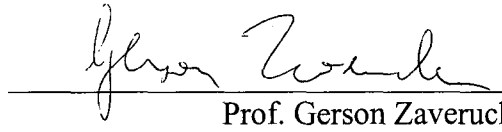
SUNRISE: UM ALGORITMO RÁPIDO DE APRENDIZADO MULTI-ESTRATÉGIA

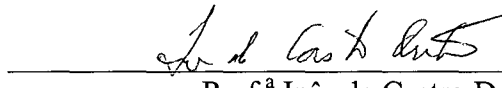
Aloísio Carlos de Pina

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

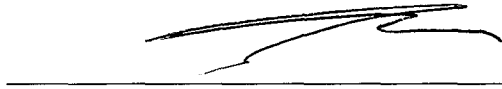
Aprovada por:



Prof. Gerson Zaverucha, Ph.D.



Prof.ª Inês de Castro Dutra, Ph.D.



Prof. André Carlos Ponce de Leon Ferreira de Carvalho, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2003

PINA, ALOÍSIO CARLOS DE

SUNRISE: Um Algoritmo Rápido de Aprendizado Multi-estratégia [Rio de Janeiro] 2003

VIII, 59 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2003)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Aprendizado Multi-estratégia
2. Aprendizado Baseado em Instâncias
3. Indução de Regras

I. COPPE/UFRJ II. Título (série)

À minha Família.

A vitória e o sucesso de cada um de nós sempre representará a vitória e o sucesso para todos nós. Amo vocês.

AGRADECIMENTOS

Agradeço a todos aqueles que direta ou indiretamente tenham colaborado de alguma forma para a realização deste trabalho.

Agradeço ao apoio financeiro fornecido pela CAPES durante todo o decorrer do trabalho. Enquanto existirem pessoas interessadas em financiar pesquisas para o avanço da tecnologia e conseqüente melhoria da sociedade, o homem continuará evoluindo.

Agradeço ao professor Gerson Zaverucha pela fundamental ajuda no desenvolvimento deste trabalho. Graças à sua experiência e inteligência, seus conselhos e orientações foram muito importantes para conclusão do trabalho. Além disso, durante todo o período que convivemos ele mostrou ser mais do que um simples orientador. Obrigado, meu amigo.

Agradeço à minha Família pelo apoio incondicional demonstrado em todos os momentos. É muito bom saber que existe alguém sempre torcendo e vibrando com nossas conquistas, rezando para que tudo transcorra sem problemas. Essa força foi e sempre será essencial para o término deste e de outros futuros trabalhos.

Acima de tudo e de todos, agradeço a DEUS pela chance de poder desenvolver este trabalho, completando mais um ciclo de minha vida com sucesso, tornando real mais um de meus sonhos. Muito obrigado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SUNRISE: UM ALGORITMO RÁPIDO DE APRENDIZADO MULTI-ESTRATÉGIA

Aloísio Carlos de Pina

Abril/2003

Orientador: Gerson Zaverucha

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta a análise, implementação e avaliação experimental do SUNRISE, um novo algoritmo de aprendizado multi-estratégia rápido. O algoritmo foi comparado com seu predecessor usando-se um teste estatístico aproximado, o teste *t* emparelhado com validação cruzada com *n* partições. Nos testes foram usados muitos conjuntos de dados, numa tentativa de incluir vários domínios, tamanhos e dificuldades. Comparando os tempos médios de execução dos dois algoritmos, pôde-se verificar que o novo algoritmo é bem mais rápido e possui acurácia similar à de seu predecessor, confirmando que o objetivo da pesquisa foi alcançado.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SUNRISE: A FAST MULTI-STRATEGY LEARNING ALGORITHM

Aloísio Carlos de Pina

April/2003

Advisor: Gerson Zaverucha

Department: Systems Engineering and Computer Science

This work presents the analysis, implementation and experimental evaluation of SUNRISE, a new fast multi-strategy learning algorithm. The algorithm was compared to its predecessor by using an approximate statistical test, the *n*-fold cross-validated paired *t*-test. Many data sets were used in the tests, in an attempt to include several domains, sizes and difficulties. Comparing the average execution times of the two algorithms, it could be verified that the new algorithm is faster and possesses similar accuracy to the one of its predecessor, confirming that the objective of the research was reached.

SUMÁRIO

1. INTRODUÇÃO

No passado, a maioria das pesquisas em Aprendizado de Máquinas relacionava-se com métodos que empregam uma única estratégia de aprendizado, isto é, métodos mono-estratégia. Com o crescente entendimento das capacidades e limitações dos métodos de aprendizado mono-estratégia houve um aumento no interesse nos sistemas multi-estratégia (MICHALSKI & TECUCI, 1994), que tentam combinar dois ou mais diferentes paradigmas de aprendizado em um único algoritmo. Tais sistemas podem aprender a partir de um escopo de entrada mais amplo e serem aplicados a uma faixa de problemas maior que os métodos mono-estratégia.

O RISE (DOMINGOS, 1994) é um algoritmo multi-estratégia bem conhecido, que tenta unir as melhores características da indução de regras (MICHALSKI, 1983) e do aprendizado baseado em instâncias (AHA *et al.*, 1991) em um único algoritmo. Infelizmente, apesar de em muitos domínios o RISE possuir acurácia maior que vários algoritmos de aprendizado, tais como PEBLS (COST & SALZBERG, 1993), CN2 (CLARK & NIBLETT, 1989) e C4.5 (QUINLAN, 1993), para conjuntos de dados grandes ele tem um tempo médio de execução muito alto.

Este trabalho apresenta a análise, implementação e avaliação experimental do SUNRISE, um novo algoritmo de aprendizado multi-estratégia baseado no algoritmo RISE e que tenta amenizar sua principal desvantagem, sua velocidade, mantendo sua alta acurácia preditiva. Além disso, foi desenvolvida uma versão modificada do algoritmo SUNRISE especialmente para a aplicação em mineração de texto: o algoritmo TextSUNRISE. O novo algoritmo foi testado e seus resultados foram avaliados.

A Tese está organizada como segue. O próximo capítulo contém uma rápida revisão dos métodos de aprendizado e dos métodos de teste mais relevantes para este trabalho, bem como as principais características do algoritmo RISE, predecessor do SUNRISE. O algoritmo SUNRISE é então apresentado e analisado no Capítulo 3. Finalmente, são apresentadas as conclusões e os planos para pesquisas futuras no Capítulo 4.

2. CONHECIMENTOS PRELIMINARES

Neste capítulo será feita um rápida revisão dos tópicos necessários para o entendimento do trabalho desenvolvido. A próxima seção contém uma revisão das características dos métodos de aprendizado mais relevantes para este trabalho. A Seção 2.2 contém algumas noções sobre estatística utilizadas na avaliação experimental do algoritmo e descreve dois métodos de teste populares usados para comparar a performance de algoritmos de aprendizado. As principais características do algoritmo que serviu de base para o desenvolvimento do SUNRISE são apresentadas na Seção 2.3.

2.1. Métodos de Aprendizado

Aprendizado indutivo é a criação explícita ou implícita de um conceito geral ou descrições de classes a partir dos exemplos. Um conjunto de treinamento de exemplos pré-classificados é dado, onde cada exemplo (também chamado de observação ou caso) é descrito por um vetor de características ou valores de atributos, e o objetivo é formar uma descrição que possa ser usada para classificar novos exemplos com alta acurácia.

Dois paradigmas de indução largamente utilizados e que parecem ter vantagens e desvantagens complementares são o aprendizado baseado na indução de regras e o aprendizado baseado em instâncias (IBL - *Instance-Based Learning*). Neste trabalho, esses dois métodos de aprendizado supervisionado serão abordados somente na tarefa de classificação, pois as funções objetivo consideradas assumem apenas valores discretos.

2.1.1. Aprendizado Baseado em Instâncias

Técnicas de aprendizado baseado em instâncias funcionam essencialmente mantendo exemplos de atributos típicos para cada classe.

O aprendizado baseado em instâncias está fundamentado na aplicação direta do conceito de similaridade. Uma função de similaridade diz ao algoritmo o quão próximas duas instâncias estão. Embora isso pareça simples, existe uma grande complexidade em se escolher a função de similaridade, especialmente em situações onde alguns dos atributos são simbólicos. Por exemplo, se a tarefa fosse classificar pessoas, e um dos atributos fosse a cor do cabelo, não é evidente o que a distância significaria nesse contexto.

O aprendizado baseado em instâncias também é conhecido por muitos outros nomes, tais como: aprendizado baseado em exemplares, baseado em memória, baseado em casos, baseado em experiências, baseado em *kernel*, *nearest-neighbor* (vizinho mais próximo), local, e *lazy* (preguiçoso). Há muitas variações do tema básico. No caso mais simples, o aprendizado é realizado armazenando-se todos os exemplos observados.

A função de classificação é aquela que, quando dado um novo caso, decide como ele se relaciona aos casos aprendidos. Por exemplo, esta função poderia retornar o exemplo de treinamento que é o mais próximo ao novo exemplo. Os exemplos armazenados usados para classificar novos casos são chamados de instâncias ou exemplares. Um exemplo novo (ou "caso teste") é classificado encontrando-se a instância mais próxima a ele de acordo com alguma função de similaridade, e atribuindo a classe da instância ao exemplo.

Uma extensão ao paradigma básico de IBL consiste em usar os k vizinhos mais próximos (*k-Nearest Neighbor*) para classificação, em vez de usar apenas o mais próximo e classificar de acordo com ele. A classe atribuída é então aquela da maioria dos k vizinhos, ou a classe que receber a maioria de votos, de forma que quanto mais próximo um vizinho estiver do exemplo de teste, maior será o valor de seu voto.

O desempenho de IBL depende criticamente da medida de similaridade (ou, inversamente, distância) usada. Em domínios numéricos (isto é, domínios onde os valores de todos os atributos são números reais), a distância bloco-cidade e distância Euclideana são candidatas naturais.

Se houver a atributos, a distância entre duas instâncias $E_1 = (e_{11}, e_{12}, ..., e_{1a}, C_1)$ e $E_2 = (e_{21}, e_{22}, ..., e_{2a}, C_2)$, com valores e_{1i} e e_{2i} para o i-ésimo atributo e classes C_1 e C_2, pode então ser definida como:

$$\Delta(E_1, E_2) = \sum_{i=1}^{a} \delta^s(e_{1i}, e_{2i})$$

que com $s = 1$ fornece a distância bloco-cidade e com $s = 2$ o quadrado da distância Euclideana. A componente da distância $\delta(x_i, x_j)$ entre dois valores x_i e x_j de um atributo pode ser simplesmente o valor absoluto de sua diferença. Entretanto, isso pode levar a atributos com valores muito dispersos tendo um peso exagerado no resultado, comparado aos atributos com dispersões menores. Uma abordagem geralmente usada (veja, por exemplo, SALZBERG, 1991) é normalizar a diferença usando seu maior valor observado:

$$\delta(x_i, x_j) = \left| \frac{x_i - x_j}{x_{\max} - x_{\min}} \right|$$

Os atributos simbólicos representam um problema mais difícil. A maioria dos sistemas de IBL (por exemplo, AHA *et al.*, 1991) usam uma simples métrica de envoltório:

$$\delta(x_i, x_j) = \begin{cases} 0 & \text{se } i = j \\ 1 & \text{caso contrário} \end{cases}$$

Essa medida é pouco informativa, e, embora seja apropriada em alguns casos, seu uso pode conduzir a um desempenho pobre. Uma alternativa mais sofisticada consiste em considerar dois valores simbólicos similares se fizerem predições similares (isto é, se eles se correlacionarem similarmente com o atributo de classe). Isso foi primeiro proposto por STANFILL & WALTZ (1986) como a parte de uma métrica de diferença de valores (VDM - *Value Difference Metric*). A distância total $\Delta(E_1, E_2)$ é computada como antes. Diferentes variantes dessa métrica foram usadas com sucesso em processamento de fala, biologia molecular e outras tarefas (STANFILL & WALTZ, 1986; COST & SALZBERG, 1993).

Uma vantagem deste método de aprendizado é que ao invés de estimar a função objetivo uma única vez para todo o espaço de exemplos, ele pode estimá-la localmente e diferentemente para cada novo exemplo a ser classificado. Também são fáceis de testar, conceitualmente simples e rápidos, uma vez que o treinamento consiste apenas em armazenar as instâncias. Tais métodos são naturalmente adequados a domínios numéricos, onde o conceito de distância tem um sentido mais concreto.

Entretanto, os métodos de IBL precisam considerar o problema da sensibilidade a atributos irrelevantes. As contribuições desses atributos à distância global constituem ruído no que diz respeito à tarefa de classificação, e podem mascarar

a importância dos componentes relevantes. Muitas abordagens foram propostas para tratar deste problema (AHA, 1990).

Um outro problema dos métodos de IBL é sua sensibilidade a ruídos. As instâncias incorretas são responsáveis por criar uma região em torno delas onde novos exemplos serão também classificados erroneamente. Diversos métodos foram apresentados com sucesso para tratar desse problema (AHA *et al.*, 1991; COST & SALZBERG, 1993).

Outra dificuldade que os métodos de aprendizado baseado em instâncias enfrentam é que às vezes o número de instâncias que necessitam ser armazenadas é grande, podendo requerer uma quantidade moderadamente grande de espaço para armazenamento. Isso não é um problema em si, mas pode impor limites na velocidade da busca pelo exemplo mais próximo. Para lidar com isso, alguns algoritmos de aprendizado baseado em instâncias descartam instâncias que são classificadas corretamente pelas demais do conjunto de treinamento, economizando algum espaço. Existem algoritmos que fazem algumas suposições sobre os dados e usam métodos estatísticos para se livrar de exemplos irrelevantes ou ruídos. Para isso utilizam-se de uma função de seleção de "instância típica" que diz ao algoritmo quais das instâncias devem ser mantidas e quais devem ser descartadas. O problema é saber quais instâncias são "típicas" e quais são "atípicas".

Além disso, o aprendizado baseado em instâncias é basicamente uma forma de aprendizado muito desestruturada: meramente armazena os exemplos sem processar os dados de forma alguma. Nenhum "conceito" é produzido em um formato legível pelo ser humano.

2.1.2. Aprendizado Baseado na Indução de Regras

Algoritmos de aprendizado baseados em indução de regras podem ser descritos como algoritmos que determinam um conjunto de regras que formam um relacionamento entre os atributos e as classes. Há uma variedade de ferramentas que funcionam dessa forma, e variam na maneira que constróem essas regras.

Os algoritmos de indução de regras (MICHALSKI, 1983) tipicamente empregam uma abordagem de cobertura de conjunto ou "separação e conquista" para a indução. Essa estratégia deriva seu nome do fato que ela forma a definição de uma classe construindo uma regra que cubra muitos exemplos positivos e poucos ou nenhum negativo, então "separando" os exemplos recentemente cobertos e começando outra vez no restante. Uma regra é composta de um conseqüente e de uma parte antecedente ou corpo. O conseqüente é a classe predita. O corpo é uma conjunção de antecedentes, onde cada um deles é uma condição que envolve um único atributo. Para atributos simbólicos, essa condição é um simples teste de igualdade. Em alguns sistemas, a negação e a disjunção de valores são possíveis. Para atributos numéricos, a condição é tipicamente a inclusão em um intervalo. Diz-se que uma regra cobre um exemplo, e reciprocamente o exemplo satisfaz a regra, se todas as condições na regra forem verdadeiras para o exemplo. Dada uma classe, seus membros no conjunto de treinamento são chamados exemplos positivos, e o restante são exemplos negativos.

A escolha da heurística de avaliação H tem importância no desempenho do algoritmo. Dada uma regra, H deveria aumentar n_+, o número dos exemplos positivos que satisfazem a regra, e diminuir n_-, o número dos exemplos negativos que a

satisfazem. A série de algoritmos AQ (MICHALSKI *et al.*, 1986) usa a acurácia aparente (isto é, a acurácia da regra no conjunto de treinamento):

$$H(n_+, n_-) = \frac{n_+}{n_+ + n_-}$$

O sistema CN2 (CLARK & NIBLETT, 1989) originalmente usava a entropia da regra (QUINLAN, 1986). Entretanto, o problema com ambas essas medidas é que elas tendem a favorecer regras excessivamente específicas: elas alcançam seu valor máximo com uma regra que cobre um único exemplo. Isto pode ser superado pelo uso da correção de Laplace (NIBLETT, 1987):

$$H(n_+, n_-) = \frac{n_+ + 1}{n_+ + n_- + c}$$

onde c é o número de classes. Essa medida tende à acurácia incorreta quando a regra tem sustentação estatística forte (isto é, quando ela cobre muitos exemplos), mas tende a $1/c$ (isto é, "máxima ignorância") quando ela cobre poucos. A correção de Laplace é usada em versões recentes do CN2 (CLARK & BOSWELL, 1991).

A classificação de um novo exemplo é realizada fazendo-se o *match* de cada regra com ele, e selecionando aquelas que ele satisfizer. Se houver somente uma tal regra, sua classe será atribuída ao exemplo. Se não houver nenhuma, a solução geralmente adotada é usar uma "regra padrão" (isto é, atribuir ao exemplo a classe que ocorre mais freqüentemente no conjunto de treinamento, ou dentre aqueles exemplos não cobertos por nenhuma regra). Finalmente, se mais de uma regra cobrir o exemplo, então duas estratégias são possíveis. Uma delas é ordenar as regras em uma "lista de decisão", e selecionar somente a primeira regra que cobrir o exemplo (RIVEST, 1987). A outra é deixar que as diferentes regras votem, e selecionar a classe que receber a maioria de votos. Foi descoberto que o uso de regras não ordenadas geralmente produz

uma acurácia mais alta (CLARK & BOSWELL, 1991), e tem também a vantagem de uma compreensibilidade maior, uma vez que em uma lista de decisão cada corpo de regra está implicitamente em conjunção com as negações de todos aqueles que o precedem.

Os algoritmos de indução de regras são mais adequados a domínios simbólicos e podem freqüentemente descartar facilmente atributos irrelevantes.

Nos algoritmos de indução de regras que não lidam com ruído, a construção de uma nova regra pára somente quando todos os exemplos negativos são excluídos. Em algoritmos tolerantes a ruídos, uma medida de significância estatística pode ser usada para parar o crescimento (como no CN2), ou uma etapa de poda posterior pode remover os antecedentes e regras supérfluos. Os atributos irrelevantes tendem a não produzir nenhuma melhoria significativa na avaliação heurística, e assim são excluídos. Entretanto, atributos que são relevantes somente em combinação com outros atributos podem também ser descartados.

Uma outra deficiência da estratégia de "separação e conquista" é que ela faz com que um número cada vez menor de exemplos esteja disponível enquanto a indução progride, para cada regra e para as sucessivas regras. Esse efeito de fragmentação pode fazer com que as últimas regras, e os últimos antecedentes de cada regra, sejam induzidos com suporte estatístico insuficiente, levando a uma maior sensibilidade a ruídos e a regras ou antecedentes incorretos ou faltando. Isso por sua vez agrava o problema de *small disjuncts*, primeiro observado por HOLTE *et al.* (1989): as regras que cobrem poucos exemplos de treinamento tendem a ser altamente propensas a erro, mas removê-las freqüentemente aumenta o erro global ainda mais.

2.2. Testes Estatísticos Aproximados

Existem vários testes estatísticos aproximados usados para comparar a performance de algoritmos de aprendizado.

O teste *t* é um teste não tendencioso que possui a propriedade de que quando a média amostral é igual ao valor hipotético, segue uma distribuição *t* com $n - 1$ graus de liberdade, sem levar em consideração o valor da variância. Quando dois algoritmos são testados com mesmos conjuntos de treinamento e de teste, seus resultados não são independentes. O teste *t* usado quando as medidas não são independentes é chamado de teste *t* emparelhado.

Nesta seção são apresentados dois testes que já foram largamente utilizados por pesquisadores da área de aprendizado de máquinas: o teste *t* emparelhado com reamostragem e o teste *t* emparelhado com validação cruzada (DIETTERICH, 1998; MITCHELL, 1997). Antes porém, é feita uma revisão sobre teste de hipóteses, intervalos de confiança e testes *one-* e *two-tailed* (DeGROOT, 1986; HyperStat Online Textbook: http://davidmlane.com/hyperstat).

2.2.1. Teste de Hipóteses

O primeiro passo ao se testar uma hipótese é especificar a hipótese nula e uma hipótese alternativa. Se a pesquisa consiste em comparar as acurácias de dois algoritmos de aprendizado *A* e *B*, a hipótese nula é que não há diferença entre os algoritmos ($\overline{\delta}_A - \overline{\delta}_B = 0$). A hipótese alternativa seria: $\overline{\delta}_A - \overline{\delta}_B \neq 0$.

O segundo passo é selecionar o nível de significância desejado. Tipicamente são usados os níveis 0,05 ou 0,01.

O terceiro passo é calcular uma estatística do parâmetro especificado pela hipótese nula. No caso da comparação de dois algoritmos de aprendizado, para testar a diferença entre as acurácias de ambos seria computada a diferença na acurácia obtida para cada rodada (ou partição) e então seria feita a média desses valores:

$$\overline{\delta} = \frac{1}{n}\sum_{i=1}^{n}(\delta_A^{(i)} - \delta_B^{(i)})$$

onde $\delta_A^{(i)}$ e $\delta_B^{(i)}$ são, respectivamente, as acurácias obtidas pelos algoritmos *A* e *B* na *i*-ésima rodada (ou partição), e *n* é o número de rodadas (ou partições).

O quarto passo é calcular o valor de probabilidade (freqüentemente chamado de valor *p*) que é a probabilidade de se obter uma diferença entre a estatística computada a partir dos dados ($\overline{\delta}$) e o valor hipotético (μ) tão grande ou maior que a diferença obtida no experimento. Os cálculos são feitos assumindo-se que a hipótese nula seja verdadeira. Para isso deve-se computar a estatística *t*:

$$t = \frac{\overline{\delta} - \mu}{s_M}$$

onde $\overline{\delta}$ é uma estimativa do parâmetro em questão e μ é o valor hipotético da média da população, valor do parâmetro especificado na hipótese nula. Assume-se que o erro padrão da estatística (s_M) é estimado a partir dos dados e que a distribuição das amostras é normal. Essa fórmula é basicamente um método para calcular a área sob a curva normal. Se a média e o desvio padrão da distribuição das amostras de uma estatística são conhecidas, é fácil calcular a probabilidade de uma estatística ser maior ou menor

que um valor específico. O erro padrão estimado da média (s_M) é computado usando a seguinte fórmula:

$$s_M = \frac{s}{\sqrt{n}}$$

onde n é o tamanho da amostra e s é uma estimativa do desvio padrão dada pela fórmula:

$$s = \sqrt{\frac{\sum_{i=1}^{n} (\delta_i - \bar{\delta})^2}{n-1}}$$

onde $\delta_i = \delta_A^{(i)} - \delta_B^{(i)}$. Uma tabela da distribuição t pode então ser usada para calcular o valor de probabilidade (p) tendo como entradas t e $n - 1$ (graus de liberdade). O teste de significância da diferença entre as acurácias de dois algoritmos de aprendizado consiste em determinar se a média das diferenças para cada rodada (ou partição) é significativamente diferente de zero. Se for, então a diferença entre as acurácias dos algoritmos é significativa. A fórmula para determinar se a média das diferenças entre as acurácias de dois algoritmos de aprendizado é significativamente diferente de zero é:

$$t = \frac{\bar{\delta} - 0}{s_M}$$

com $n - 1$ graus de liberdade, onde n é o número de rodadas (ou partições).

O valor de probabilidade p computado é comparado com o nível de significância escolhido. Se p for menor ou igual ao nível de significância, então diz-se que o resultado é "estatisticamente significativo". Se a probabilidade for maior que o nível de significância, diz-se que o resultado é "não estatisticamente significativo" ou "estatisticamente insignificante".

Se o resultado for estatisticamente significativo, então a hipótese nula é rejeitada em favor da hipótese alternativa. Caso contrário, a hipótese nula não é rejeitada. Se a hipótese nula rejeitada fosse $\overline{\delta}_A - \overline{\delta}_B = 0$, então a hipótese alternativa aceita seria $\overline{\delta}_A \neq \overline{\delta}_B$.

2.2.2. Intervalos de Confiança

O intervalo de confiança para a média quando o desvio padrão é estimado, é dado por:

$$\overline{\delta} \pm t \cdot s_M$$

onde $\overline{\delta}$ é a média das amostras, s_M é uma estimativa do desvio padrão da média e t depende dos graus de liberdade e do nível de confiança escolhido.

O valor de t pode ser determinado a partir de uma tabela da distribuição t. O número de graus de liberdade para t é igual ao número de graus de liberdade para a estimativa do desvio padrão da média ($n - 1$).

Existe um relacionamento extremamente próximo entre intervalos de confiança e teste de hipóteses. Quando um intervalo de confiança de 95% é construído, todos os valores no intervalo são considerados valores plausíveis para o parâmetro sendo estimado. Valores fora do intervalo são rejeitados como não plausíveis.

Se o valor do parâmetro especificado pela hipótese nula estiver contido no intervalo de confiança de 95% estimado para tal parâmetro, então a hipótese nula não pode ser rejeitada ao nível 0,05. Se o valor especificado pela hipótese nula não estiver no intervalo, então a hipótese nula pode ser rejeitada ao nível 0,05 ou menor.

Similarmente, se um intervalo de 99% de confiança é construído, então os valores fora do intervalo são rejeitados ao nível 0,01. A hipótese nula não é rejeitada se o valor de parâmetro especificado por ela estiver no intervalo, uma vez que a hipótese nula seria plausível.

A hipótese nula é usualmente uma hipótese de não diferença. Assim, hipóteses em que o valor hipotético é zero são as mais comuns. Quando o valor hipotético é zero existe um relacionamento simples entre teste de hipótese e intervalos de confiança: se o intervalo contém zero então a hipótese nula não pode ser rejeitada ao nível de confiança determinado; se o intervalo não contém zero então a hipótese nula pode ser rejeitada.

Esse é apenas um caso especial da regra geral que declara que a hipótese nula pode ser rejeitada se o intervalo não contiver o valor hipotético e não pode ser rejeitada se o intervalo contiver o valor hipotético.

Sempre que um teste de significância rejeita a hipótese nula de que um parâmetro é zero, todos os valores no intervalo de confiança serão positivos ou todos os valores no intervalo de confiança serão negativos, evidenciando a direção do efeito.

Para analisar a diferença entre a performance de dois algoritmos de aprendizado, a hipótese nula é que as acurácias médias dos dois algoritmos são iguais, ou seja, $\overline{\delta}_A - \overline{\delta}_B = 0$. O intervalo de confiança da diferença entre as acurácias médias é então computado. Somente valores no intervalo são considerados plausíveis para a diferença entre as médias das acurácias dos dois algoritmos. Se o valor especificado pela hipótese nula (zero), não estiver no intervalo, a hipótese de que não há diferença entre as acurácias médias dos dois algoritmos pode ser rejeitada ao nível de significância escolhido.

2.2.3. Testes *One-tailed* e *Two-tailed*

Uma probabilidade computada considerando as diferenças em ambas as direções é chamada de probabilidade *two-tailed*. Existem situações onde apenas uma direção é considerada. Por exemplo, ao se comparar as acurácias médias de dois algoritmos, pode-se querer saber se $\overline{\delta}_A - \overline{\delta}_B$ é ou não maior que zero. Entretanto, se $\overline{\delta}_A - \overline{\delta}_B$ não for maior que zero, o pesquisador pode não se importar se é igual a zero ou menor que zero. Quando somente uma direção é de interesse do pesquisador, então um teste *one-tailed* pode ser feito. Valores de probabilidade para testes *one-tailed* são sempre metade do valor para testes *two-tailed* se o efeito é na direção especificada. Testes *one-tailed* são usados quando o pesquisador prediz a direção do efeito de antemão. Esse uso dos testes *one-tailed* é questionável porque o pesquisador pode somente rejeitar a hipótese nula se o efeito estiver na direção predita. Se o efeito estiver na outra direção, então a hipótese nula não pode ser rejeitada, não importando o quão forte seja o efeito. Freqüentemente, o aspecto mais interessante de um efeito é que ele é contrário às expectativas. Portanto, um pesquisador que se compromete a ignorar efeitos em uma direção pode ser forçado a escolher entre ignorar um resultado potencialmente importante e usar técnicas de inferência estatística desonestamente. Testes *one-tailed* não são usados com freqüência. A menos que seja indicado o contrário, um teste deve ser assumido ser *two-tailed*.

2.2.4. Teste *t* Emparelhado com Reamostragem

Este é atualmente um dos métodos de teste mais populares na literatura de aprendizado de máquinas.

Uma série de usualmente 30 rodadas é conduzida. Em cada rodada, o conjunto de dados disponível D é dividido aleatoriamente em um conjunto de treinamento R de um tamanho específico (tipicamente dois terços dos dados) e um conjunto de teste T. Os algoritmos de aprendizado A e B são ambos treinados com R e os classificadores resultantes são testados em T.

Sejam $\delta_A^{(i)}$ e $\delta_B^{(i)}$ as proporções observadas de exemplos de teste que os algoritmos A e B, respectivamente, classificaram corretamente durante a rodada i. Se for assumido que as 30 diferenças $\delta_i = \delta_A^{(i)} - \delta_B^{(i)}$ foram tiradas independentemente de uma distribuição normal, então pode-se aplicar o teste t de *Student*, computando a estatística

$$t = \frac{\overline{\delta} \cdot \sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n} (\delta_i - \overline{\delta})^2}{n-1}}},$$

onde $\overline{\delta} = \frac{1}{n} \sum_{i=1}^{n} \delta_i$.

Sob a hipótese nula, essa estatística tem uma distribuição t com $n - 1$ graus de liberdade.

Há muitas desvantagens em potencial nessa abordagem. Primeiro, as diferenças individuais δ_i não têm uma distribuição normal, uma vez que $\delta_A^{(i)}$ e $\delta_B^{(i)}$ não têm distribuição normal. Segundo, os δ_i's não são independentes, pois há sobreposição dos

conjuntos de teste nas rodadas, bem como há sobreposição dos conjuntos de treinamento nas rodadas. Essas violações nas suposições básicas do teste *t* causam grandes problemas que tornam este teste inseguro para ser usado. O algoritmo do teste *t* emparelhado com reamostragem é apresentado na Tabela 1.

2.2.5. Teste *t* Emparelhado com Validação Cruzada

Este teste é idêntico ao anterior exceto que ao invés de construir cada par de conjuntos de treinamento e de teste dividindo *D* aleatoriamente em *R* e *T*, o conjunto de dados *D* é dividido aleatoriamente em *n* conjuntos disjuntos de igual tamanho, T_1, ..., T_n. Então são conduzidas *n* rodadas. Em cada rodada, o conjunto de teste é T_i e o conjunto de treinamento é a união de todos os outros T_j, $j \neq i$. A mesma estatística *t* é computada.

A vantagem dessa abordagem é que cada conjunto de teste é independente dos outros. Entretanto, este teste ainda sofre do problema que os conjuntos de treinamento se sobrepõem. Em uma validação cruzada com 10 partições, cada par de conjuntos de treinamento compartilham 80% dos exemplos. Essa sobreposição pode evitar que o teste estatístico obtenha uma boa estimativa da quantidade de variação que seria observada se cada conjunto de treinamento fosse completamente independente dos outros conjuntos de treinamento. O algoritmo do teste *t* emparelhado com validação cruzada é mostrado na Tabela 2.

Considerando as desvantagens do teste *t* emparelhado com reamostragem, o método de teste utilizado nesta pesquisa foi o teste *t* emparelhado com validação cruzada com *n* partições.

Tabela 1. Teste *t* emparelhado com reamostragem.

Entrada: D é o conjunto de dados, n é o número de rodadas, usualmente 30, *sig* é o nível de significância desejado, tipicamente 0,05 ou 0,01.

Saída: $\overline{\delta}$ é a diferença média entre as acurácias dos algoritmos comparados.

Procedimento Reamostragem (D, n, *sig*)

Para $i = 1$ até n,

Particione D aleatoriamente em um conjunto de treinamento R, com dois terços dos elementos de D, e um conjunto de teste T, com o restante dos elementos.

Sejam A e B os algoritmos de aprendizado a serem comparados.

$h_A = A(R)$

$h_B = B(R)$

δ_i = Acurácia(T, h_A) – Acurácia(T, h_B)

$$\overline{\delta} = \frac{1}{n}\sum_{i=1}^{n}\delta_i$$

$$t = \frac{\overline{\delta}\cdot\sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n}(\delta_i - \overline{\delta})^2}{n-1}}}$$

p = Tabela-t (t, $n - 1$)

Se $p \leq sig$ então Reporte "Diferença estatisticamente significativa".

Senão Reporte "Diferença não estatisticamente significativa".

Retorne $\overline{\delta}$.

Tabela 2. Teste *t* emparelhado com validação cruzada.

Entrada: D é o conjunto de dados, n é o número de partições, usualmente 10, *sig* é o nível de significância desejado, tipicamente 0,05 ou 0,01.

Saída: $\bar{\delta}$ é a diferença média entre as acurácias dos algoritmos comparados.

Procedimento ValidaçãoCruzada (D, n, *sig*)

Particione D aleatoriamente em n subconjuntos disjuntos T_1, T_2, ..., T_n de igual tamanho, onde esse tamanho é pelo menos 30.

Para $i = 1$ até n, faça

Use T_i como conjunto de teste e o restante dos dados como conjunto de treinamento R_i.

Sejam A e B os algoritmos de aprendizado a serem comparados.

$h_A = A(R)$

$h_B = B(R)$

δ_i = Acurácia(T, h_A) – Acurácia(T, h_B)

$$\bar{\delta} = \frac{1}{n}\sum_{i=1}^{n}\delta_i$$

$$t = \frac{\bar{\delta}\cdot\sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n}(\delta_i - \bar{\delta})^2}{n-1}}}$$

p = Tabela-t (t, $n - 1$)

Se $p \leq sig$ então Reporte "Diferença estatisticamente significativa".

Senão Reporte "Diferença não estatisticamente significativa".

Retorne $\bar{\delta}$.

2.3. O Algoritmo RISE

O algoritmo RISE (*Rule Induction from a Set of Exemplars* - indução de regras a partir de um conjunto de exemplares) é um método de aprendizado multi-estratégia que tenta superar algumas das limitações do aprendizado baseado em instâncias e do aprendizado baseado na indução de regras, unificando os dois (DOMINGOS, 1994).

2.3.1. Definições

Uma regra no RISE é composta de um conseqüente, que é a classe predita, e de uma parte antecedente, que é uma conjunção de condições. Cada condição envolve somente um atributo: para atributos simbólicos é um teste da igualdade (por exemplo, cor = verde) e para atributos numéricos é a pertinência em um intervalo fechado em ambos os lados (por exemplo, $3 \leq x \leq 7$). Em cada regra há no máximo uma condição envolvendo cada atributo, podendo haver nenhuma. Uma instância é simplesmente uma regra em que o conseqüente é a classe da instância, existe exatamente uma condição por atributo e todos os intervalos são degenerados (por exemplo, $4 \leq x \leq 4$, isto é, $x = 4$).

O RISE classifica um novo exemplo atribuindo-lhe a classe da regra mais próxima na base de conhecimento. A distância entre uma regra e um exemplo é definida como segue. Seja $E = (e_1, e_2, ..., e_a, C_E)$ um exemplo com valor e_i para o i-ésimo atributo (a_i) e classe C_E. Seja $R = (A_1, A_2, ..., A_a, C_R)$ uma regra com classe C_R e condição A_i no i-ésimo atributo. Se não houver condição no i-ésimo atributo, A_i = True (verdadeiro),

caso contrário A_i é $e_i = r_i$ se o atributo for simbólico ou A_i é $r_{i,lower} \leq e_i \leq r_{i,upper}$ se o atributo for numérico. A distância $\Delta(R, E)$ entre R e E é então definida como:

$$\Delta(R, E) = \sum_{i=1}^{a} \delta^2(i)$$

onde a componente da distância $\delta(i)$ para o i-ésimo atributo é:

$$\delta(i) = \begin{cases} 0 & \text{se } A_i = True \\ SVDM(r_i, e_i) & \text{se } a_i \text{ é simbólico e } A_i \neq True \\ \delta_{num}(i) & \text{se } a_i \text{ é numérico e } A_i \neq True \end{cases}$$

onde por sua vez $SVDM(r_i, e_i)$ é a métrica de diferença de valores simplificada, uma versão simplificada da *VDM*, que define a distância entre dois valores simbólicos como:

$$SVDM(r_i, e_i) = \sum_{h=1}^{c} \left| P(C_h \mid r_i) - P(C_h \mid e_i) \right|$$

onde c é o número de classes, C_h é a h-ésima classe, $P(C_h \mid r_i)$ é a proporção de regras cuja classe é C_h, dado que têm o i-ésimo atributo com valor r_i, $P(C_h \mid e_i)$ é a proporção de exemplos de treinamento cuja classe é C_h, dado que têm o i-ésimo atributo com valor e_i, e:

$$\delta_{num}(i) = \begin{cases} 0 & \text{se } r_{i,lower} \leq e_i \leq r_{i,upper} \\ \dfrac{e_i - r_{i,upper}}{e_{i,\max} - e_{i,\min}} & \text{se } e_i > r_{i,upper} \\ \dfrac{r_{i,lower} - e_i}{e_{i,\max} - e_{i,\min}} & \text{se } e_i < r_{i,lower} \end{cases}$$

$e_{i,max}$ e $e_{i,min}$ são respectivamente os valores máximo e mínimo para o atributo encontrados no conjunto de treinamento.

A distância de um valor numérico desconhecido a qualquer outro é definida como 0. Se o valor de um atributo simbólico estiver faltando, é atribuído a ele o valor especial "?". Esse valor é tratado como um valor simbólico legítimo e sua *SVDM* a

todos os outros valores do atributo é computada e usada. Um valor desconhecido é considerado aproximadamente equivalente a um dado valor possível se ele se comporta similarmente a este, e distinto deste valor caso contrário.

Se duas ou mais regras estiverem igualmente próximas a um exemplo, o RISE seleciona como ganhadora a regra com a maior acurácia de Laplace. Isso significa que regras nem muito gerais e nem muito específicas são favorecidas desnecessariamente. Ao invés disso, a preferência vai para as regras com alta acurácia aparente, bem como forte suporte estatístico. No caso de acurácias iguais, o RISE escolhe a classe mais freqüente dentre aquelas representadas, e se ainda houver empate, a vencedora é escolhida aleatoriamente.

Uma regra cobre um exemplo se todas as suas condições forem verdadeiras para o exemplo. Uma regra ganha um exemplo se ela for a regra mais próxima ao exemplo de acordo com a métrica de distância e a política da resolução de conflitos descritas. Uma regra pode cobrir um exemplo e não o ganhar.

A acurácia de um conjunto de regras é definida como a fração dos exemplos que são classificados corretamente. Um conjunto de regras classifica um exemplo corretamente quando a regra mais próxima ao exemplo tiver a mesma classe que ele.

2.3.2. Funcionamento

O RISE induz todas as regras em paralelo (não utiliza o algoritmo clássico de cobertura). Além disso, a avaliação heurística é realizada para todo o conjunto de regras de uma vez. Mudanças em uma regra são avaliadas em termos do seu efeito na acurácia

global do conjunto de regras. Isso é feito com o objetivo de atenuar o problema de fragmentação o tanto quanto possível.

O sentido da busca no RISE é específico-geral. As regras são generalizadas descartando-se condições para atributos simbólicos e alargando intervalos para os numéricos. A Tabela 3 mostra o algoritmo de como uma regra é generalizada minimamente para cobrir um exemplo previamente fora de seu escopo. Todas as condições de atributos simbólicos que não são satisfeitas pelo exemplo são descartadas, e todos os intervalos são estendidos para incluir o valor do atributo do exemplo na borda, se necessário. Considera-se que um valor numérico desconhecido faz o *match* com qualquer outro valor, e um valor simbólico desconhecido no exemplo ou na regra (mas não em ambos) faz com que a condição correspondente seja descartada.

O RISE repetidamente encontra o exemplo mais próximo a regra, que seja da mesma classe e ainda não tenha sido coberto por esta, e então tenta generalizar minimamente a regra para cobrir tal exemplo. Se a generalização da regra tem efeito positivo ou nulo na acurácia global, a mudança é mantida. O processo pára quando nenhuma mudança produz melhora. O algoritmo RISE é apresentado na Tabela 4. O conjunto de regras inicial é o conjunto de treinamento (isto é, cada instância é candidata à regra). O conjunto de regras final pode conter algumas instâncias não generalizadas bem como regras mais abstratas. Se com a generalização duas regras tornam-se idênticas, elas são "fundidas". Em cada laço o novo conjunto de regras é adotado mesmo se sua acurácia aparente for igual à do antigo. Isso é uma aplicação direta da *Occam's Razor*: quando duas teorias parecem ter o mesmo desempenho, prefira a mais simples.

Em cada passo somente a mudança na acurácia necessita ser considerada. Cada exemplo memoriza qual é a regra mais próxima (isto é, a regra que o ganha) e a distância até ela. Com essa informação, tudo que é necessário quando uma regra é generalizada é fazer o *match* dessa única regra com todos os exemplos, e verificar se ela ganha algum que não ganhava antes. Se um exemplo que era classificado errado for agora classificado corretamente, o numerador de Acurácia(*RS*), que é definida como a fração dos exemplos que são classificados corretamente, é incrementado. Se o inverso ocorrer, ele é decrementado. Caso contrário nenhuma mudança é feita. Se a soma dos incrementos e decrementos for maior ou igual a 0, a regra nova é adotada, e as estruturas relevantes são atualizadas. Senão ela é rejeitada.

Tabela 3. Generalização de uma regra para cobrir um exemplo.

Entradas: $R = (A_1, A_2, ..., A_a, C_R)$ é uma regra, $E = (e_1, e_2, ..., e_a, C_E)$ é um exemplo.

A_i é True, $e_i = r_i$, ou $r_{i,lower} \leq e_i \leq r_{i,upper}$.

Saída: $R' = (A'_1, A'_2, ..., A'_a, C'_R)$ é a regra generalizada.

Procedimento Generalização-Mais-Específica (R, E)

Seja $R' = R$.

Para cada atributo a_i,

Se A_i = True então nada é feito.

Senão se a_i for simbólico e $e_i \neq r_i$ então A'_i = True.

Senão se a_i for numérico e $e_i > r_{i,upper}$ então $r'_{i,upper} = e_i$.

Senão se a_i for numérico e $e_i < r_{i,lower}$ então $r'_{i,lower} = e_i$.

Retorne R'.

Tabela 4. O algoritmo RISE.

Entrada: *ES* é o conjunto de treinamento.

Saída: *RS* é o conjunto de regras.

Procedimento RISE (*ES*)

Seja *RS* = *ES*.

Compute Acurácia (*RS*).

Repita

Para cada regra *R* em *RS*,

Encontre o exemplo *E* mais próximo a *R* ainda não coberto por ela, e da mesma classe de *R*.

Seja *R'* = Generalização-Mais-Específica (*R*, *E*).

Seja *RS'* = *RS* com *R* substituído por *R'*.

Se Acurácia (*RS'*) ≥ Acurácia (*RS*) então

Substitua *RS* por *RS'*,

Se *R'* for idêntica a uma outra regra em *RS* então

Apague *R'* de *RS*.

Até que nenhum aumento em Acurácia (*RS*) seja obtido.

Retorne *RS*.

2.3.3. Complexidade Temporal

A complexidade temporal de pior caso do RISE é comparável à de outros algoritmos de indução de regras.

Seja e o número de exemplos no conjunto de treinamento, a o número de atributos de cada exemplo, v_s o número médio de valores observados por o atributo simbólico, v_n o número médio de valores observados por atributo numérico, r o número de regras, e c o número de classes.

Suponha por enquanto que todos os atributos são simbólicos.

A fase de inicialização do algoritmo consiste de três operações: copiar os exemplos para as regras ($O(ea)$), compilar uma tabela de distâncias *SVDM* ($O(ea + av_s^2c)$) e encontrar a regra mais próxima de cada exemplo, computando a acurácia inicial do conjunto de regras ($O(e^2a)$). O tempo total necessário para a inicialização é portanto $O(e^2a + av_s^2c)$.

O coração do algoritmo consiste de quatro etapas: encontrar o exemplo mais próximo de uma regra ($O(ea)$), generalizar a regra para cobri-lo ($O(a)$), comparar a regra alterada com todos os exemplos para ver se alguns deles são ganhos com a alteração ($O(ea)$), e (se a mudança for adotada) comparar a regra com todas as outras regras para verificar se há duplicações ($O(ra)$). Essas operações consomem o tempo total de $O(ea + ra)$. Uma vez que cada laço "Repita" consiste em realizar essas quatro etapas para todas as r regras, cada laço leva no pior caso tempo $O[r\ (ea + ra)]$. Como no RISE cada exemplo produz no máximo uma regra, $r \leq c$ e no pior caso esse tempo é $O(e^2a)$.

O laço "Repita" pode ser realizado até $O(ea)$ vezes, pois no pior caso somente uma condição de uma regra será descartada em cada laço inteiro, fazendo com que alguma mudança em uma outra regra se torne possível no próximo laço. Multiplicando esse valor pelo custo de um único laço, obtém-se a complexidade temporal $O(e^3a^2)$. Uma vez que $e \geq v_s$ e considerando que em geral $a \geq c$, esse valor ($O(e^3a^2)$) domina a complexidade do algoritmo, sendo o limite superior da complexidade temporal de pior caso do RISE.

A introdução de valores numéricos simplesmente aumenta os valores anteriores por um fator de v_n, uma vez que a remoção de uma condição, que era realizada em apenas um passo, pode agora ser substituída pela expansão do intervalo correspondente, que pode ser realizada em no máximo $O(v_n)$ passos.

3. O ALGORITMO SUNRISE

Apesar de alcançar uma acurácia preditiva muito boa, o algoritmo RISE é lento quando aplicado a conjuntos de dados grandes. O SUNRISE é um algoritmo derivado do algoritmo RISE, que tenta superar esse problema.

Na Seção 3.1 é descrita a idéia básica do SUNRISE. Na Seção 3.2 é feita uma pequena análise da complexidade temporal do algoritmo desenvolvido. Na Seção 3.3 são discutidas as limitações do SUNRISE. Na Seção 3.4 é apresentado um estudo empírico extensivo que compara o SUNRISE com seu predecessor. Na Seção 3.5 é apresentado o TextSUNRISE, um algoritmo desenvolvido a partir do SUNRISE para a aplicação na área de mineração de texto.

3.1. Tendência a Aprendizado Baseado em Instâncias

Analisando os resultados dos experimentos conduzidos com o RISE e apresentados em (DOMINGOS, 1996), verifica-se que seu subsistema IBL quase sempre obtém acurácia maior que seu subsistema de indução de regras. O SUNRISE tem características que fazem com que seus resultados tenham tendência a parecerem com os de um sistema IBL.

O SUNRISE tenta generalizar as regras mais de uma vez antes de incluí-las no conjunto de regras e só aceita as mudanças se o efeito na acurácia global for estritamente positivo, ou seja, uma nova regra somente é adicionada ao conjunto de regras se o conjunto alcançar uma acurácia maior que antes de sua inclusão. Isso induz a

geração de regras muito mais específicas. Dessa forma, o SUNRISE tende a se comportar mais como um algoritmo IBL do que como um algoritmo de indução de regras, conseguindo assim uma das maiores vantagens dos métodos IBL, que é a velocidade, sem abandonar a vantagem da abstração das regras.

Após uma generalização com efeito positivo na acurácia global do conjunto de regras, em vez de aceitar as mudanças e recomeçar o processo com outra regra, o SUNRISE tenta generalizar a regra novamente e isso pode acontecer até um número máximo de vezes. Esse número é o valor *k*, parâmetro do SUNRISE cujo valor deve ser determinado experimentalmente. A regra só é generalizada novamente se sua distância em relação ao exemplo mais próximo for menor que a distância da regra em relação ao exemplo mais próximo a ela antes de ser generalizada. Isso faz com que a regra seja generalizada apenas com exemplos cada vez mais próximos. O algoritmo SUNRISE é apresentado na Tabela 5.

Tabela 5. O algoritmo SUNRISE.

Entrada: *ES* é o conjunto de treinamento, *k* é o parâmetro do SUNRISE.

Saída: *RS* é o conjunto de regras.

Procedimento SUNRISE (*ES*, *k*)

Seja *RS* = *ES*.

Compute Acurácia (*RS*).

Repita

Para cada regra *R* em *RS*,

Repita *k* vezes

Encontre o exemplo *E* mais próximo a *R* ainda não coberto por ela, e da mesma classe de *R*.

Seja *R'* = Generalização-Mais-Específica (*R*, *E*).

Seja *RS'* = *RS* com *R* substituído por *R'*.

Se Acurácia (*RS'*) > Acurácia (*RS*) então

R = *R'*

Senão Quebre o laço

Se Acurácia (*RS'*) > Acurácia (*RS*) então

Substitua *RS* por *RS'*,

Se *R'* for idêntica a uma outra regra em *RS* então

Apague *R'* de *RS*.

Até que nenhum aumento em Acurácia (*RS*) seja obtido.

Retorne *RS*.

3.2. Complexidade Temporal

A complexidade temporal de pior caso do SUNRISE é a mesma do RISE, uma vez que a complexidade de ambos difere apenas de uma constante multiplicativa *k*.

Apesar de possuírem a mesma complexidade temporal de pior caso, isso não significa que na prática os dois algoritmos terão o mesmo tempo de execução.

3.3. Limitação

Como o SUNRISE tem características que fazem com que ele se comporte mais como um sistema IBL, sua sensibilidade a atributos irrelevantes e ruído é maior que a do RISE. Identificar e eliminar ou atributos irrelevantes ou exemplos de treinamento não confiáveis antes da construção de um classificador são técnicas usadas para ajudar a melhorar o processo de indução. Tais métodos de limpeza podem ser mais custosos que o próprio processo de construir um classificador. Além disso, são tomadas decisões irreversíveis ao se remover atributos ou exemplos.

A abordagem *wrapper* (KOHAVI & JOHN, 1997) tenta identificar o melhor subconjunto de atributos para serem usados com um algoritmo em particular, e pode ser usado para superar a limitação do SUNRISE.

3.4. Avaliação Experimental

Foi realizada uma avaliação experimental extensiva com o objetivo de comparar o desempenho do novo algoritmo, o SUNRISE, com o de seu predecessor, o algoritmo RISE. Esta seção descreve as características e relata os resultados dos experimentos.

3.4.1. Conjuntos de Dados

Nos experimentos foram usados 24 conjuntos de dados, numa tentativa de incluir vários domínios, tamanhos e dificuldades. Todos os conjuntos de dados foram extraídos do repositório da UCI (MURPHY & AHA, 1995). Na Tabela 6 são apresentados os conjuntos de dados utilizados e um resumo de suas principais características: o número de exemplos, o número total de atributos, o número de atributos numéricos, o número de classes, a porcentagem de valores desconhecidos, e se ou não o conjunto de dados contém exemplos inconsistentes, isto é, exemplos idênticos com classes diferentes.

Tabela 6. Conjuntos de dados usados no estudo empírico.

Domínio	Exs.	Atribs.	Num.	Classes	Descon.	Incons.
Annealing	798	38	9	5	64,9	Não
Audiology	200	69	0	24	2,1	Não
Chess endgames	3196	36	0	2	0,0	Não
Credit screening	690	15	6	2	0,6	Não
DNA promoters	106	57	0	2	0,0	Não
Echocardiogram	131	7	6	2	4,4	Sim
Glass	214	9	9	6	0,0	Não
Heart disease	303	13	6	2	0,2	Não
Hepatitis	155	19	6	2	5,7	Não
Horse colic	300	22	7	2	24,3	Sim
Iris	150	4	4	3	0,0	Não
LED	100	7	0	10	0,0	Sim
Liver disease	345	6	6	2	0,0	Não
Mushroom	8124	22	0	2	1,4	Não
Pima diabetes	768	8	8	2	0,0	Não
Post-operative	90	8	8	3	0,4	Sim
Solar flare	323	12	3	6	0,0	Sim
Sonar	208	60	60	2	0,0	Não
Soybean	47	35	0	4	0,0	Não
Splice junctions	3190	60	0	3	0,0	Sim
Thyroid disease	3163	25	7	2	6,7	Sim
Voting records	435	16	0	2	5,6	Não
Wine	178	13	13	3	0,0	Não
Zoology	101	16	1	7	0,0	Não

3.4.2. Metodologia Experimental

O método de teste utilizado nos testes experimentais do RISE (DOMINGOS, 1995) foi o teste *t* emparelhado com reamostragem, onde cada par de conjuntos de treinamento e de teste é construído dividindo-se aleatoriamente o conjunto de dados. Uma vez que esse método é fortemente criticado por DIETTERICH (1998), ele não foi usado nos experimentos com o SUNRISE. Segundo Dietterich, nessa abordagem as violações nas suposições básicas do teste *t* causam grandes problemas que tornam o teste inseguro para ser usado.

O método de teste utilizado nesta pesquisa foi o teste *t* emparelhado *two-tailed* com validação cruzada com *n* partições. Para os conjuntos de dados com mais de 299 exemplos, $n = 10$. Para os demais conjuntos de dados, *n* foi calculado de forma a fornecer o maior número possível de partições sem violar a suposição do teste *t* de que os conjuntos de teste devem possuir no mínimo 30 exemplos.

3.4.3. Ajuste do Parâmetro *k*

Antes de comparar a performance dos dois algoritmos, foram realizados testes experimentais com os conjuntos de dados para ajustar o parâmetro *k* do algoritmo SUNRISE.

Para ajustar o parâmetro *k*, foi feita uma validação cruzada interna, onde cada conjunto de treinamento R_i de cada partição foi redividido em um subconjunto de treinamento R'_j e um conjunto de validação V_j (MITCHELL, 1997; KOHAVI, 1995).

Variando-se o parâmetro, o algoritmo foi então treinado no subconjunto de treinamento R'_j e testado no conjunto de validação para cada subpartição. A configuração que forneceu a melhor acurácia no menor tempo na classificação dos exemplos do conjunto de validação foi então registrada. Após realizar esse procedimento para cada partição, o algoritmo foi então treinado com o conjunto de treinamento R_i e testado com o conjunto de teste T_i. Dentre as configurações já registradas, a que forneceu maior acurácia no menor tempo na classificação dos exemplos conjunto de teste foi então considerada a melhor e mantida. Isso foi feito para cada conjunto de dados, e o valor de k que alcançou a melhor performance com a maioria dos conjuntos de dados foi $k = 3$.

É interessante notar que para $k = 0$, o SUNRISE se comporta como um algoritmo IBL. No caso $k = 1$, algoritmo SUNRISE é praticamente igual ao RISE, exceto pelo fato de que uma nova regra só é aceita se houver aumento na acurácia global do conjunto de regras, ou seja, não é mais aplicada a *Occam's Razor* (DOMINGOS, 1996).

3.4.4. Resultados Experimentais

Na Tabela 7 são apresentados os tempos de execução de ambos os algoritmos para cada um dos conjuntos de dados testados. Todos os valores são médias de 50 execuções em um computador Pentium MMX 200MHz com 64MBytes de memória RAM. Comparando os tempos médios de execução dos dois algoritmos, pôde-se verificar que o SUNRISE é bem mais rápido que seu predecessor na grande maioria dos testes.

As acurácias médias são apresentadas na Tabela 8. Embora o RISE apresente acurácia média maior que o SUNRISE em alguns conjuntos de dados testados, em poucos deles a diferença observada entre os algoritmos é estatisticamente significativa (conjunto de dados LED, por exemplo). Em compensação, o SUNRISE também obteve algumas vitórias sobre o RISE com diferença estatisticamente significativa, com nível de significância de até 0,01 (conjunto de dados Sonar, por exemplo).

Um dos motivos da baixa performance do SUNRISE aplicado ao conjunto de dados LED é a presença de inconsistências nos dados contidos nesse conjunto. Tais dados constituem ruído, e o baixo desempenho evidencia a sensibilidade do SUNRISE a dados ruidosos.

Tabela 7. Tempos de execução (em segundos).

Domínio	RISE	SUNRISE			
		$k = 0$	$k = 1$	$k = 2$	$k = 3$
Annealing	150,846	9,802	37,549	53,538	57,164
Audiology	7,895	1,686	3,939	6,631	6,686
Chess endgames	3761,120	461,285	3731,835	5789,087	6126,340
Credit screening	571,835	16,010	95,846	147,109	190,571
DNA promoters	2,176	0,528	0,968	1,297	1,187
Echocardiogram	2,041	0,173	0,832	1,107	1,217
Glass	6,741	0,807	3,445	5,423	5,532
Heart disease	54,692	3,208	15,626	28,373	31,560
Hepatitis	14,104	0,532	2,291	3,719	3,884
Horse colic	166,615	3,923	18,923	35,681	44,967
Iris	2,730	0,258	0,587	0,972	1,137
LED	6,670	1,615	6,505	8,593	8,483
Liver disease	52,769	3,153	19,032	29,637	36,340
Mushroom	12314,087	2191,120	4439,362	5432,494	5492,000
Pima diabetes	794,307	14,857	119,747	196,065	233,593
Post-operative	0,253	0,034	0,198	0,198	0,253
Solar flare	15,351	1,945	8,813	18,098	16,560
Sonar	40,203	2,840	10,972	16,796	23,664
Soybean	19,802	5,791	14,417	23,593	28,593
Splice junctions	50632,605	1392,824	7515,900	14608,043	14690,077
Thyroid disease	51502,605	291,615	4435,736	6708,812	6936,615
Voting records	44,197	5,021	24,857	45,571	52,714
Wine	12,675	0,917	2,785	4,049	4,983
Zoology	0,253	0,089	0,143	0,198	0,253

Tabela 8. Acurácias médias (intervalo de confiança de 95%).

Domínio	RISE	SUNRISE			
		$k = 0$	$k = 1$	$k = 2$	$k = 3$
Annealing	97,1 ±1,7	97,3 ±1,4	98,4 ±1,0	98,2 ±1,2	98,4 ±1,0
Audiology	80,5 ±5,5	78,0 ±3,3	76,5 ±1,6	79,0 ±8,3	79,5 ±5,9
Chess endgames	98,4 ±0,5	97,0 ±0,6	97,7 ±0,7	98,1 ±0,5	98,1 ±0,5
Credit screening	82,8 ±2,4	81,7 ±2,1	84,6 ±2,6	83,7 ±2,6	83,7 ±2,7
DNA promoters	86,6 ±4,0	91,4 ±0,0	83,8 ±14,7	85,7 ±18,7	80,0 ±18,7
Echocardiogram	63,2 ±4,7	60,9 ±9,5	70,3 ±14,9	64,8 ±10,2	64,8 ±10,2
Glass	64,7 ±11,1	67,1 ±9,6	68,0 ±10,7	61,9 ±10,0	62,3 ±7,3
Heart disease	79,9 ±5,3	79,3 ±4,8	83,0 ±5,4	81,6 ±6,6	81,6 ±6,2
Hepatitis	80,6 ±4,0	82,5 ±3,5	81,2 ±4,3	84,5 ±5,9	84,5 ±3,3
Horse colic	83,9 ±4,6	76,3 ±5,9	83,6 ±3,2	84,3 ±2,5	85,3 ±4,7
Iris	95,9 ±4,5	95,9 ±4,5	95,3 ±4,7	95,3 ±4,7	95,3 ±4,7
LED	70,3 ±4,8	55,0 ±6,5	67,0 ±6,8	56,9 ±6,5	57,3 ±6,4
Liver disease	62,3 ±4,1	63,2 ±5,5	61,4 ±4,9	64,7 ±5,6	64,1 ±4,6
Mushroom	100,0 ±0,0	100,0 ±0,0	100,0 ±0,0	100,0 ±0,0	100,0 ±0,0
Pima diabetes	72,6 ±2,4	71,4 ±1,3	73,9 ±2,6	73,2 ±2,1	72,7 ±3,6
Post-operative	62,2 ±34,4	57,7 ±20,8	61,1 ±25,2	64,4 ±17,2	62,2 ±12,6
Solar flare	72,8 ±6,2	71,5 ±6,1	71,2 ±5,0	71,5 ±5,8	70,9 ±5,9
Sonar	83,4 ±1,9	88,9 ±4,6	88,9 ±6,3	89,6 ±3,0	88,9 ±3,5
Soybean	90,3 ±3,4	94,6 ±3,2	92,6 ±2,7	93,0 ±2,6	92,6 ±2,9
Splice junctions	92,4 ±1,1	94,0 ±1,0	95,6 ±0,7	93,8 ±0,9	93,7 ±1,0
Thyroid disease	96,2 ±0,9	93,9 ±0,6	95,5 ±0,8	95,5 ±0,9	95,4 ±0,9
Voting records	94,4 ±1,1	93,7 ±2,0	94,8 ±2,0	94,1 ±1,9	95,3 ±1,7
Wine	97,1 ±4,3	94,2 ±2,5	98,8 ±1,9	97,7 ±2,9	98,2 ±1,9
Zoology	93,9 ±7,5	96,9 ±0,0	95,9 ±4,3	95,9 ±4,3	96,9 ±7,5

Para entender melhor os resultados, as Tabelas 9 e 10 contêm algumas medidas de performance comparativas.

Tabela 9. Resumo dos resultados da comparação da acurácia do SUNRISE com a acurácia do RISE.

Medida	$k = 0$	$k = 1$	$k = 2$	$k = 3$
Número de vitórias	9	13	14	14
Número de derrotas	13	10	9	8
Número de vitórias estatisticamente significativas	4	1	2	2
Número de derrotas estatisticamente significativas	4	2	2	2

Tabela 10. Resumo dos resultados da comparação do tempo de execução do SUNRISE com o tempo de execução do RISE.

Medida	$k = 0$	$k = 1$	$k = 2$	$k = 3$
Número de vitórias	24	24	19	19
Número de derrotas	0	0	5	5

A primeira medida de performance é o número de vitórias, que é o número de conjuntos de dados onde o SUNRISE alcançou uma acurácia maior que o RISE. Complementarmente, o número de derrotas é o número de conjuntos de dados onde o SUNRISE não conseguiu alcançar uma acurácia maior ou igual à do RISE. Analisando essas duas medidas de performance, pode-se verificar que para $k = 0$, o RISE alcança acurácia maior em mais conjuntos de dados do que o SUNRISE. Isso já era de se esperar, pois, para $k = 0$, o SUNRISE comporta-se como um método IBL, que equivale ao subsistema IBL do RISE. Estudos anteriores já comprovaram que o RISE obtém

melhores resultados do que seus subsistemas funcionando independentemente. Para $k = 1$, a situação se inverte, e o SUNRISE tem mais vitórias que o RISE. Para $k = 2$ e $k = 3$, essa condição se mantém, havendo um pequeno aumento na diferença observada. Embora essas duas medidas de performance possam ajudar a deduzir as relações entre os resultados fornecidos por cada algoritmo, elas não representam com exatidão a diferença entre eles, pois algumas das diferenças entre as acurácias podem ser desprezíveis. Para que uma vitória (ou derrota) tenha realmente significado é necessário que a diferença entre as acurácias seja estatisticamente significativa.

Na terceira e na quarta linha da Tabela 9 são apresentados, respectivamente, o número de vitórias e o número de derrotas estatisticamente significativas do SUNRISE. Embora os resultados não sejam tão bons quanto os que aparentavam pela análise do número de vitórias, deve-se observar que o objetivo principal do SUNRISE é ser mais rápido que o RISE mantendo o nível de acurácia. Olhando por esse aspecto, os resultados são surpreendentemente bons, pois o número de derrotas estatisticamente significativas se manteve baixo em todos os testes, havendo ocasiões onde houve inesperadas vitórias. Portanto, essas medidas de performance mostram que parte do objetivo do SUNRISE foi alcançada, que é manter a boa acurácia preditiva do RISE.

A Tabela 10 contém as medidas de performance relacionadas a tempo de execução. Apesar de haver conjuntos de dados onde o RISE conseguiu ser mais rápido que o SUNRISE (para $k = 2$ e $k = 3$), o que é importante notar é que o SUNRISE conseguiu ser mais rápido em 86 dos 96 testes realizados (incluindo todos os valores de k utilizados). Além disso, em todos os testes o número de derrotas do SUNRISE no que diz respeito a tempo de execução foi de no máximo 5 para um dado k.

A fim de determinar se o SUNRISE é um algoritmo de aprendizado realmente rápido ou apenas mais rápido que o RISE, os tempos de execução do SUNRISE foram comparados com os tempos de execução dos algoritmos PEBLS, CN2 e C4.5 apresentados em DOMINGOS (1995). Os tempos de execução do RISE apresentados aqui e na referência em questão foram usados para calcular proporcionalmente os tempos de execução que os demais algoritmos teriam caso fossem testados nas mesmas condições do SUNRISE. Os resultados mostraram que o SUNRISE não é apenas mais rápido que o RISE, como também é rápido quando comparado a outros algoritmos de aprendizado. Entretanto, devido à diferença nos métodos de teste utilizados, nas implementações desenvolvidas e nos computadores nos quais elas foram executadas, tais resultados proporcionais não são totalmente confiáveis e por isso não serão apresentados aqui.

Uma vez que o SUNRISE apresenta tempo médio de execução menor que o RISE e alcança uma boa acurácia, os resultados mostram que o SUNRISE parece ser a escolha mais indicada para tratar de grandes conjuntos de dados.

3.5. O Algoritmo TextSUNRISE

A mineração de texto (*text mining*) (FELDMAN & DAGAN, 1995) consiste em aplicar técnicas de mineração de dados a texto não estruturado. A mineração de dados, também conhecida como descoberta de conhecimentos a partir de bancos de dados (KDD - *Knowledge Discovery from Databases*) é um tópico de recente significativo interesse. KDD considera a aplicação de métodos estatísticos e de aprendizado de máquinas para descobrir novos relacionamentos em grandes bancos de dados relacionais. Entretanto, para muitas aplicações, a informação disponível está na forma de documentos em linguagem natural não estruturados ao invés de bancos de dados estruturados. Deve-se portanto construir um sistema de extração de informações que extraia um banco de dados estruturado a partir de um conjunto de textos. Técnicas KDD padrão podem então ser aplicadas ao banco de dados resultante para descobrir relacionamentos interessantes. Especificamente, são induzidas regras para predizer a informação em cada campo do banco de dados dadas as informações em todos os outros campos. Métodos padrão de aprendizado de regras para classificação podem ser empregados para essa tarefa.

A idéia do SUNRISE foi estendida para a aplicação em mineração de texto, criando-se um algoritmo como o TextRISE (NAHM & MOONEY, 2002), a fim de obter um classificador mais eficiente para textos não estruturados.

3.5.1. Modificações no SUNRISE

Para ser aplicado à mineração de texto, o algoritmo SUNRISE foi alterado para ser capaz de extrair um conjunto de regras a partir de um conjunto de dados formado por exemplos onde os atributos e as classes são conjuntos de palavras. Essas alterações foram feitas de forma similar às que foram necessárias para o desenvolvimento do algoritmo TextRISE, cujo algoritmo é mostrado na Tabela 11. Os atributos serão chamados de *slots* antecedentes e a classe será chamada de *slot* conseqüente. O objetivo do novo algoritmo é predizer palavras pertencentes ao conjunto de palavras do *slot* conseqüente a partir das palavras pertencentes aos conjuntos dos *slots* antecedentes. O novo algoritmo criado recebeu o nome de TextSUNRISE e é apresentado na Tabela 12. Para avaliar a performance do TextSUNRISE, foi implementada também uma versão do TextRISE. A implementação do algoritmo TextRISE criada para este estudo é substancialmente mais simples do que a implementação descrita em NAHM & MOONEY (2002), portanto ela será chamada de T-RISE para evitar ambigüidade.

As principais diferenças do algoritmo SUNRISE para o algoritmo TextSUNRISE residem na métrica de similaridade e na medida de acurácia utilizadas. Como medida de performance, a acurácia na classificação foi substituída pela medida da similaridade média do texto predito com o texto real. A similaridade entre os conjuntos de palavras foi medida utilizando-se o modelo de espaço vetorial para texto, que é descrito a seguir.

O modelo de espaço vetorial (VSM - *Vector Space Model*) é largamente utilizado no campo de captação de informação (IR - *Information Retrieval*) (BAEZA-YATES & RIBEIRO-NETO, 1999). A maioria dos programas de busca na *Internet*

também usam medidas de similaridade baseadas nesse modelo para ordenar documentos. O modelo cria um espaço no qual ambos os documentos são representados por vetores. Para uma coleção fixa de documentos, um vetor *m*-dimensional é gerado para cada documento a partir de conjuntos de termos com pesos associados, onde *m* é o número de termos únicos na coleção de documentos. Então, uma função de similaridade vetorial, tal como o produto interno, pode ser usada para computar a similaridade entre os documentos.

A similaridade entre um documento x_j e um documento x_q, pode ser definida como o produto interno do vetor documento X_j com o vetor documento X_q:

$$sim(x_j, x_q) = X_j \cdot X_q = \frac{\sum_{i=1}^{m} w_{ij} \cdot w_{iq}}{\sqrt{\sum_{i=1}^{m} (w_{ij})^2 \cdot \sum_{i=1}^{m} (w_{iq})^2}}$$

onde *m* é o número de termos únicos na coleção de documentos, e os pesos associados aos termos dos documentos, w_{ij} e w_{iq}, serão definidos a seguir.

O esquema de pesos TFIDF (*Term Frequency times Inverse Document Frequency* - freqüência do termo vezes freqüência inversa nos documentos) (SALTON, 1989) é usado para atribuir maior peso a termos mais importantes do documento. TFIDF mantém duas suposições sobre a importância de um termo. Primeiro, quanto mais um termo aparece em um documento, mais importante ele é (freqüência do termo ou fator *tf*). A outra suposição é que quanto mais um termo aparece na coleção inteira de documentos, menos importante ele é, uma vez que ele não caracteriza bem nenhum documento em particular (freqüência inversa nos documentos ou fator *idf*).

Logo, o peso w_{ij} do termo t_i em um documento d_j é dado por:

$$w_{ij} = tf_{ij} \cdot idf_i$$

onde tf_{ij} é a freqüência do termo t_i no documento d_j e idf_i corresponde à freqüência inversa nos documentos, dada por:

$$idf_i = \log_2\left(\frac{N}{n_i}\right)$$

onde N é o número total de documentos na coleção e n_i é o número de documentos onde o termo t_i ocorre pelo menos uma vez.

O procedimento do SUNRISE de generalização de uma regra para cobrir um exemplo também teve que ser alterado para se adequar ao novo formato dos exemplos e regras. A generalização mais específica é realizada computando-se a interseção entre os conjuntos de palavras de cada *slot* formador dos exemplos ou regras. O algoritmo de generalização é mostrado na Tabela 13. Uma regra cobre um exemplo se todos os seus antecedentes são subconjuntos dos antecedentes correspondentes do exemplo.

O programa desenvolvido não faz categorização automática ou extração de informação. Os textos foram rotulados com um número limitado de categorias fixas, usando-se um programa auxiliar simples, que também se encarrega de eliminar muitas *stop-words* (por exemplo, "um", "você", "é" etc.).

Tabela 11. O algoritmo TextRISE.

Entrada: *ES* é o conjunto de treinamento.

Saída: *RS* é o conjunto de regras.

Procedimento TextRISE (*ES*)

Seja *RS* = *ES*.

Compute Acurácia-Texto (*RS*).

Repita

Para cada regra *R* em *RS*,

Encontre o exemplo *E* com maior similaridade a regra *R*, ainda não coberto por ela e da mesma classe.

Seja *R'* = Generalização-Mais-Específica-Texto (*R*, *E*).

Seja *RS'* = *RS* com *R* substituído por *R'*.

Se Acurácia-Texto (*RS'*) ≥ Acurácia-Texto (*RS*) então

Substitua *RS* por *RS'*,

Se *R'* for idêntica a uma outra regra em *RS* então

Apague *R'* de *RS*.

Até que nenhum aumento em Acurácia-Texto (*RS*) seja obtido.

Retorne *RS*.

Tabela 12. O algoritmo TextSUNRISE.

Entrada: *ES* é o conjunto de treinamento, *k* é o parâmetro do TextSUNRISE.

Saída: *RS* é o conjunto de regras.

Procedimento TextSUNRISE (*ES*, *k*)

Seja *RS* = *ES*.

Compute Acurácia-Texto (*RS*).

Repita

Para cada regra *R* em *RS*,

Repita *k* vezes

Encontre o exemplo *E* com maior similaridade a regra *R*, ainda não coberto por ela e da mesma classe.

Seja *R'* = Generalização-Mais-Específica-Texto (*R*, *E*).

Seja *RS'* = *RS* com *R* substituído por *R'*.

Se Acurácia-Texto (*RS'*) > Acurácia-Texto (*RS*) então

R = *R'*

Senão Quebre o laço

Se Acurácia-Texto (*RS'*) > Acurácia-Texto (*RS*) então

Substitua *RS* por *RS'*,

Se *R'* for idêntica a uma outra regra em *RS* então

Apague *R'* de *RS*.

Até que nenhum aumento em Acurácia-Texto (*RS*) seja obtido.

Retorne *RS*.

Tabela 13. Procedimento do TextSUNRISE para generalização de uma regra para cobrir um exemplo.

Entradas: $R = (A_1, A_2, ..., A_a, C_R)$ é uma regra, $E = (E_1, E_2, ..., E_a, C_E)$ é um exemplo.

A_i, E_i, C_R e C_E são conjuntos de palavras, possivelmente vazios.

Saída: $R' = (A'_1, A'_2, ..., A'_a, C'_R)$ é a regra generalizada.

Procedimento Generalização-Mais-Específica-Texto (R, E)

Para $i = 1$ até a,

$$A'_i = A_i \cap E_i$$

$C'_R = C_R \cap C_E$

Retorne R'.

3.5.2. Conjunto de Dados

Para comparar o desempenho dos algoritmos TextSUNRISE e T-RISE, 691 páginas com descrições de filmes foram coletadas na *Internet* (http://www.warner.com.br/) para construir um banco de dados textual.

Para construir o conjunto de dados, primeiro foi extraído um modelo estruturado a partir das páginas com as descrições dos filmes. A partir do modelo foram construídas instâncias para cada descrição de filme, com cinco *slots* predefinidos (*Título*, *Direção*, *Elenco*, *Gênero* e *Sinopse*) que foram preenchidos com palavras extraídas do texto. Apenas *slots* com palavras foram considerados. Entretanto, *slots* com

valores numéricos, tais como *ano de produção*, poderiam ser fornecidos como atributos adicionais de entrada para a previsão de outros *slots*. Então o TextSUNRISE e o T-RISE foram aplicados para extrair regras de predição a partir desse banco de dados estruturado.

3.5.3. Metodologia Experimental

Todos os experimentos foram conduzidos usando-se validação cruzada com 10 partições. Foram induzidas regras para predizer os componentes dos *slots Título*, *Direção*, *Elenco*, *Gênero* e *Sinopse*, uma vez que estes são usualmente preenchidos com múltiplos componentes de valores discretos com relacionamento em potencial entre seus valores.

Para determinar a acurácia de um conjunto de regras, foi medida a precisão ao se predizer a presença de alguma palavra no conjunto conseqüente a partir das palavras contidas nos conjuntos antecedentes. A predição é julgada correta se e somente se ela contiver pelo menos uma palavra que também pertence ao conjunto correspondente ao *slot* predito.

Os resultados foram avaliados estatisticamente por um teste *t* emparelhado. Para cada *slot* predito, os sistemas foram comparados para determinar se a diferença entre suas acurácias foi estatisticamente significativa.

3.5.4. Resultados Experimentais

Os resultados experimentais obtidos com a implementação dos algoritmos para o banco de dados textual construído são mostrados nas Tabelas 14 e 15.

A Tabela 14 apresenta os tempos médios de execução dos algoritmos. Os resultados mostram que, no que diz respeito a tempo de execução, o algoritmo TextSUNRISE não apresenta um ganho em relação ao T-RISE tão grande quanto o SUNRISE apresenta em relação ao RISE. Isso acontece porque a função de similaridade do TextSUNRISE tem um custo muito maior que a do SUNRISE, pois cada *slot* pode conter dezenas, ou até mesmo centenas de palavras. Em termos de performance computacional o maior problema é a complexidade temporal quadrática no pior caso de se medir a similaridade de muitos pares de itens. A isso soma-se o tempo gasto na tentativa de se induzir regras. Ainda assim, vale notar que para $k = 0$, ou seja, usando o TextSUNRISE como um algoritmo de IBL, o tempo de execução é sempre menor que o do T-RISE, pois nenhuma regra é induzida, para $k = 1$, o TextSUNRISE perde para o T-RISE apenas em 1 teste, e para $k = 2$ ou $k = 3$, o TextSUNRISE ganha do T-RISE em 2 dos 5 testes.

Tabela 14. Tempos de execução do T-RISE e do TextSUNRISE (em segundos).

Conseqüente	T-RISE	TextSUNRISE			
		$k = 0$	$k = 1$	$k = 2$	$k = 3$
Título	5268,186	4085,824	4414,945	4635,604	4929,890
Direção	6162,637	4017,523	4833,516	5046,153	5160,439
Elenco	4123,131	3772,362	4045,494	4254,945	4441,098
Gênero	4503,406	3400,329	5288,571	5341,703	5382,362
Sinopse	2418,186	2125,659	2284,560	2462,967	2543,626

A Tabela 15 apresenta as acurácias médias de predição para ambos os algoritmos com confiança de 95%. Comparando as acurácias médias, é possível verificar que o TextSUNRISE possui acurácia média maior que o T-RISE em quase todos os testes, sendo que em 2 deles (*Elenco* e *Gênero*) a diferença é estatisticamente significativa ao nível 0,05. Pode-se verificar também que a acurácia do TextSUNRISE é praticamente a mesma para todos os valores de k testados, inclusive $k = 0$. Isso pode acontecer por dois motivos: ou as regras geradas apresentam a mesma acurácia do conjunto de instâncias inicial, ou, o que é mais provável, poucas regras são generalizadas e sua similaridade aos exemplos de teste é baixa.

Tabela 15. Acurácias médias do T-RISE e do TextSUNRISE (intervalo de confiança de 95%).

Conseqüente	T-RISE	TextSUNRISE			
		$k = 0$	$k = 1$	$k = 2$	$k = 3$
Título	9,7 ±1,2	10,5 ±1,7	10,5 ±1,7	10,4 ±1,8	10,5 ±1,7
Direção	1,4 ±0,3	5,8 ±1,2	5,8 ±1,2	5,8 ±1,2	5,9 ±1,1
Elenco	19,3 ±2,7	21,0 ±2,7	21,0 ±2,7	21,0 ±2,7	21,0 ±2,7
Gênero	46,8 ±4,5	66,2 ±3,5	66,2 ±3,3	66,3 ±3,5	66,3 ±3,5
Sinopse	72,8 ±3,6	72,7 ±3,5	72,6 ±3,1	72,1 ±3,3	72,1 ±3,6

A conclusão que se pode tirar desses resultados é que, caso o objetivo seja alcançar a maior acurácia preditiva no menor tempo, se for possível manter o conjunto de dados textual completo na forma de instâncias, o TextSUNRISE usado como sistema IBL é mais adequado à mineração de texto. Caso contrário, o T-RISE pode ser aplicado para minerar regras mais simples, de maior inteligibilidade e que ocuparão bem menos espaço, mas que por outro lado fornecerão uma acurácia menor. Embora os resultados com esse conjunto de dados em especial tenham dado margem a essa interpretação, uma avaliação mais completa se faz necessária em outros domínios.

4. CONCLUSÕES E TRABALHOS FUTUROS

Esta pesquisa apresentou um algoritmo de aprendizado multi-estratégia desenvolvido para tentar superar a limitação de velocidade de seu predecessor. A avaliação experimental extensiva demonstrou o sucesso dessa tentativa.

Além de ser mais rápido que seu predecessor, tornando possível o aprendizado em conjuntos de dados intratáveis com a utilização do RISE, o algoritmo SUNRISE mostrou ser capaz de alcançar uma acurácia média tão boa quanto à do RISE e em alguns casos até superior a deste.

Embora nos testes o TextSUNRISE não tenha se mostrado muito superior ao T-RISE em termos de velocidade, ele apresentou acurácia superior na grande maioria dos testes. Portanto, tais resultados mostram que o TextSUNRISE pode ser uma boa opção em aplicações onde o que mais importa é a acurácia preditiva e não a simplicidade e inteligibilidade das regras geradas.

Uma possível pesquisa futura seria a incorporação da abordagem *wrapper* ao SUNRISE, para diminuir sua sensibilidade a atributos irrelevantes. Pode-se também incluir um sistema de poda que elimine do conjunto final de regras aquelas que tenham baixa acurácia ou que classifiquem poucos exemplos, para obter um conjunto de regras menor e mais acurado. É importante que esse processo seja realizado cuidadosamente para que nem a acurácia e nem a velocidade do algoritmo sejam prejudicados. Essas melhorias no SUNRISE podem ser também estendidas ao TextSUNRISE. Este por sua vez precisa ser devidamente testado em outros conjuntos de dados, para se ter uma idéia melhor de suas capacidades e limitações.

REFERÊNCIAS BIBLIOGRÁFICAS

- AHA, D. W., 1990, *A study of instance-based learning algorithms for supervised learning tasks: Mathematical, empirical, and psychological evaluations*. In: Technical Report 90-42, University of California, Department of Information and Computer Science, Irvine, CA.

- AHA, D. W., KIBLER, D., ALBERT, M. K., 1991, "Instance-based learning algorithms", *Machine Learning*, 6, pp. 37-66.

- BAEZA-YATES, R., RIBEIRO-NETO, B., 1999, *Modern Information Retrieval*. New York, ACM Press.

- CLARK, P., BOSWELL, R., 1991, "Rule induction with CN2: Some recent improvements". In: *Proceedings of the Sixth European Working Session on Learning*, Porto, Portugal: Springer-Verlag, pp. 151-163.

- CLARK, P., NIBLETT, T., 1989, "The CN2 induction algorithm", *Machine Learning*, 3, pp. 261-283.

- COST, S., SALZBERG, S., 1993, "A weighted nearest neighbor algorithm for learning with symbolic features", *Machine Learning*, 10, pp. 57-78.

- DeGROOT, M. H., 1986, *Probability and statistics*. Reading, MA: Addison-Wesley.

- DIETTERICH, T. G., 1998, "Approximate statistical tests for comparing supervised classification learning algorithms", *Neural Computation*, 10, pp. 1895-1924.

- DOMINGOS, P., 1994, "The RISE system: Conquering without separating". In: *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, New Orleans: IEEE Computer Society Press, pp. 704-707.

- DOMINGOS, P., 1995, *The RISE 2.0 system: A case study in multistrategy learning*. In: Technical Report 95-2, University of California, Department of Information and Computer Science, Irvine, CA.

- DOMINGOS, P., 1996, "Unifying Instance-Based and Rule-Based Induction", *Machine Learning*, 24, pp. 141-168.

- FELDMAN, R., DAGAN, I., 1995, "Knowledge discovery in textual databases (KDT)". In: *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal.

- HOLTE, R. C., ACKER, L. E., PORTER, B. W., 1989, "Concept learning and the problem of small disjuncts". In: *Proceedings of the Eleventh International Joint Conference on Articial Intelligence*, Detroit, MI: Morgan Kaufmann, pp. 813-818.

- KOHAVI, R., 1995, "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1137-1143.

- KOHAVI, R., JOHN, G. H., 1997, "Wrappers for feature subset selection", *Artificial Intelligence*, 97, pp. 273-324.

- MICHALSKI, R. S., 1983, "A theory and methodology of inductive learning", *Artificial Intelligence*, 20, pp. 111-161.

- MICHALSKI, R. S., MOZETIC, I., HONG, J., LAVRAC, N., 1986, "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains". In: *Proceedings of the Fifth National Conference onArticial Intelligence*, Philadelphia, PA: AAAI Press, pp. 1041-1045.

- MICHALSKI, R. S., TECUCI, G., 1994, *Machine learning: A multistrategy approach*. San Mateo, CA, Morgan Kaufmann.

- MITCHELL, T. M., 1997, *Machine Learning*. New York, McGraw-Hill.

- MURPHY, P. M., AHA, D. W., 1995, *UCI repository of machine learning databases*. In: Machine-readable data repository, University of California, Department of information and Computer Science, Irvine.

- NAHM, U. Y., MOONEY, R. J., 2002, "Text Mining with Information Extraction". In: *Proceedings of the Spring Symposium on Mining Answers from Texts and Knowledge Bases*, Stanford, CA, pp. 60-67.

- NIBLETT, T., 1987, "Constructing decision trees in noisy domains". In: *Proceedings of the Second European Working Session on Learning*, Bled, Yugoslavia: Sigma, pp. 67-78.

- QUINLAN, J. R., 1986, "Induction of decision trees", *Machine Learning*, 1, pp. 81-106.

- QUINLAN, J. R., 1993, *C4.5: Programs for Machine Learning*. San Mateo, CA, Morgan Kaufmann.

- RIVEST, R. L., 1987, "Learning decision lists", *Machine Learning*, 2, pp. 229-246.

- SALTON, G., 1989, *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley.

- SALZBERG, S., 1991, "A nearest hyperrectangle learning method", *Machine Learning*, 6, pp. 251-276.

- STANFILL, C., WALTZ, D., 1986, "Toward memory-based reasoning", *Communications of the ACM*, 29, pp. 1213-1228.