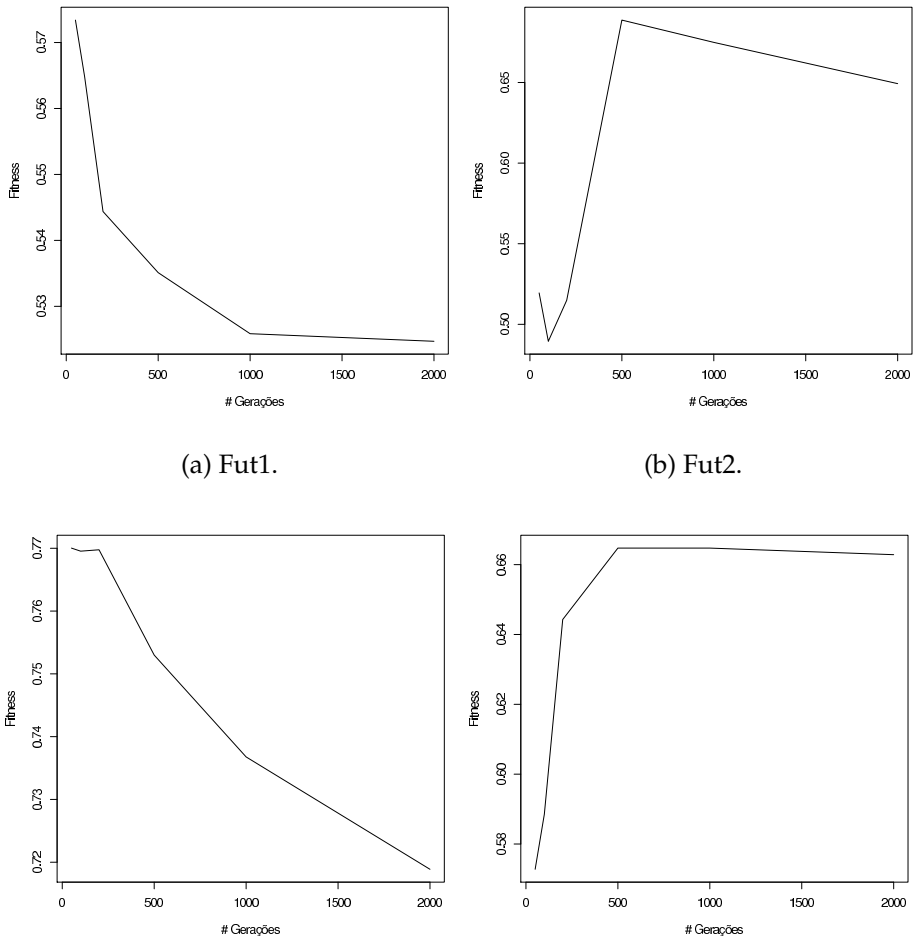
Seguimos com uma avaliação dos parâmetros para cada base. Para analisarmos os gráficos referentes ao AG padrão é importante observarmos, para cada parâmetro, que:

- População - Uma população pequena pode diminuir o desempenho dos classificadores, visto que fornece uma pequena cobertura do espaço de soluções do problema. Já uma população maior fornece uma cobertura mais representativa do domínio de soluções, entretanto requer maiores recursos computacionais ou que o algoritmo trabalhe por um período de tempo muito maior (o que não é uma boa escolha, dado que o desempenho é fundamental para o bom funcionamento de algoritmos que trabalham com fluxo de dados);

- Gerações - Poucas gerações podem não ser suficientes para encontrar um bom indivíduo enquanto várias gerações tendem a especializar os indivíduos no conjunto de treinamento;

- Cruzamento - Quanto mais alta a taxa de cruzamento, mais rápido serão introduzidas novas estruturas na população. Porém, se considerarmos uma taxa muito alta podemos perder estruturas com boas aptidões e valores de cruzamento baixo podem tornar o algoritmo lento;

- Mutação - Uma taxa de mutação muito alta tende a tornar o algoritmo essencialmente aleatório e uma taxa baixa pode indicar que uma dada posição fique estagnada em um valor;



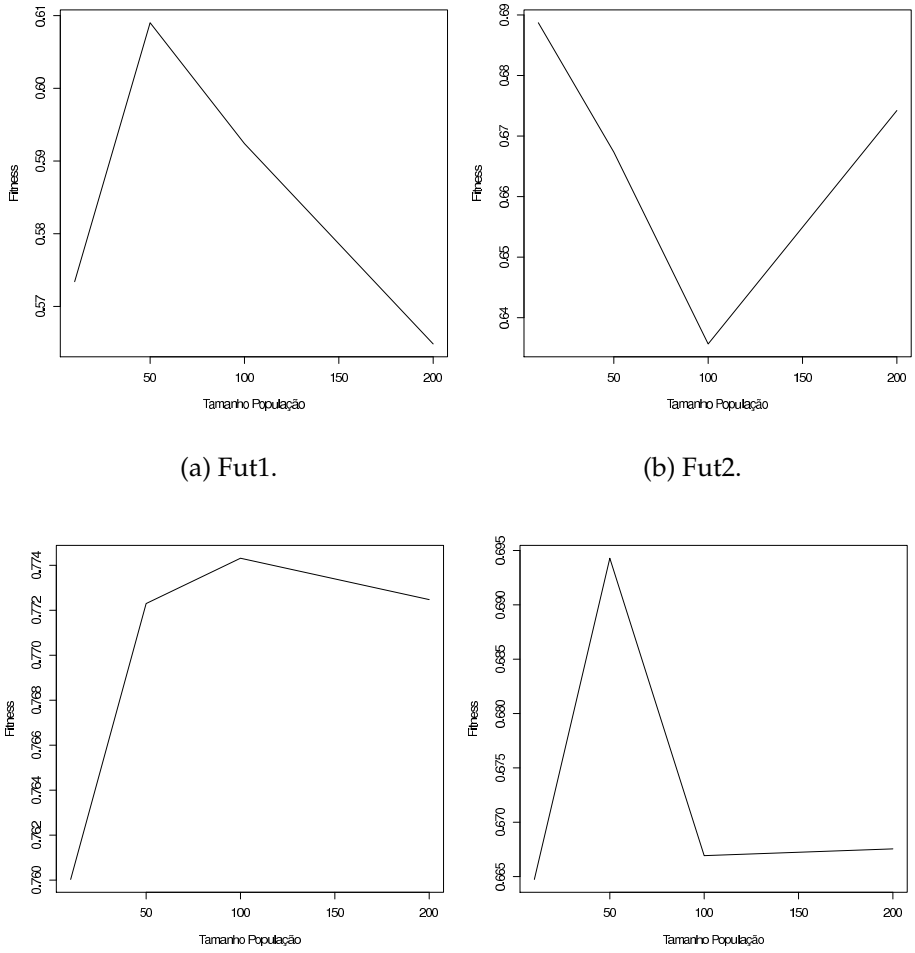
(a) Fut1. (b) Fut2.

(c) Eleições. (d) Dengue.

Figura 4.5. Resultado para alterações no número de gerações para cada base.

Nas Figuras 4.5, 4.6, 4.7 e 4.8 podemos analisar os gráficos com resultados de *fitness* (medida F1) para alterações nos parâmetros relacionados ao AG em todas as bases. A Figura 4.5 apresenta as alterações causadas com a modificação do parâmetro quantidade de gerações e a Figura 4.6 apresenta as alterações causadas pela modificação do parâmetro tamanho da população. Já as Figuras 4.7 e 4.8 apresentam as alterações causadas com a modificação dos parâmetros taxa de cruzamento e taxa de mutação, respectivamente.

Podemos observar através dos resultados que em relação a quantidade de gerações nas bases Eleições e Fut1 um número de gerações pequeno foi suficiente, e na medida em que aumentamos as gerações o método tende a especializar no conjunto de treinamento perdendo qualidade no conjunto de teste. Na base dengue a medida que aumentamos o número de gerações o resultado tende a estabilizar, não ocor-



(a) Fut1. (b) Fut2.

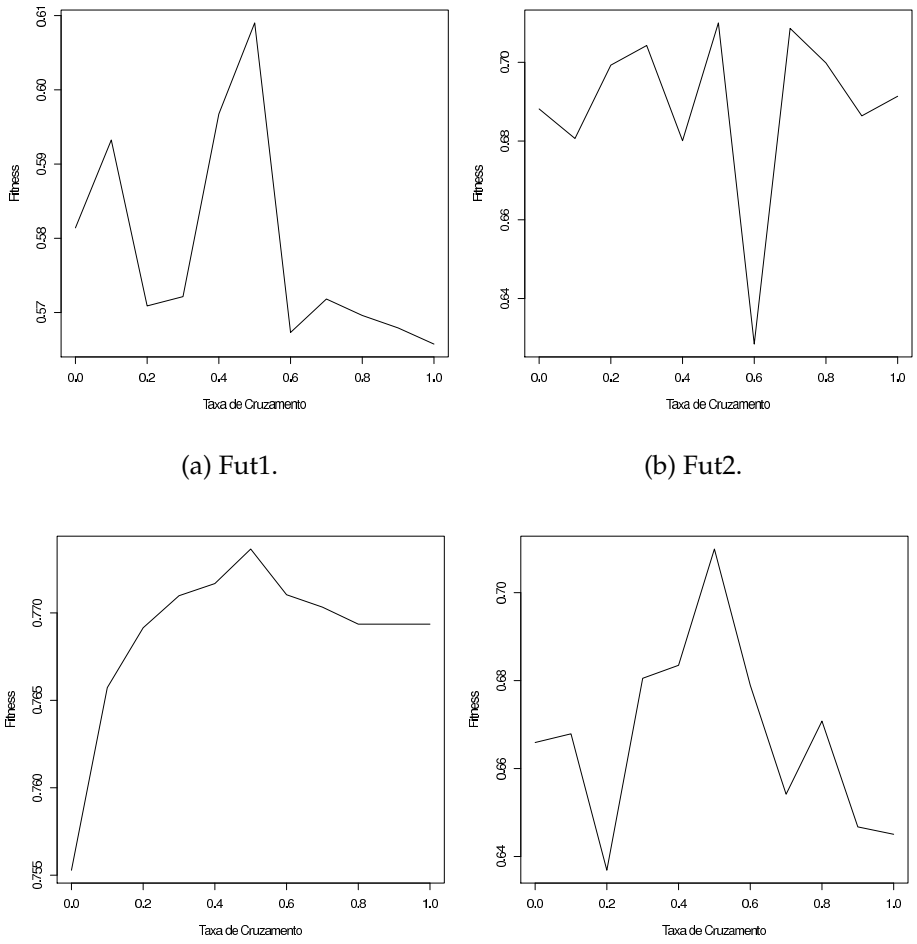
(c) Eleições. (d) Dengue.

Figura 4.6. Resultado para alterações do tamanho da população para cada base.

rendo especialização no conjunto de treinamento. Já na base Fut2 a quantidade mínima de gerações testadas não obteve os melhores resultados, ocorreu um aumento na qualidade aumentando a quantidade de gerações e acima de aproximadamente 500 gerações os resultados tendem a cair.

Quanto ao tamanho da população, o comportamento apresentou semelhança entre as bases, quando utilizado um número pequeno de indivíduos a técnica tende à fornecer uma pequena cobertura do espaço de soluções e uma quantidade de indivíduos grandes tende a tornar difícil o refinamento do modelo.

Em relação a taxa de cruzamento, utilizando um valor muito baixo os resultados não foram os melhores. Isto poderia ser resolvido com um maior tempo de processamento, mas como estamos trabalhando com fluxo de dados, não podemos consumir um tempo alto de processamento (por exemplo aumentando a quantidade



(a) Fut1. (b) Fut2.

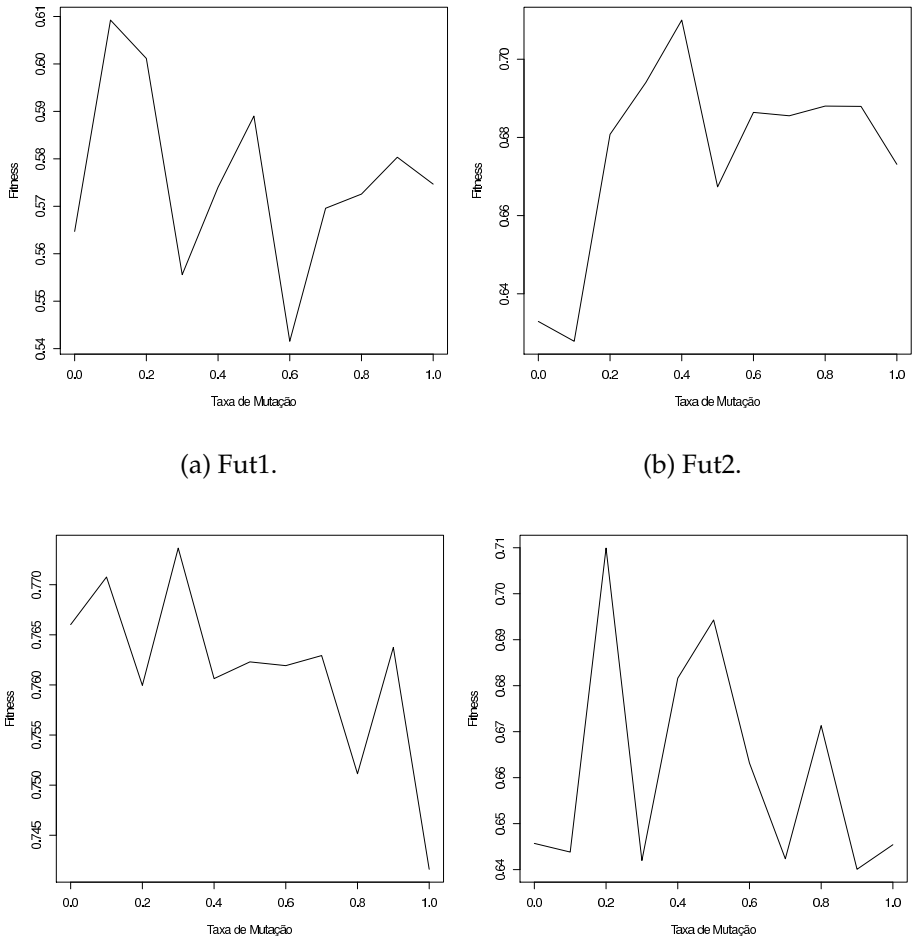
(c) Eleições. (d) Dengue.

Figura 4.7. Resultado para alterações na taxa de cruzamento para cada base.

de gerações) para encontrar um modelo. Já um valor alto introduz muito rápido novos modelos, perdendo assim estruturas com boa aptidão.

A mutação, por sua vez, obteve os melhores resultados para todas as bases entre 0.1 e 0.4 atendendo ao esperado de que uma taxa de mutação muito alta tende a tornar o algoritmo essencialmente aleatório e uma taxa baixa pode indicar que uma dada posição fique estagnada em um valor.

Em relação aos parâmetros relacionados ao fluxo de dados (Figuras 4.9, 4.10, 4.11, 4.12, 4.13 e 4.14) podemos observar que a seleção dos termos, pela entropia, melhorou o resultado somente na base Dengue. Entretanto, é importante notar que durante a seleção dos termos também utilizamos esta informação ajudando a diminuir o espaço de busca por novas soluções. Quanto a quantidade de termos presentes, esta não influenciou nas bases Fut2 e Dengue. Isso ocorreu devido ao fato de



(a) Fut1.

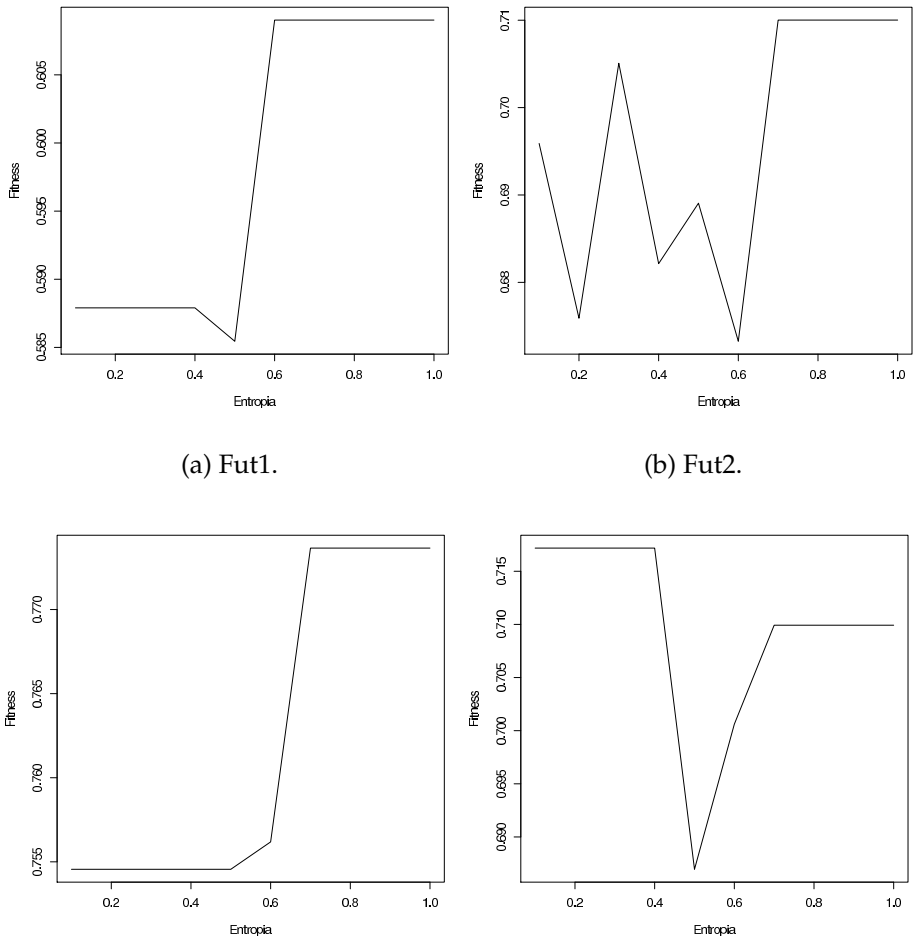
(b) Fut2.

(c) Eleições.

(d) Dengue.

Figura 4.8. Resultado para alterações na taxa de mutação para cada base.

que todos os indivíduos gerados nas execuções retornaram como melhor indivíduo soluções que não possuem nenhum termo para o vetor Presentes, ou seja, apenas o vetor de termos Ausentes foi suficiente para realizar a classificação. Um exemplo de indivíduo criado pelo sistema para a base Dengue possui os termos [foco, vida, governo, surto, deixa, reforça, dia, água, mortes, saúde, parada, banho, mosquito, contra, jovempan, medida, poderia, história, pneus, após] no vetor Ausentes e nenhum termo no vetor Presentes. Outro exemplo de indivíduo criado pelo sistema, mas agora para a base Fut2, possui os termos [tv, robinho, 0-1, brazil, brazilian, made, clock, defense, great, lead, draft, good, line, just, beautiful, wc2010, mins, x] no vetor Ausentes e nenhum termo no vetor Presentes. Desta forma, apenas o vetor de termos Ausentes foi capaz de realizar a classificação.



(a) Fut1. (b) Fut2.

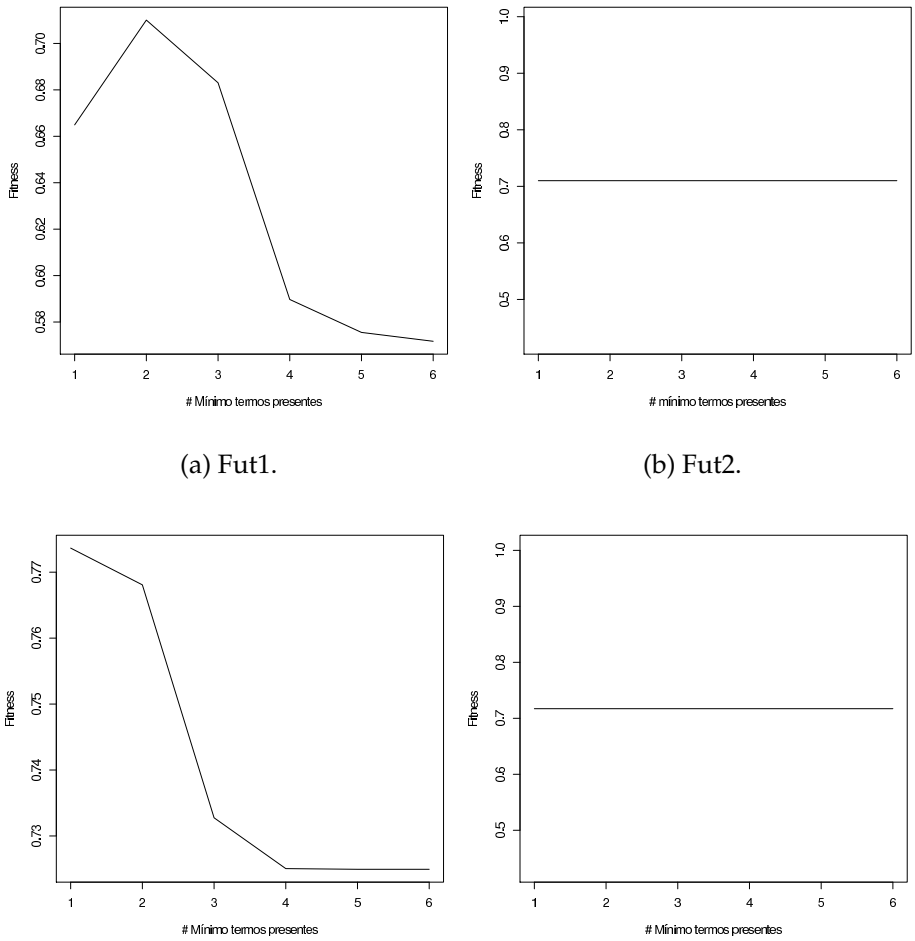
(c) Eleições. (d) Dengue.

Figura 4.9. Resultado para alterações na entropia máxima para cada base.

Uma característica interessante é que as duas bases que geraram indivíduos com o vetor Presentes vazio obtiveram melhores resultados quando determinaram que nenhum dos termos do vetor Ausentes podem aparecer na instância, para que esta seja considerada de determinada classe. Além disso, as bases Fut1 e Eleições conseguiram melhores resultados quando exigiram valores baixos para o vetor de termos Presentes e a quantidade de termos máximos do vetor Ausentes maior.

As bases Fut1 e Fut2 alcançaram melhores resultados com um conjunto de treinamento maior, enquanto as bases Eleições e Dengue exigiram um repositório menor. Este fato pode indicar que o vocabulário das duas últimas bases diversifica com maior frequência.

Já o parâmetro relacionado ao número mínimo de instâncias para poder verificar se ocorreu mudança na qualidade do classificador obteve comportamento



(a) Fut1. (b) Fut2.

(c) Eleições. (d) Dengue.

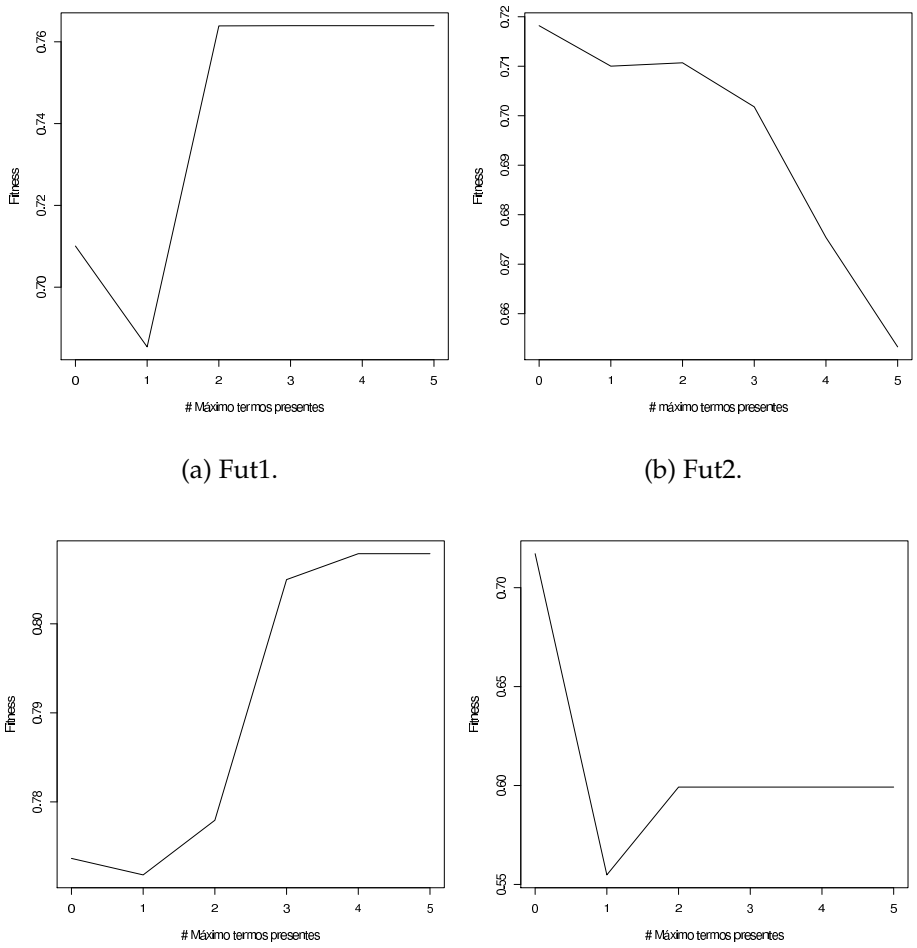
Figura 4.10. Resultado para alterações no número mínimo de termos presentes para cada base.

diferente para todas as bases, devido ao comportamento diversificado de cada uma delas. No caso da base Fut1 por exemplo, como a mudança é mais brusca, os dados devem ser verificados em intervalos pequenos para que a perda deste classificador não cause uma queda brusca na qualidade do classificador.

Quanto ao λ, como as alterações das bases Fut2, Eleições e Dengue são mais suaves, devemos deixar um valor mais baixo, o que não ocorre com a base Fut1.

Ao término destes testes, encontramos os seguintes valores de parâmetros de configuração do algoritmo que apresentaram melhores resultados, segundo os conjuntos de dados adotados para testes:

- Fut1 - P = 50, G = 50, M = 0.1, C = 0.5, EM = 0.6, TP = 2, TA = 2, TT = 100, NIM = 10 e λ = 5



(a) Fut1.

(b) Fut2.

(c) Eleições.

(d) Dengue.

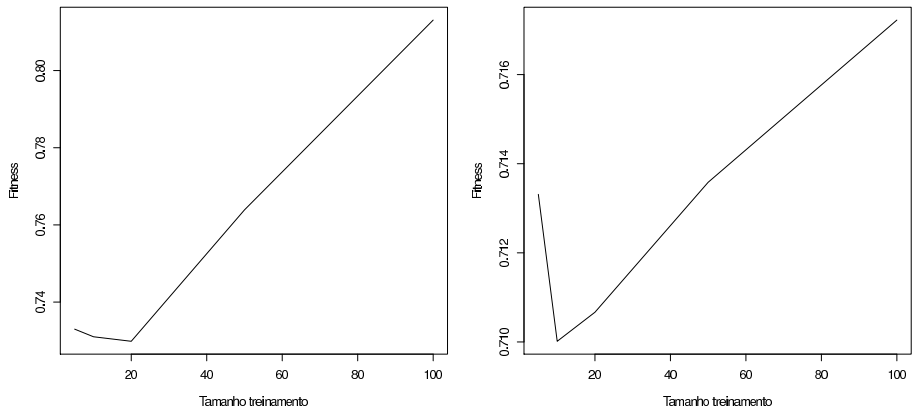
Figura 4.11. Resultado para alterações no número máximo de termos ausentes para cada base.

- Fut2 - P = 10, G = 500, M = 0.4, C = 0.5, EM = 0.7, TP = 1, TA = 0, TT = 100, NIM = 30 e $\lambda = 0$

- Eleições - P = 100, G = 50, M = 0.3, C = 0.5, EM = 0.7, TP = 1, TA = 4, TT = 5, NIM = 5 e $\lambda = 0$

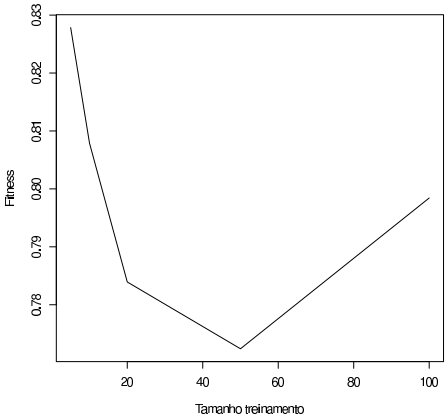
- Dengue - P = 50, G = 500, M = 0.2, C = 0.5, EM = 0.1, TP = 1, TA = 0, TT = 10, NIM = 10 e $\lambda = 0$

Na seção seguinte comparamos os resultados com dois algoritmos bem consolidados apresentados na literatura.

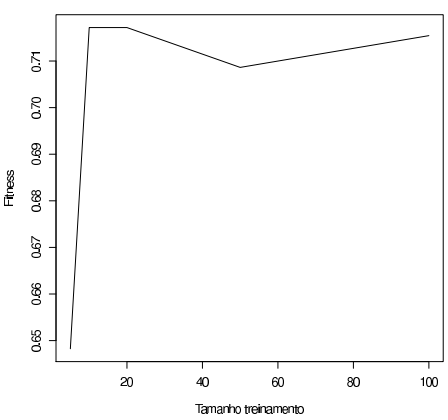


(a) Fut1.

(b) Fut2.



(c) Eleições.



(d) Dengue.

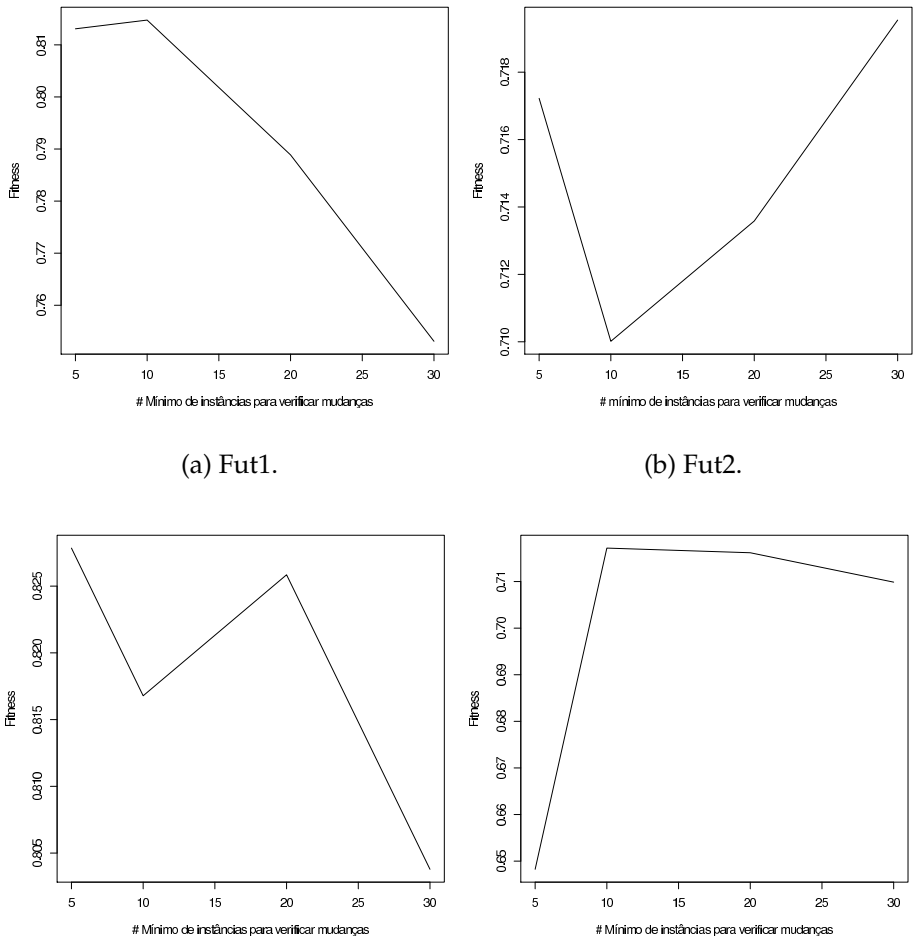
Figura 4.12. Resultado para alterações no tamanho do conjunto de treinamento para cada base.

4.2.2 Análise dos resultados e Comparação com métodos apresentados na literatura.

Os resultados obtidos por cada algoritmo para as bases de dados Fut1, Fut2, Eleições e Dengue, são apresentados na Tabela 4.2. Os dados informados pela tabela são:

- Média (M) – Média dos resultados de F1 (F1 médio) encontrados para 20 execuções considerando os melhores parâmetros.

- Desvio Padrão (DP) – Desvio Padrão dos resultados obtidos pelos experimentos.



(a) Fut1. (b) Fut2.

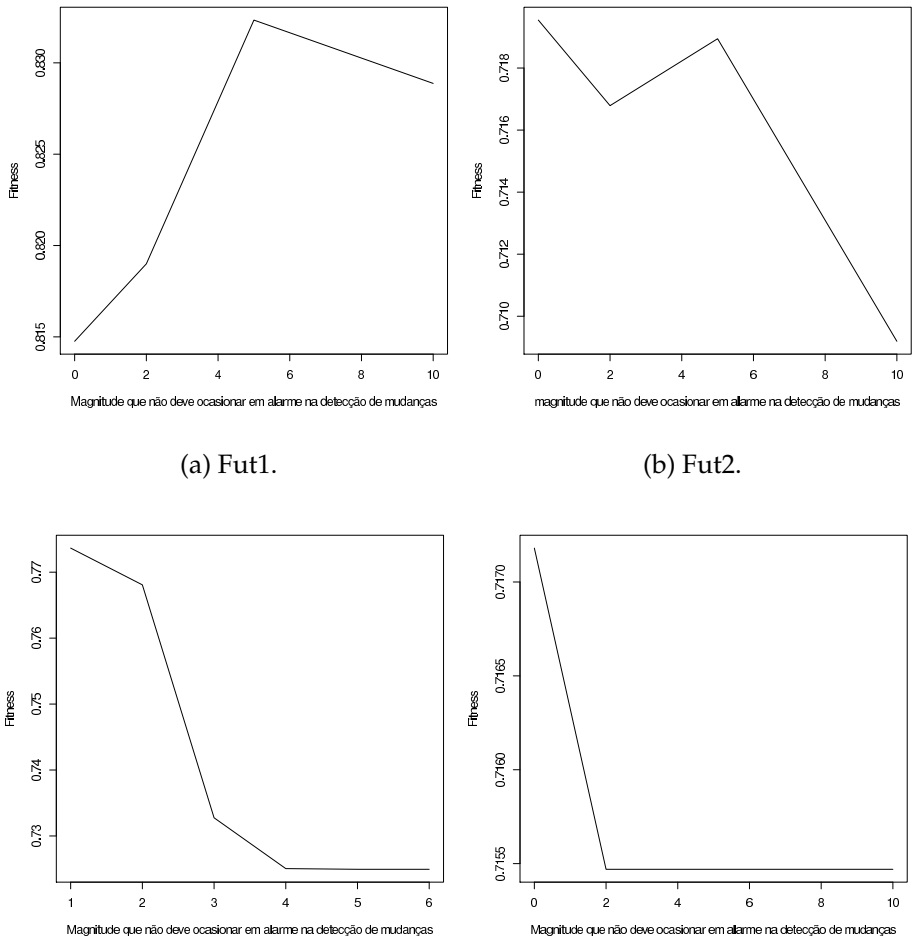
(c) Eleições. (d) Dengue.

Figura 4.13. Resultado da alteração no número mínimo de instâncias para verificar mudanças para cada base.

O algoritmo apresentado nesta pesquisa é representado por *AG*, sendo os demais: *Multinomial Naive Bayes* (NBM) e *Hoeffding Tree* (HT). Vale ressaltar também que a técnica utilizada para avaliação dos algoritmos foi a *Interleaved Test-Then-Train* (Seção 2.1.4), ou seja, cada exemplo é usado para testar o modelo antes de ser usado no conjunto de treinamento e a precisão é atualizada de forma incremental. Além disso, os métodos *NBM* e *HT* não apresentam DP devido ao fato de não ser possível realizar validação cruzada considerando que os dados tem uma ordem de chegada.

Para analise dos resultados foi utilizado o teste estatístico Kruskal-Wallis. Os testes estatísticos baseados em valores, caso desta pesquisa, podem ser divididos em dois grandes grupos [Mann, 2006]:

- Testes Paramétricos – se baseiam na hipótese de que as variáveis em análise



(a) Fut1. (b) Fut2.

(c) Eleições. (d) Dengue.

Figura 4.14. Resultado para alterações no λ para cada base.

Tabela 4.2. Resultado alcançados por cada classificador.

	Fut1		Fut2		ELEIÇÃO		DENGUE	
	M	DP	M	DP	M	DP	M	DP
AG	0,83	0,004	0,72	0,02	0,83	0,01	0,72	0,005
NBM	0,91	-	0,73	-	0,89	-	0,53	-
HT	0,92	-	0,72	-	0,80	-	0,67	-

apresentam uma distribuição normal;

- Testes Não-paramétricos – conhecidos também como testes de distribuição livre, neste não é necessário supor que a variável em análise apresenta distribuição normal.

Desta forma, antes de decidir o teste a ser realizado, foi investigada a distribuição das amostras a serem analisadas. O teste utilizado para essa verificação foi o Shapiro-Wilk. Esta escolha baseou-se na sua capacidade de adaptação a uma variada gama de problemas sobre avaliação de normalidade. Os resultados apresentados pelo teste Shapiro-Wilk mostraram que as amostras não apresentavam normalidade, foi então utilizado para análise estatística, o teste de Kruskal-Wallis, conhecido também como Teste H. Este corresponde a um teste não-paramétrico e destina-se a casos de mais de duas opções na comparação de dados. A hipótese nula deste teste é de que as distribuições das populações sob consideração sejam todas idênticas. Já a hipótese alternativa, é de que pelo menos uma das distribuições das populações seja diferente e que, por conseguinte, nem todas as distribuições das populações sejam idênticas [Mann, 2006]. As variáveis envolvidas neste teste foram: o classificador utilizado e a *fitness*.

A análise realizada possibilitou observar, com 5% de significância, que o método proposto obteve melhores resultados para a base Dengue, na base Eleições o algoritmo *NBM* obteve o melhor resultado, porém o método proposto foi melhor que o algoritmo *HT*, na base Fut1 as outras técnicas obtiveram resultados melhores e na base Fut2 não foi encontrada diferença estatística de acordo com a hipótese considerada.

Comparando os resultados da Tabela 4.2 e os gráficos que caracterizam as bases de dados (Seção 4.1), podemos observar que o classificador obteve melhores resultados nas bases com mudanças mais suaves. Isso pode ocorrer por dois motivos:

- Demora para recuperar em casos de mudanças bruscas - como o *AG* necessita de um número mínimo de instâncias para verificar se ocorreram mudanças, isto pode atrasar um pouco a descoberta do *Concept Drift* se comparados a algoritmos incrementais;

- Em casos de mudanças mais suaves a utilização da evolução da população do *AG* foi capaz de adaptar a nova população às pequenas alterações na distribuição dos dados, principalmente conseguindo eliminar termos desatualizados.

Além dos resultados alcançados é importante observar o tempo de execução desses algoritmos. Quando trabalhamos com fluxo de dados o tempo de execução pode tornar um algoritmo inviável para determinado problema. Na Tabela 4.3 podemos analisar o tempo gasto por cada algoritmo referente a cada uma das bases. Vale ressaltar aqui que para o método proposto foi informado o tempo médio e o

desvio padrão das execuções. O tempo gasto pelo AG foi um pouco maior que os gastos pelo algoritmo *NBM* e menor que os gastos pelo algoritmo *HT* para as bases Fut2 e Eleições, e menor nas bases Dengue e Fut1. Este ganho no desempenho do *AG* é alcançado principalmente pela seleção dos termos com menor entropia, o que diminui consideravelmente o espaço de soluções.

Tabela 4.3. Tempo de execução (em minutos) para cada base de dados.

	Fut1		Fut2		DENGUE		ELEIÇÃO	
	M	DP	M	DP	M	DP	M	DP
AG	2,45	0,15	1,89	1,79	0,25	0,13	10,34	8,8
NBM	3,07	-	0,8	-	0,3	-	4,26	-
HT	11,15	-	3,49	-	0,08	-	50,35	-

Outra informação investigada foi a quantidade de atualizações para cada base. A Tabela 4.4 apresenta o número médio de atualizações e o desvio padrão para cada base de dados considerando o melhor resultado encontrado. Para as bases Dengue, Eleições e Fut2 o melhor resultado foi obtido atualizando sempre que a verificação de alterações era realizada. Isto pode ser explicado devido a diversidade de vocabulário contida nestas bases e ao fato destas três bases possuírem mudanças suaves na distribuição. Além disso, a base Fut2 obteve usuários de diversas regiões e com opiniões diversificadas sobre o jogo. Já a base Fut1 foi gerada por um público com opinião bem semelhante, visto que a maioria são brasileiros (devido ao idioma português), estavam torcendo para o mesmo time e a base possui mudanças bruscas na distribuição dos dados.

Tabela 4.4. Quantidade de atualizações para cada base de dados.

	Fut1		Fut2		ELEIÇÃO		DENGUE	
	M	DP	M	DP	M	DP	M	DP
AG	2	0,8	47	0	13315	0	6	0

Capítulo 5

Conclusões e Trabalhos Futuros

Nesta dissertação, apresentamos um algoritmo para classificação de fluxo de dados baseado em Algoritmos Genéticos. A ideia principal consistiu em aproveitar a característica de evolução do AG para evoluir os classificadores junto com a evolução dos dados. O foco da pesquisa foram bases de redes sociais como o *Twitter*, com vocabulário extenso e textos curtos. Adaptações no AG padrão foram aqui introduzidas, procurando adaptar estes algoritmos para lidar com fluxo de dados.

O resultado foi um AG que utiliza uma técnica de repositório, que testa variações do modelo utilizando um teste estatístico *Page-Hinkley* (PH) e atualiza o modelo aproveitando os operadores de evolução dos AGs.

O método proposto foi avaliado para 4 bases de dados distintas, e os seus resultados foram comparados a 2 algoritmos bem adaptados para problemas semelhantes aos tratados nesta pesquisa. Resultados experimentais apresentados demonstraram que o método obteve equivalência estatística nos resultados referentes a base Fut2, melhores resultados na base Dengue, na base Eleições o algoritmo *NBM* obteve o melhor resultado, porém o método proposto foi melhor que o algoritmo *HT* e na base Fut1 as outras técnicas obtiveram resultados melhores.

Diferentes abordagens podem ser seguidas com o objetivo de melhorar o método. O primeiro ponto trata-se de diminuir o número de parâmetros que deve ser atualmente definido pelo usuário. Através dos experimentos preliminares realizados nessa dissertação, percebeu-se quais parâmetros apresentam maiores variações e quais podem ser melhor controlados, e essa informação servirá de insumo para deixar o método mais independente de parâmetros. Além disso, comparações com outros trabalhos, tais como os propostos em Vivekanandan & Nedunchezhian [2011, 2010], servirão para mostrar ainda mais a eficácia do algoritmo apresentado.

Outras direções para trabalhos futuros incluem trabalhar com comitês de clas-

sificação, aproveitando que o AG mantêm uma população de indivíduos (possíveis soluções para o problema). Em termos de eficiência, podemos também citar a utilização de Unidades de Processamento Gráfico (*Graphic Processing Unit* - GPU). As GPU's foram propostas inicialmente para processamento gráfico, sendo adaptadas para problemas genéricos devido a característica de processamento paralelo e ao alto poder computacional para cálculos [Viana, 2009]. Alguns trabalhos, com bons resultados, que utilizam esta técnica em conjunto com AE podem ser encontradas em Yu et al. [2005] e Tsutsui & Fujimoto [2009].

Algumas outras adaptações para trabalharmos com aprendizado semi-supervisionado podem ser interessantes, assim como a introdução de nichos e espécies no AG poderia melhorar a diversidade dos classificadores gerados.

Referências Bibliográficas

Al-Kateb, M.; Lee, B. S. & Wang, X. S. (2007). Adaptive-Size Reservoir Sampling over Data Streams. Em *Proceedings of the 19th International Conference on Scientific and Statistical Database Management*, SSDBM '07, pp. 22--, Washington, DC, USA. IEEE Computer Society.

Arango, H. (2005). *Bioestatística Teórica e Computacional*. Guanabara KooGan S.A, 2ª Edição. Rio de Janeiro,Brasil, 2st edição.

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK.

Bäck, T. (2002). Evolutionary Computation: A Guided Tour. *Bulletin of the EATCS*, 77:132--166.

Bifet, A. & Frank, E. (2010). Sentiment knowledge discovery in twitter streaming data. Em *Proceedings of the 13th international conference on Discovery science*, DS'10, pp. 1--15, Berlin, Heidelberg. Springer-Verlag.

Bifet, A. & Kirkby, R. (2009). Data Stream Mining - A Practical Approach.

Cha, M.; Haddadi, H.; Benevenuto, F. & Gummadi, K. P. (2010). Measuring user influence in Twitter: The million follower fallacy. Em *in ICWSM '10: Proceedings of international AAAI Conference on Weblogs and Social*.

Chaudhry, N.; Shaw, K. & Abdelguerfi, M. (2005). *Stream data management*. Kluwer international series on advances in database systems. Springer.

Diakopoulos, N. A. & Shamma, D. A. (2010). Characterizing debate performance via aggregated twitter sentiment. Em *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pp. 1195--1198, New York, NY, USA. ACM.

Domingos, P. & Hulten, G. (2000). Mining high-speed data streams. Em *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pp. 71--80, New York, NY, USA. ACM.

Eiben, A. E. & Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer-Verlag.

Folino, G. & Papuzzo, G. (2010). Handling Different Categories of Concept Drifts in Data Streams Using Distributed GP. Em Esparcia-Alcázar, A.; Ekárt, A.; Silva, S.; Dignum, S. & Uyar, A., editores, *Genetic Programming*, volume 6021 of *Lecture Notes in Computer Science*, pp. 74--85. Springer Berlin / Heidelberg. 10.1007/978-3-642-12148-7_7.

Folino, G.; Pizzuti, C. & Spezzano, G. (2007). Mining distributed evolving data streams using fractal GP ensembles. Em *Proceedings of the 10th European conference on Genetic programming*, EuroGP'07, pp. 160--169, Berlin, Heidelberg. Springer-Verlag.

Folino, G.; Pizzuti, C. & Spezzano, G. (2008). Training distributed gp ensemble with a selective algorithm based on clustering and pruning for pattern classification. *Evolutionary Computation, IEEE Transactions on*, 12(4):458--468.

Freitas, A. A. (2002). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edição.

Gama, J. & Pinto, C. (2006). Discretization from data streams: applications to histograms and data mining. Em *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pp. 662--667, New York, NY, USA. ACM.

Gama, J. & Rodrigues, P. (2009). An Overview on Mining Data Streams. Em Abraham, A.; Hassanien, A.-E.; de Leon F. de Carvalho, A. & Snášel, V., editores, *Foundations of Computational, IntelligenceVolume 6*, volume 206 of *Studies in Computational Intelligence*, pp. 29--45. Springer Berlin / Heidelberg. 10.1007/978-3-642-01091-0_2.

Gardner, W. A. (1984). Learning characteristics of stochastic-gradient-descent algoritms: a general study, analysis, and critique. *Signal Processing*, 6(2):113–133.

Gomide, J.; Veloso, A.; Meira, W.; Almeida, V.; Benevenuto, F.; Ferraz, F. & Teixeira, M. (2012). Dengue surveillance based on a computational model of spatio-temporal locality of twitter. *Proceedings of the ACM*, pp. 1--8.

Hashemi, S. & Yang, Y. (2009). Flexible decision tree for data stream classification in the presence of concept change, noise and missing values. *Data Min. Knowl. Discov.*, 19(1):95--131.

Helmbold, D. P. & Long, P. M. (1994). Tracking Drifting Concepts By Minimizing Disagreements. *Mach. Learn.*, 14:27--45.

Jansen, B. J.; Zhang, M.; Sobel, K. & Chowdury, A. (2009). Micro-blogging as online word of mouth branding. Em *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pp. 3859--3864, New York, NY, USA. ACM.

Kibriya, A. M.; Frank, E.; Pfahringer, B. & Holmes, G. (2004). Multinomial naive bayes for text categorization revisited. Em *Proceedings of the 17th Australian joint conference on Advances in Artificial Intelligence*, AI'04, pp. 488--499, Berlin, Heidelberg. Springer-Verlag.

Kifer, D.; Ben-David, S. & Gehrke, J. (2004). Detecting change in data streams. Em *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pp. 180--191. VLDB Endowment.

Klinkenberg, R. & Joachims, T. (2000). Detecting Concept Drift with Support Vector Machines. Em *In Proceedings of the Seventeenth International Conference on Machine Learning (ICML*, pp. 487--494. Morgan Kaufmann.

Klinkenberg, R. & Renz, I. (1998). Adaptive Information Filtering: Learning in the Presence of Concept Drifts. Em *Workshop Notes of the ICML/AAAI-98 Workshop Learning for Text Categorization*, pp. 33--40. AAAI Press.

Kumar, D. J. N.; Murthy, J. V. R.; Satapathy, S. C. & Pullela, S. V. V. S. R. K. (2011). A study of decision tree induction for data stream mining using boosting genetic programming classifier. Em *Proceedings of the Second international conference on Swarm, Evolutionary, and Memetic Computing - Volume Part I*, SEMCCO'11, pp. 315--322, Berlin, Heidelberg. Springer-Verlag.

Lanquillon, V. C. (2001). *Enhancing text classification to improve information filtering*.

Li, W.; Jin, X. & Ye, X. (2007). Detecting Change in Data Stream: Using Sampling Technique. Em *Proceedings of the Third International Conference on Natural Computation - Volume 01*, ICNC '07, pp. 130--134, Washington, DC, USA. IEEE Computer Society.

Maloof, M. A. & Michalski, R. S. (2000). Selecting Examples for Partial Memory Learning. *Mach. Learn.*, 41(1):27--52.

Mann, P. (2006). *Introdução a estatíscica*. LTC-Livros Tecnicos e Cientificos Editora S.A., 5ª Edição. Rio de Janeiro,Brasil, 5st edição.

McCallum, A. & Nigam, K. (1998). A comparison of event models for naive bayes text classification. Em *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*, pp. 41--48. AAAI Press.

Mitchell, T. M. (1997). *Machine Learning*. New York, NY, USA : McGraw-Hill.

Mola, F. & Miele, R. (2006). Evolutionary Algorithms for Classification and Regression Trees. Em Zani, S.; Cerioli, A.; Riani, M. & Vichi, M., editores, *Data Analysis, Classification and the Forward Search*, Studies in Classification, Data Analysis, and Knowledge Organization, pp. 255--262. Springer Berlin Heidelberg.

Pei, M.; Goodman, E. D.; Iii, W. F. P. & Ding, Y. (1995). Genetic algorithms for classification and feature extraction. Em *Annual Meeting, Classification Society of North America*.

Pietramala, A.; Policicchio, V.; Rullo, P. & Sidhu, I. (2008). A Genetic Algorithm for Text Classification Rule Induction. Em Daelemans, W.; Goethals, B. & Morik, K., editores, *Machine Learning and Knowledge Discovery in Databases*, volume 5212 of *Lecture Notes in Computer Science*, pp. 188--203. Springer Berlin / Heidelberg. 10.1007/978-3-540-87481-2_13.

Rezende, S. O. (2003). *Sistemas Inteligentes - Fundamentos e Aplicações*. Manole, Barueri, SP, Brasil.

Silva, I. S. (2012). Análise Adaptativa de Fluxo de Sentimentos . Dissertação de mestrado, Belo Horizonte, MG, Brasil.

Silva, I. S.; Gomide, J.; Veloso, A.; Meira, J. W. & Ferreira, R. (2011). Effective sentiment stream analysis with self-augmenting training and demand-driven projection. Em *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, SIGIR '11, pp. 475--484, New York, NY, USA. ACM.

Street, W. N. & Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. Em *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pp. 377--382, New York, NY, USA. ACM.

Tan, P.-N.; Steinbach, M. & Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Tanwani, A. K. & Farooq, M. (2009). Performance evaluation of evolutionary algorithms in classification of biomedical datasets. Em *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pp. 2617--2624, New York, NY, USA. ACM.

Tsutsui, S. & Fujimoto, N. (2009). Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study. Em *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pp. 2523--2530, New York, NY, USA. ACM.

Veloso, A. & Meira-Junior, W. (2011). *Demand-Driven Associative Classification*. Springerbriefs in Computer Science. Springer.

Viana, J. R. M. (2009). Programação em GPU: Presente, Passado e Futuro. volume 1, pp. 1--25, Parnaíba. Escola Regional de Computação do Ceará (ERCEMAPI)).

Vivekanandan, P. & Nedunchezhian, R. (2010). A fast genetic algorithm for mining classification rules in large datasets. *International Journal on Soft Computing (IJSC)*, 1(1):10--20.

Vivekanandan, P. & Nedunchezhian, R. (2011). Mining data streams with concept drifts using genetic algorithm. *Artif. Intell. Rev.*, 36:163--178.

Wang, H.; Fan, W.; Yu, P. S. & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. Em *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pp. 226--235, New York, NY, USA. ACM.

Widmer, G. & Kubat, M. (1996a). Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69--101.

Widmer, G. & Kubat, M. (1996b). Learning in the Presence of Concept Drift and Hidden Contexts. Em *Machine Learning*, pp. 69--101.

Yu, Q.; Chen, C. & Pan, Z. (2005). Parallel Genetic Algorithms on Programmable Graphics Hardware. Em *Lecture Notes in Computer Science 3612*, p. 1051. Springer.

Zhu, X. & Goldberg, A. (2009). *Introduction to Semi-Supervised Learning*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool.