Pontifícia Universidade Católica
do Rio de Janeiro

David Steinbruch

Um estudo de algoritmos para classificação automática de textos utilizando naive-Bayes

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Daniel Schwabe
Co-orientador: Ruy Luiz Milidiú

Rio de Janeiro, setembro de 2006

Pontifícia Universidade Católica
do Rio de Janeiro

David Steinbruch

Um estudo de algoritmos para classificação automática de textos utilizando naive-Bayes

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Daniel Schwabe
Orientador
PUC-Rio

Ruy Luiz Milidiú
Co-orientador
PUC-Rio

Marcus Vinicius Soledade Poggi de Aragão
PUC-Rio

Eduardo Sany Laber
PUC-Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro

05 de setembro de 2006

David Steinbruch

Graduou-se em Engenharia de Computação na PUC-Rio em 2003. Atuou como programador no desenvolvimento de soluções web e aplicações de mineração de textos. Possui interesse acadêmico e profissional nas áreas de Inteligência Artificial, Hipertexto e Multimídia.

Ficha Catalográfica

Steinbruch, David

Um estudo de algoritmos para classificação automática de textos utilizando naive-Bayes / David Steinbruch ; orientador: Daniel Schwabe. – 2006.

78 f. : il. ; 30 cm

Dissertação (Mestrado em Informática)– Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006.

Inclui bibliografia

1. Informática – Teses. 2. Aprendizado de máquina. 3. Categorização de textos. 4. Classificação de textos. 5. Multirótulo. 6. Naive-Bayes. 7. Internet. I. Schwabe, Daniel. II. Milidiú, Ruy Luiz. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Este trabalho é dedicado a minha família,
pela compreensão e pelo apoio,
e ao meu orientador e coorientador,
pela motivação.

Agradecimentos

À PUC-Rio, ao departamento de informática e ao CNPq pela oportunidade.

Ao meu orientador, Daniel Schwabe, e co-orientador, Ruy Luiz Milidiú, pela motivação, paciência, confiança e ajuda.

A todos os professores, funcionários do Departamento de Informática pelo apoio dado quando precisei.

A Deborah e Emanuelle pela paciência e ajuda nos processos burocráticos.

Aos meus amigos que cursaram o mestrado comigo e familiares pelo incentivo e apoio nos momentos difíceis.

Aos meus irmãos Daniel, Rachel e Natan pelos ótimos momentos juntos.

Aos meus pais Luna e Beni e meu padrasto Samy por tudo.

Resumo

Steinbruch, David; Schwabe, Daniel. **Um estudo de algoritmos para classificação automática de textos utilizando naive-Bayes**. Rio de Janeiro, 2006. 78p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A quantidade de informação eletrônica vem crescendo de forma acelerada, motivada principalmente pela facilidade de publicação e divulgação que a Internet proporciona. Desta forma, é necessária a organização da informação de forma a facilitar a sua aquisição. Muitos trabalhos propuseram resolver este problema através da classificação automática de textos associando a eles vários rótulos (classificação multirótulo). No entanto, estes trabalhos transformam este problema em subproblemas de classificação binária, considerando que existe independência entre as categorias. Além disso, utilizam limiares ("thresholds"), que são muito específicos para o conjunto de treinamento utilizado, não possuindo grande capacidade de generalização na aprendizagem. Esta dissertação propõe dois algoritmos de classificação automática de textos baseados no algoritmo multinomial naive Bayes e sua utilização em um ambiente on-line de classificação automática de textos com realimentação de relevância pelo usuário. Para testar a eficiência dos algoritmos propostos, foram realizados experimentos na base de notícias Reuters 21758 e na base de documentos médicos Ohsumed.

Palavras-chave

Aprendizado de Máquina; Categorização de Textos, Classificação de Textos; Multirótulo; Naive-Bayes; Internet

Abstract

Steinbruch, David; Schwabe, Daniel**. A study of multilabel text clasification algorithms using naive-Bayes.** Rio de Janeiro, 2006. 78p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The amount of electronic information has been growing fast, mainly due to the easiness of publication and spreading that Internet provides. Therefore, is necessary the organisation of information to facilitate its retrieval. Many works have solved this problem through the automatic text classification, associating to them several labels (multilabel classification). However, those works have transformed this problem into binary classification subproblems**,** considering there is not dependence among categories. Moreover, they have used thresholds, which are very sepecific of the classifier document base, and so, does not have great generalization capacity in the learning process. This thesis proposes two text classifiers based on the multinomial algorithm naive Bayes and its usage in an on-line text classification environment with user relevance feedback. In order to test the proposed algorithms efficiency, experiments have been performed on the Reuters 21578 news base, and on the Ohsumed medical document base.

Keywords

Machine Learning; Text Categorization; Text Classification; Multilabel; Naive-Bayes; Internet

Sumário

Siglas

Micro Recall	Micro Averaged Recall
Macro Recall	Macro Averaged Recall
Micro Precision	Micro Averaged Precision
Macro Precision	Macro Averaged Precision
Micro F1	Micro Averaged F1
Macro F1	Macro Averaged F1
MESH	Medical Subject Heading
R(10)	Subconjunto de 10 categorias da base de dados Reuters 21758
R(90)	Subconjunto de 90 categorias da base de dados Reuters 21758
CONSTRUE	Categorization of News STories, Rapidly, Uniformly and Extensibly
TF	Term frequency
IDF	Inverse Document Frequencie
Reuters 21578	Base de notícias da Reuters com 21578 documentos
Ohsumed	Subconjunto de 348.566 documentos da base MEDLINE
MEDLINE	Medical Literature Analysis and Retrieval System Online
ModApté	Subonjunto de documentos da base Reuters 21578, composto de 9.603 documentos de treinamento e 3.299 documentos de teste
DF	Document Frequency
IG	Information Gain
MI	Mutual Information
CHI	CHI-squared statistic

Lista de tabelas

Lista de figuras

1
Introdução

1.1
Caracterização do problema

A apresentação de informação sob a forma eletrônica vem crescendo de modo acelerado nas últimas décadas, principalmente por conta da internet. O grande alcance desse meio de comunicação e a facilidade de disponibilização de conteúdo são os principais fatores que incentivam a disposição de textos na internet.

Dessa forma é necessário criar maneiras de acesso a esses textos de forma rápida e objetiva.

Quando se trata de notícias de jornal, a necessidade de obtenção de informação de forma rápida e objetiva é mais crítica, pois o usuário deseja estar sempre atualizado e informado sobre as notícias publicadas nas suas fontes de preferência.

Com os mecanismos atuais de aquisição de informação na internet, como por exemplo, a ferramenta de busca mais popular Google, nem sempre os resultados são de acordo com a necessidade do usuário, retornando muita informação irrelevante ao contexto da busca. Assim sendo, só resta ao usuário realizar uma busca manual e exaustiva no conjunto de respostas retornado pela busca. Um dos principais fatores para isso é que freqüentemente o usuário não sabe expressar sua busca por palavras chaves, muitas vezes expressando sua consulta com poucas palavras e com palavras que possam gerar ambigüidade [Baeza-Yates et al., 2004]. Além disso, não existe nenhuma noção de semântica nesse tipo de busca, retornando ao usuário textos irrelevantes que simplesmente possuem as palavras chaves expressas na sua consulta [Baeza-Yates & Ribeiro-Neto, 1999].

Uma solução para o problema de organização e aquisição de informação é a classificação automática de textos. Com a classificação automática de textos o usuário pode especificar categorias de interesse, por exemplo, *economia do Brasil,*

e política externa, e o sistema buscar na internet, ou em fontes especificadas pelo usuário, retornando ao usuário documentos referentes aos tópicos desejados. Para isso, é necessária a criação de um conhecimento de domínio sobre os tópicos especificados. Até os anos 80, a técnica mais popular para extração desse conhecimento era através da construção manual de regras realizada por engenheiros do conhecimento, como por exemplo, o sistema CONSTRUE [Hayes & Weinstein, 1990], criado pelo grupo Carnegie para a agência de notícias Reuters. A grande desvantagem desses sistemas é que exigia um esforço humano considerável para a criação manual das regras e caso o conhecimento de domínio fosse atualizado, seria necessária novamente uma intervenção dos engenheiros do conhecimento.

A partir de então, foram propostas abordagens que se baseiam em técnicas de aprendizado de máquina. Nessas abordagens, através de um processo indutivo de aprendizado, é criado automaticamente um classificador, dado um conjunto de relações entre documentos e rótulos associados. Uma vez criado o classificador, dado um documento não conhecido, o algoritmo classifica o documento nas categorias aprendidas na fase de treinamento.

No entanto, muitas das abordagens propostas resolveram o problema de classificação binária de textos não considerando o caso mais geral que é a possibilidade de um documento estar associado a mais de um rótulo. Dentre as propostas que trataram o problema de classificação multirótulo, a maioria transforma este problema em subproblemas de classificação binária, considerando que existe independência entre as categorias. Além disso, utilizam limiares, que são muito específicos para o conjunto de treinamento utilizado, não possuindo grande capacidade de generalização na aprendizagem.

1.2 Objetivo

Esta dissertação propõe dois algoritmos de classificação automática de textos baseados no algoritmo multinomial naive Bayes e sua aplicação em um ambiente on-line de classificação automática de notícias com realimentação de relevância pelo usuário, combinando técnicas de aprendizado de máquina e mineração de textos.

O sistema proposto é constituído de três etapas: na primeira o usuário especifica um conjunto de fontes, um conjunto de categorias de interesse, e o sistema busca documentos nas fontes de informação. Em seguida o usuário associa os documentos às categorias podendo classificar um documento em mais de uma categoria e também criar novas categorias. Por último, o sistema busca novos documentos e os classifica de acordo com o treinamento realizado anteriormente. O processo de classificação e treinamento se repete tantas vezes quanto o usuário queira.

Muitas são as vantagens de se utilizar tal técnica: A semântica das categorias é subjetiva, possibilitando ao usuário uma melhor especificação de sua necessidade de informação. Outra vantagem é que a possibilidade de realimentação do usuário e a adaptação do sistema às correções de falsos positivos e falsos negativos implicam em uma grande flexibilidade do sistema e convergência para uma alta acurácia. Além disso, a categorização de um documento em mais de uma classe é bem mais realista, uma vez que é comum a sobreposição de categorias.

Para testar a eficiência dos algoritmos propostos, foram realizados testes na base de dados da Reuters e na base de dados Ohsumed.

1.3 Trabalhos relacionados

A classificação de textos é uma área muito rica no que se refere à quantidade de trabalhos publicados e pesquisas. A tarefa de classificação de textos data da década de 60, porém com a criação da internet se tornou cada vez mais necessária, uma vez que a quantidade de informação disponível vem crescendo de forma acelerada.

Desta forma, muitos classificadores foram propostos, entre eles o classificador naive Bayes, utilizado nesse trabalho. Tal classificador assume que existe independência entre as palavras de um texto (por isso é chamado de "naive"), o que na maioria das tarefas é uma suposição falsa. Porém, Domingos & Pazzani, [1997] mostraram teoricamente que a suposição de independência de palavras na maioria dos casos não prejudica a eficiência do classificador.

Basicamente, existem dois tipos de modelos estatísticos para os classificadores naive Bayes. O modelo multinomial, usado nessa dissertação, que representa um documento como um vetor de freqüências das palavras no texto e o modelo binário, que representa um documento como um vetor binário de palavras, considerando apenas a ocorrência das palavras no texto. O modelo binário foi apresentado em [Robertson & Sparck Jones, 1976; Koller & Sahami, 1997]. Já o modelo multinomial foi apresentado em [Joachims, 1998; McCallum & Nigam, 1998]. Lewis, [1998] apresenta uma comparação teórica entre o modelo binário e o modelo multinomial. Já McCallum & Nigam, [1998], através de experimentos, comparam o modelo multinomial com o modelo binário e concluem que o modelo multinomial apresenta melhores resultados.

Outras técnicas foram propostas além do classificador naive Bayes como, por exemplo, support vector machines [Vapnik 1998], que atualmente tem-se mostrado muito eficiente [Joachims, 1998], K-nearest neighbors [Yang & Liu, 1999], árvores de decisão [Breiman et al., 1984], Boosting [Schapire & Singer, 2000] e redes neurais [Shavlik & Eliassi-Rad, 1998].

Grande parte dos trabalhos resolve o problema da classificação binária e para resolver o problema multirótulo, propõe a transformação do problema em subproblemas binários. A técnica mais conhecida, chamada de "one-vs-all" cria para cada classe um classificador binário, capaz de distingui-la das demais classes do domínio. Assim, um exemplo associado a uma categoria c_i é um exemplo positivo para c_i, e um exemplo negativo para todas as outras classes as quais o documento não pertence.

Entretanto, McCallum [1999] propõe um algoritmo bayesiano que soluciona o problema de classificação multirótulo através de um modelo misto e "Expectation Maximization".

1.4 Organização da dissertação

Os capítulos a seguir estão organizados da seguinte forma. O capítulo 2 apresenta os fundamentos teóricos de máquina de aprendizado, a tarefa de classificação automática de textos, duas principais abordagens do algoritmo naive Bayes e as principais medidas para apresentar a eficiência de classificadores de

texto. No capítulo 3 são apresentados os dois algoritmos de classificação propostos nessa dissertação, assim como sua aplicação em um ambiente online de filtragem de documentos com realimentação de relevância do usuário. O capítulo 4 apresenta os resultados obtidos em testes realizados na base de notícias da Reutes e na base de documentos médicos Ohsumed e analisa os resultados obtidos. O capitulo 5 apresenta os principais trabalhos relacionados. Finalmente, o capitulo 6 apresenta as conclusões e sugestões para trabalhos futuros.

2
Fundamentos Teóricos

2.1
Aprendizado de Máquina

Aprendizado de Máquina é uma área de Inteligência Artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática. Possui um grande número de aplicações como, por exemplo, máquinas de busca, diagnóstico médico, detecção automática de fraude de cartão de crédito, análise de aplicações financeiras, classificação de seqüências de DNA e classificação automática de textos.

Um programa aprende um conjunto de tarefas *T* com uma medida de desempenho *D* a partir de uma experiência *E*, se seu desempenho de aprendizado *D* aumenta com a experiência *E* [Mitchell, 1997], ou seja, se é capaz de tomar decisões baseado em experiências acumuladas por meio da solução bem sucedida de problemas anteriores. Por exemplo, uma aplicação que necessite reconhecer pessoas através de um sensor tem como tarefa *T* associar imagens captadas pelo sensor a pessoas, medida de desempenho *D* como sendo a percentagem de pessoas corretamente reconhecidas e experiência de treinamento *E* como sendo um conjunto de pares ordenados compostos por uma imagem e o nome de uma pessoa.

Existe uma série de problemas que são de difícil solução que podem ser bem resolvidos com as técnicas de aprendizado de máquina. Por exemplo: algumas tarefas só podem ser bem definidas através de exemplos, onde não é conhecida nenhuma relação à priori entre os dados de entrada e a saída desejada. Além disso, tarefas que são sujeitas a muitas mudanças podem ser aprendidas pela máquina, uma vez que a máquina se adapta a tais mudanças. A capacidade de adaptação de sistemas de aprendizado de máquina também permite que tais sistemas sejam portados para domínios completamente diferentes sem a necessidade de recriar o sistema para suportar o novo domínio. Assim, se uma máquina é capaz de

aprender a classificar notícias de jornais em uma hierarquia de categorias, a mesma máquina também é capaz de aprender a filtrar spams ou até mesmo resolver problemas de ambigüidade em palavras no processamento de linguagem natural.

Para desenvolver um sistema de aprendizado é necessário tomar algumas decisões de projeto. Primeiramente, deve-se escolher que tipo de exemplos de treinamento será usado. Por exemplo, no problema de classificação automática de textos, o treinamento será um conjunto de pares compostos de um documento e sua respectiva classe.

Após a escolha do tipo de dados a ser usado como exemplo de treinamento, o próximo passo é definir exatamente que tipo de conhecimento será aprendido. O problema de aquisição de conhecimento pode, então, ser reduzido ao problema de se aprender uma função objetivo *f*, que associa valores de entrada (os específicos das tarefas a serem aprendidas) com valores de saída correspondentes, que auxiliam nas decisões ou ações a serem tomadas. O sistema gera uma hipótese *h* que se aproxima da função objetivo *f* conforme realizado o treinamento.

Em aprendizado de máquina existem diversas formas de aprendizagem, como por exemplo, Simbólico, Estatístico, Baseado em Exemplos, Conexionista e Evolutivo.

Os sistemas de aprendizado simbólico constroem representações simbólicas de um conceito através da análise de exemplos e contra-exemplos, na forma de alguma expressão lógica, árvore de decisão, regras ou rede semântica.

O aprendizado estatístico utiliza modelos estatísticos para encontrar uma aproximação da função objetivo. O problema a ser resolvido é, dado um conjunto de exemplos de distribuição de probabilidade não conhecida, descobrir a qual distribuição um novo exemplo pertence.

O aprendizado baseado em exemplos utiliza a idéia de que para determinar a saída da função objetivo, dado um valor de entrada não conhecido, deve-se buscar outro valor de entrada similar cuja saída é conhecida e assumir que o novo exemplo terá o mesmo valor de saída. Para isso, é necessário guardar os exemplos de treinamento em memória, sendo, por isso, chamado de sistemas de aprendizado "lazy", em oposição aos sistemas de aprendizado "eager", que utilizam os exemplos para induzir o modelo, descartando-os logo em seguida.

O aprendizado conexionista utiliza redes neurais, que são construções matemáticas não lineares altamente interconectadas inspiradas no fenômeno de aprendizado do cérebro humano. As conexões entre os neurônios artificiais possuem pesos que são obtidos através do treinamento, ou seja, o conhecimento adquirido durante o treinamento de uma rede neural fica armazenado nos pesos das ligações entre os neurônios artificiais. Muitas são as aplicações práticas das redes neurais, como por exemplo, reconhecimento de imagens, reconhecimento de genes em uma seqüência de DNA e previsão do movimento da bolsa de valores, a partir de uma série histórica.

O aprendizado evolutivo se baseia na lei da seleção natural de Darwin. O algoritmo é iniciado com uma população de estimadores para a função objetivo. Tais estimadores competem para realizar a reprodução e com isso produzir indivíduos similares. Indivíduos que possuem desempenho fraco tendem a ser descartados da população de estimadores, enquanto indivíduos de desempenho alto tendem a proliferar. Dessa forma, conforme o algoritmo é executado, o melhor estimador da função objetivo vai sendo aprimorado.

2.2
O problema da classificação

Em geral, existem dois tipos de raciocínio. O raciocínio dedutivo e o raciocínio indutivo. O raciocínio dedutivo é uma forma de inferência na qual a conclusão tem o mesmo grau de certeza que as premissas, em oposição ao raciocínio indutivo, onde a conclusão pode ter um grau de certeza inferior às premissas.

Um exemplo de raciocínio dedutivo:

Todos os pássaros possuem asas.

Um canário é um pássaro.

Logo, um canário possui asas.

Um exemplo de raciocínio indutivo:

Todos os corvos observados são pretos.

Logo, todos os corvos são pretos.

Através do raciocínio indutivo, é possível obter conclusões genéricas sobre um conjunto particular de exemplos. Em aprendizado de máquina, o raciocínio indutivo é utilizado generalizar conceitos, para aproximar funções, para descobrir novas funções através de exemplos fornecidos.

O aprendizado indutivo deve ser aplicado com cuidado, pois se o número de exemplos for insuficiente, ou se os exemplos não forem bem escolhidos, as hipóteses obtidas podem ser inconsistentes.

Classificação e regressão são dois tipos de aplicações de aprendizado indutivo com a finalidade de predizer valores de uma função objetivo *f*, dado um valor de entrada ainda não conhecido.

A regressão consiste em aproximar uma função contínua a partir de um conjunto de exemplos composto de pontos. Tal método pode ser aplicado na previsão de preços de um produto, valores de ações no mercado de ações e prever a ocorrência de um determinado evento (regressão logística).

Na tarefa de classificação, o objetivo do algoritmo de aprendizado é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados, dado um conjunto de classes e um conjunto de exemplos de treinamento.

O aprendizado indutivo pode ser subdivido em aprendizado não supervisionado e aprendizado supervisionado.

No aprendizado supervisionado, o conjunto de treinamento consiste de pares ordenados constituídos de um objeto (tipicamente vetores) e o seu respectivo valor da função objetivo. A saída da função objetivo pode ser contínua, no caso da regressão, ou pode ser discreta, no caso da classificação, onde os valores de saídas são rótulos de categorias.

No aprendizado não supervisionado, o conjunto de treinamento consiste apenas de exemplos sem nenhum valor de função associado. Tipicamente, o problema se resume em particionar os exemplos de treinamento em agrupamentos, ou clusters. Ainda assim, pode-se considerar o problema como um caso de aprendizagem de uma função objetivo, pois o valor da função é o nome do agrupamento ao qual um objeto de entrada pertence.

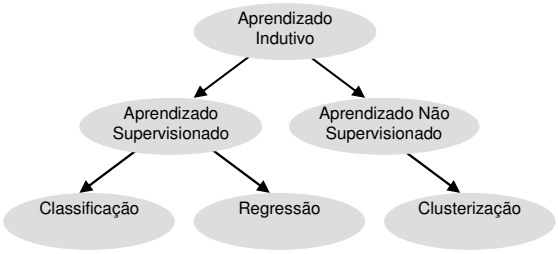

Figura 1 – Hierarquia do Aprendizado

2.3 Classificação automática de textos

A classificação de textos é a tarefa de associar textos em linguagem natural a rótulos pré-definidos. Esse problema vem sendo tratado desde os anos 60, porém só nos anos 90, a classificação de textos começou a ser amplamente aplicada, graças ao desenvolvimento de máquinas mais potentes e da facilidade de publicação de textos em forma eletrônica.

A classificação de textos é uma área que engloba conceitos de extração de informação e de aprendizado de máquina e possui muitas características em comum com outras tarefas como extração de conhecimento e mineração de textos.

A classificação de textos pode ser aplicada em uma grade variedade de contextos como, por exemplo: indexação automática de textos [Maron, 1961], identificação de autores de textos [Mosteller & Wallace, 1964], filtragem de e-mails [Graham, 2002], classificação hierárquica de páginas da internet [McCallum et al., 1998] e geração automática de metadados [Giles et al., 2003].

Considere um conjunto de documentos $D = \{d_1, d_2,..., d_{|D|}\}$ e um conjunto de categorias $C = \{c_1, c_2,..., c_{|C|}\}$. O problema da classificação automática de textos consiste em estimar uma função objetivo $\Phi : D \times C \rightarrow \{0,1\}$, que associa um valor booleano a cada par $(d_j, c_i) \in D \times C$. Um valor 1 associado ao par (d_j, c_i) indica que o documento d_j pertence à categoria c_i, enquanto que um valor 0 indica que o documento d_j não pertence à categoria c_i.

2.4
Classificação unirótulo e multirótulo

Na classificação de textos podem existir restrições relacionadas à quantidade de categorias do conjunto *C* e a quantidade de categorias associadas a cada documento d_j.

Considerando a restrição de quantidade de categorias no conjunto *C*, pode-se subdividir o problema da classificação em classificação binária, onde cada documento $d_j \in D$ está associado a uma categoria c_i ou ao seu complemento $\overline{c_i}$, e na classificação multicategoria, onde o conjunto de categorias *C* possui mais de duas categorias.

Considerando a restrição de quantidade de categorias associadas a cada documento, a classificação unirótulo corresponde ao caso em que um documento $d_j \in D$ está associado a exatamente uma categoria. Nesse caso, as categorias são mutuamente exclusivas.

A classificação multirótulo corresponde ao caso em que um documento $d_j \in D$ está associado a zero ou mais categorias, onde tais categorias não são mutuamente exclusivas.

Pode-se verificar que a classificação do tipo binária é um caso particular da classificação unirótulo.

2.5
Conjuntos de treinamento, validação e teste

A classificação de textos baseada nas técnicas de máquina de aprendizado necessita de um conjunto de documentos (corpus) $\Omega = \{d_1,...,d_{|\Omega|}\} \subset D$ pré-classificados nas categorias do conjunto $C = \{c_1,...,c_{|C|}\}$. Para a construção de um classificador divide-se o conjunto Ω em três subconjuntos disjuntos: um conjunto de treinamento $T_r = \{d_1,...,d_{|Tr|}\}$, um conjunto de validação $T_v = \{d_1,...,d_{|T_v|}\}$ e um conjunto de teste $T_e = \{d_1,...,d_{|T_e|}\}$. Em alguns casos, o conjunto de validação T_v é vazio.

O classificador é construído através do conjunto de treinamento T_r, e os parâmetros são ajustados através de repetidos testes realizados no conjunto de

validação T_v. Então, através do conjunto de teste T_e, são realizados testes para medir a eficiência do algoritmo de classificação.

Os subconjuntos são disjuntos para assegurar que os resultados experimentais obtidos, através do conjunto de teste, sejam de um conjunto diferente do usado para realizar o aprendizado, tornando os resultados estatisticamente válidos.

2.6 Representação de documentos

A forma mais simples de representar documentos é associar ao documento d_j um vetor de pesos $\vec{d}_j = \{w_{1j}, ..., w_{|V|j}\}$ onde V é o conjunto de termos que ocorrem em pelo menos um documento de T_r e o peso w_{kj} que representa, grosso modo, quanto o termo t_k contribui para a semântica do documento d_j.

Existem diversas abordagens que diferem na definição do que significa o termo t_k e como calcular os pesos w_{kj}.

A representação mais abordada na literatura de classificação de textos é conhecida como "bag of words". Nessa abordagem, cada termo corresponde a uma única palavra no conjunto de palavras do conjunto de treinamento.

Lewis, [1992], mostrou que representações de documentos mais sofisticadas, como frases, resultaram em um pior desempenho em experimentos rodados na base de notícias da Reuters. Além disso, [Scott & Matwin, 1999] acrescentaram informação semântica à tarefa de classificação de textos e não obtiveram resultados satisfatórios.

Em contraste, outros trabalhos apresentaram melhor desempenho na utilização de frases [Mladenić & Grobelnik, 1998] e reconhecimento de nomes próprios [Basili, 2000], comparados à representação tradicional "bag of words". Desta forma, a eficácia de modelos mais complexos ainda necessita de mais estudos.

Com relação ao cálculo dos pesos w_{kj}, a abordagem mais conhecida e muito comum em sistemas de recuperação de informação é a combinação de duas medidas, "Term Frequencie" (*tf*) e "Inverse Document Frequencie" (*idf*) definida como:

$$tfidf(t_k, d_j) = tf(t_k, d_j) idf(t_k, d_j)$$
$$tf(t_k, d_j) = \#(t_k, d_j) \qquad (1)$$
$$idf(t_k, d_j) = \log \frac{|Tr|}{\#Tr(t_k)}$$

Onde $\#(t_k, d_j)$ é a freqüência do termo t_k no documento d_j, $\#Tr(t_k)$ é o número de documentos do conjunto T_r que possuem pelo menos uma ocorrência do termo t_k.

O termo $tf(t_k, d_j)$ indica a importância do termo t_k no documento d_j. Dessa forma, se um termo é muito freqüente no documento d_j, então ele deve ser importante para representação do documento. Porém, um termo que seja freqüente em muitos documentos do conjunto T_r não é um termo representativo para o documento d_j. Desta forma, calcula-se a medida *idf*, que é inversamente proporcional à quantidade de documentos em que o termo t_k ocorre.

Com o propósito de igualar o tamanho dos vetores que representam os documentos, o valor da medida *tfidf* é normalizado:

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|V|} (tfidf(t_s, d_j))^2}} \qquad (2)$$

Outra abordagem muito usada em classificadores probabilísticos é representar um documento através de pesos binários, onde um valor 0 representa a ausência e o valor 1 representa a presença do termo no documento [Robertson & Sparck Jones, 1976].

A abordagem utilizada nesse trabalho representa um documento através de um vetor de freqüências, onde cada componente corresponde à freqüência do termo no documento.

2.7 Pré-processamento de documentos

A preparação dos textos é a primeira fase do processo de criação indutiva de classificadores de texto. Esta fase envolve a seleção de termos que melhor expressam o conteúdo de textos, ou seja, toda a informação que não refletir nenhuma idéia considerada importante é desconsiderada.

Desta forma, a seleção de termos reduz a quantidade de termos e, por conseguinte, a dimensão dos vetores que representam os documentos. Uma

redução da dimensão dos vetores implica em uma menor quantidade de memória utilizada e em menor processamento. Além disso, reduz a possibilidade de "overfitting", fenômeno que ocorre quando o classificador é ajustado de forma muito específica para o conjunto de treinamento, implicando em uma baixa perfomance na classificação de documentos não conhecidos pelo classificador [Sebastiani, 1999].

2.7.1 Stopwords

"Stopwords" são palavras consideradas não relevantes para a análise de textos. Na maioria das vezes são palavras auxiliares ou conectivas, não fornecendo nenhuma informação discriminativa na expressão do conteúdo do texto. Palavras como pronomes, artigos, preposições e conjunções podem ser consideraradas "stopwords".

Para cada língua existe um conjunto de "stopwords" (também conhecido como "stoplist"). Uma vez definida a lista de "stopwords", para cada documento, são retiradas as ocorrências no texto de todas as "stopwords".

2.7.2 Stemming

A tarefa de "stemming" consiste em agrupar palavras que possuem a mesma raiz morfológica.

Algumas técnicas de "stemming" serão apresentadas a seguir, com o objetivo de elucidar as diferentes abordagens utilizadas pelos algoritmos existentes.

2.7.2.1. Método de Stemmer S

O método mais simples de "stemming" é o stemmer S [Harman, 1991], no qual apenas alguns finais de palavras são removidos. Por exemplo, os finais de palavras da língua inglesa "ies", "es" e "s" (com exceções). Embora o stemmer S não descubra muitas fusões, alguns sistemas o usam, pois ele é conservador e raramente surpreende o usuário.

2.7.2.2. Método de Porter

O método de "stemming" de Porter [Porter, 1980] consiste na identificação das diferentes inflexões de uma mesma palavra e sua substituição por um radical comum. O algoritmo remove 60 sufixos diferentes em uma abordagem composta de cinco fases.

Termos com um "stem" comum muitas vezes possuem significados similares, por exemplo:

ESTUDO

ESTUDOS

ESTUDAR

ESTUDADO

2.7.2.3. Método de Lovins

O método de Lovins [Lovins, 1968] é composto de um único passo, é sensível ao contexto e usa um algoritmo de combinação mais longa para extrair em torno de 250 sufixos distintos. Tal método retira no máximo um sufixo por palavra, removendo o sufixo mais longo. Comparado aos outros dois métodos apresentados, este método é o mais agressivo.

2.7.3 Freqüência de Documentos (DF)

Freqüência de documentos (DF) é o número de documentos no qual um termo ocorre. A idéia desse método é calcular a freqüência DF de cada termo e remover o termo do vocabulário do corpus, se a freqüência for menor que um determinado limiar. A suposição básica é a de que termos raros são não-informativos para predizer a categoria ou não influenciam o desempenho global.

Freqüência de documentos é a técnica mais simples de redução de termos. Ela é facilmente escalável para conjuntos de muitos documentos, com uma complexidade computacional praticamente linear em relação à quantidade de documentos.

2.7.4
Ganho de Informação (IG)

Ganho de informação é amplamente empregado como um critério de importância de termos no campo de aprendizado de máquina. Através dessa técnica, é medida a "quantidade de informação" relativa à predição da categoria, pela presença ou ausência de um termo em um documento. Dado um conjunto de categorias $C = \{c_1, c_2,..., c_{|C|}\}$, o ganho de informação de um termo t_k é definido como:

$$G(t_k) = -\sum_{i=1}^{|C|} P(c_i)\log P(c_i)$$
$$+P(t_k)\sum_{i=1}^{|C|} P(c_i \mid t_k)\log P(c_i \mid t_k) \quad (3)$$
$$+P(\overline{t_k})\sum_{i=1}^{|C|} P(c_i \mid \overline{t_k})\log P(c_i \mid \overline{t_k})$$

Dado um conjunto de documentos de treinamento, para cada termo, é calculado o ganho de informação e são removidos do corpus os termos que possuírem um ganho de informação inferior a um limiar pré-determinado.

2.7.5
Informação Mútua (MI)

Informação mútua é um critério normalmente usado em modelagem estatística da linguagem em associações de palavras. Dado um termo t_k e uma categoria c_i, considere A o número de vezes que t_k e c_i co-ocorrem, B o número de vezes que t_k ocorre sem c_i, C o número de vezes que c_i ocorre sem t_k e N o número total de documentos de treinamento. A medida de informação mútua é definida como:

$$I(t_k, c_i) = \frac{P(t_k \wedge c_i)}{P(t_k)P(c_i)} \quad (4)$$

Estima-se a medida de informação mútua usando-se:

$$I(t_k, c_i) \approx \frac{\log A \times N}{(A+C)(A+B)} \quad (5)$$

A medida $I(t_k, c_i)$ tem o valor de zero caso t_k e c_i forem independentes. Para medir a importância de um termo globalmente, combinam-se as pontuações de um termo específicas para cada categoria em duas formas alternativas:

$$I_{avg}(t_k) = \sum_{i=1}^{|C|} P(c_i) I(t_k, c_i) \quad (6)$$

$$I_{\max}(t_k) = \max_{i=1}^{|C|} \{ I(t_k, c_i) \} \quad (7)$$

Desta forma, para cada termo, é calculada a informação mútua e são removidos do corpus os termos que possuírem um valor inferior a um limiar pré-determinado.

2.7.6 Estatística χ^2 (CHI)

A estatística χ^2 mede o grau de dependência entre um termo t_k e uma categoria c_i. Considerando A o número de vezes que t_k e c_i co-ocorrem, B o número de vezes que t_k ocorre sem c_i, C o número de vezes que c_i ocorre sem t_k e D o número de vezes que nem c_i nem t_k ocorrem, e N o número total de documentos de treinamento, a medida é definida por:

$$\chi^2(t_k, c_i) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)} \quad (8)$$

Assim como a medida de informação mútua, calcula-se o grau de importância global de um termo t_k de duas formas:

$$\chi^2(t_k) = \sum_{i=1}^{|C|} P(c_i) \chi^2(t_k, c_i) \quad (9)$$

$$\chi^2_{\max}(t_k) = \max_{i=1}^{|C|} \{ \chi^2(t_k, c_i) \} \quad (10)$$

2.8 Classificadores probabilísticos

Classificadores probabilísticos utilizam probabilidades para aproximar a função objetivo do problema da classificação de textos.

Desta forma, dado um conjunto de categorias $C = \{c_1, ..., c_{|C|}\}$ e um documento $\vec{d}_j = \{w_{1j}, ..., w_{|V|j}\}$ não conhecido, o algoritmo estima a probabilidade do documento pertencer a cada uma das categorias do conjunto C, representada

por $P(c_i \mid \vec{d}_j)$, e classifica o documento $\vec{d}_j$ na categoria com maior probabilidade estimada.

Desta forma, a função que aproxima a função objetivo da tarefa de classificação pode ser definida como:

$$\breve{\Phi}(\vec{d}_j, c_k) = 1 \leftrightarrow c^*(\vec{d}_j) = c_k$$
$$\breve{\Phi}(\vec{d}_j, c_k) = 0 \leftrightarrow c^*(\vec{d}_j) \neq c_k \quad (11)$$

$$\begin{aligned} c^*(\vec{d}_j) &= \arg\max_{c_i}\{P(c_i \mid \vec{d}_j)\} \\ &= \arg\max_{c_i}\{\log P(c_i \mid \vec{d}_j)\} \\ &= \arg\max_{c_i}\{\log(P(c_i)P(\vec{d}_j \mid c_i))\} \end{aligned} \quad (12)$$

Para calcular a probabilidade $P(c_i \mid \vec{d}_j)$, o algoritmo faz uso do teorema de Bayes:

$$P(c_i \mid \vec{d}_j) = \frac{P(c_i)P(\vec{d}_j \mid c_i)}{P(\vec{d}_j)} \quad (13)$$

Onde $P(c_i)$ é a probabilidade de um documento escolhido aleatoriamente pertencer à categoria c_i, $P(\vec{d}_j)$ é a probabilidade de um documento ser representado pelo vetor $\vec{d}_j$ e $P(\vec{d}_j \mid c_i)$ é a probabilidade de um documento ser representado pelo vetor $\vec{d}_j$, dado que ele pertence à categoria c_i.

2.8.1 Naive Bayes

O classificador naive Bayes assume que existe independência entre os termos de um documento. Tal hipótese simplificadora é muito criticada por não representar a realidade, porém Domingos & Pazzani, [1997] mostraram teoricamente que a suposição de independência de palavras na maioria dos casos não prejudica a eficiência do classificador.

Basicamente, existem dois tipos de modelos estatísticos para os classificadores naive Bayes que serão apresentados nas duas seções a seguir.

2.8.1.1. Modelo binário

O modelo binário representa um documento através de um vetor binário, onde um valor 0 na posição *k* significa que o documento não possui nenhuma ocorrência do termo t_k e um valor 1 significa que o documento possui pelo menos uma ocorrência do termo t_k. Desta forma, calcula-se:

$$P(\vec{d}_j \mid c_i) = \prod_{k=1}^{|V|} P(w_{kj} \mid c_i) \quad (14)$$

$$P(w_{kj} \mid c_i) = P(t_k \mid c_i)^{w_{kj}} (1 - P(t_k \mid c_i))^{1-w_{kj}} \quad (15)$$

$$P(t_k \mid c_i) = \frac{1 + \sum_{x=1}^{|Tr|} w_{kx} \Phi(d_x, c_i)}{2 + \sum_{s=1}^{|V|} \sum_{x=1}^{|Tr|} w_{sx} \Phi(d_x, c_i)} \quad (16)$$

Com o propósito de evitar que a probabilidade $P(\vec{d}_j \mid c_i)$ seja 0 simplesmente porque uma palavra do documento d_j não ocorre em nenhum documento da categoria c_i, o valor 1 é somado ao numerador e o valor 2 é somado ao denominador. Esta técnica é chamada de amortização de Laplace, sendo muito utilizada na literatura.

Substituindo-se a equação (15) na equação (14) tem-se:

$$\begin{aligned} P(\vec{d}_j \mid c_i) &= \prod_{k=1}^{|V|} P(t_k \mid c_i)^{w_{kj}} (1 - P(t_k \mid c_i))^{1-w_{kj}} \\ &= \left(\frac{P(t_k \mid c_i)}{1 - P(t_k \mid c_i)} \right)^{w_{kj}} (1 - P(t_k \mid c_i) \end{aligned} \quad (17)$$

Desta forma, através da equação (12), pode-se derivar:

$$c^*(\vec{d}_j) = \arg\max_{c_i} \left\{ \begin{array}{l} \log P(c_i) + \sum_{k=1}^{|V|} w_{kj} \log \frac{P(t_k \mid c_i)}{1 - P(t_k \mid c_i)} + \\ \sum_{k=1}^{|V|} w_{kj} \log(1 - P(t_k \mid c_i)) \end{array} \right\} \quad (18)$$

Para calcular $P(c_i)$:

$$P(c_i) = \frac{\sum_{k=1}^{|Tr|} \Phi(d_k, c_i)}{|Tr|} \quad (19)$$

2.8.1.2. Modelo multinomial

Já o modelo multinomial representa um documento através de um vetor de freqüências, onde o peso w_{kj} representa a freqüência do termo t_k no documento d_j.

O modelo se baseia na distribuição multinomial e calcula $P(\vec{d}_j \mid c_i)$ da seguinte forma:

$$P(\vec{d}_j \mid c_i) = P(\mid d_j \mid) \mid d_j \mid ! \prod_{k=1}^{|V|} \frac{P(t_k \mid c_i)^{w_{kj}}}{w_{kj}!} \quad (20)$$

$$P(t_k \mid c_i) = \frac{1 + \sum_{x=1}^{|Tr|} w_{kx} \Phi(d_x, c_i)}{|V| + \sum_{s=1}^{|V|} \sum_{x=1}^{|Tr|} w_{sx} \Phi(d_x, c_i)} \quad (21)$$

Substituindo-se a formula (20) na fórmula (12), tem-se:

$$c^*(d_j) = \arg\max_{c_i} \left(\log P(c_i) + \sum_{k=1}^{|V|} w_{kj} \log P(t_k \mid c_i) \right) \quad (22)$$

2.9 Medidas de desempenho

Nesta seção serão apresentadas as principais medidas de eficiência para classificadores e inclusive as medidas de eficiência utilizadas nos experimentos realizados sobre os dois algoritmos propostos.

2.9.1 Matriz de contingência

A matriz de contingência de um classificador oferece uma medida efetiva do modelo de classificação, uma vez que apresenta o número de classificações corretas versus as classificações preditas pelo algoritmo para cada classe. Dado um conjunto de teste T_e e um conjunto de categorias $C = \{c_1, c_2, ..., c_{|C|}\}$, os resultados são totalizados em duas dimensões: classificação verdadeira e classificação predita, onde :

$$M(c_i, c_j) = \sum_{\{d_k \in T_e : \Phi(d_k, c_i) = 1\}} \| \breve{\Phi}(d_k, c_j) = 1 \| \quad (23)$$

Classe	Predita c_1	Predita c_2	...	Predita $c_{\|C\|}$
Verdadeira c_1	$M(c_1, c_1)$	$M(c_1, c_2)$	...	$M(c_1, c_{\|C\|})$
Verdadeira c_2	$M(c_2, c_1)$	$M(c_2, c_2)$	...	$M(c_2, c_{\|C\|})$
⋮	⋮	⋮	⋮	⋮
Verdadeira $c_{\|C\|}$	$M(c_2, c_1)$	$M(c_2, c_1)$	$M(c_2, c_1)$	$M(c_{\|C\|}, c_{\|C\|})$

Tabela 1 – Matriz de contingência de um classificador.

O número de acertos para cada classe, se localiza na diagonal principal da matriz. Os demais elementos representam erros na classificação. A matriz de contingência de um classificador ideal possui todos esses elementos iguais à zero.

Por simplicidade, considere um problema de classificação binária de uma classe c_i. O problema deve classificar documentos de teste em c_i ou $\overline{c_i}$. Desta forma, tem-se a matriz de contingência:

Classe	Predita c_i	Predita $\overline{c_i}$
Verdadeira c_i	Verdadeiros positivos TP_i	Falsos negativos FN_i
Verdadeira $\overline{c_i}$	Falsos positivos FP_i	Verdadeiros negativos TN_i

Tabela 2 – Matriz de contingência de um classificador binário.

2.9.2 Precision e Recall

A eficiência da tarefa de classificação normalmente é calculada através de medidas clássicas de aquisição de informação, chamadas de recall (p) e de precision (π). Dada uma categoria c_i, recall e precision associadas à categoria c_i são definidas como:

$$p_i = \frac{TP_i}{TP_i + FN_i} \quad (24)$$

$$\pi_i = \frac{TP_i}{TP_i + FP_i} \quad (25)$$

Além das duas medidas associadas a cada categoria, também são usadas medidas globais:

Micro recall:

$$\mu\rho = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)} \quad (26)$$

Micro precision:

$$\mu\pi = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)} \quad (27)$$

Macro recall:

$$M\rho = \frac{\sum_{i=1}^{|C|} \rho_i}{|C|} \quad (28)$$

Macro precision:

$$M\pi = \frac{\sum_{i=1}^{|C|} \pi_i}{|C|} \quad (29)$$

Essas duas medidas globais normalmente apresentam resultados um pouco diferentes, principalmente devido à distribuição de documentos de treinamento em relação às categorias. Caso a distribuição de documentos seja disforme, um classificador com um bom desempenho deve dar mais ênfase para as medidas de macro recall e macro precision e menos ênfase para as medidas de micro recall e micro precision.

Para um classificador ter uma performance boa, não basta ter somente a medida de recall alta, ou a medida de precision alta, isoladamente.

Considere o exemplo de um classificador que classifica um documento em todas as classes. Tal classificador possuirá a medida recall muito alta, uma vez que a quantidade de falsos negativos é nula, porém uma medida precision baixa.

Desta forma, é necessária uma medida que combine as duas medidas. Muitas medidas de combinação de recall e precision, sendo que as mais utilizadas são:

- Eleven-point average precision: os parâmetros do classificador, (por exemplo, limiares), são ajustados com o propósito da medida p variar de 0.0, 0.1,..., 0.9, 1.0. Além disso, a medida π é calculada para as 11 iterações e é calculada a média dos 11 valores de π.
- Breakeven point: são realizadas diversas iterações, variando-se os parâmetros do classificador e é gerado gráfico das medidas p e π. O

breakeven point é o ponto onde as medidas p e π possuem o mesmo valor.

- A função F_β, que é uma combinação das medidas p e π, e é definida como: $F_\beta = \frac{(\beta^2+1)\pi\rho}{\beta^2\pi+\rho}$ (30). Normalmente, um valor β =1 é usado, dando igual importância para as medidas p e π, e é chamada de F_1. A medida F_1 foi utilizada nesse trabalho para medir o desempenho dos algoritmos propostos.

2.10 Métodos de avaliação de classificadores

Uma vez construído um classificador de textos, deve-se mostrar sua eficiência através de metodologias de avaliação de desempenho que permitam que seus resultados sejam comparáveis com outros classificadores.

Nesta seção serão apresentados os principais métodos de avaliação de desempenho de classificadores.

2.10.1 Resubstituição

O método de resubstituição consiste em construir e testar o classificador no mesmo conjunto de documentos, ou seja, o conjunto de documentos de teste é exatamente igual ao conjunto de documentos de treinamento. Tal método fornece uma estimativa altamente otimista da eficácia do algoritmo. Porém, este método não garante que o bom desempenho no conjunto de treinamento se estenda para conjuntos independentes de teste. Desta forma, diversos métodos de reamostragem foram propostos, os quais são descritos a seguir. Todos estão baseados no mesmo princípio: não deve haver documentos em comum entre os conjuntos de treinamento, validação e de teste.

2.10.2 Holdout

O método holdout divide os documentos em uma porcentagem fixa de documentos p para treinamento e $(1 - p)$ para teste, considerando normalmente p

> 1/2. Valores típicos são $p = 2/3$ e $(1 - p) = 1/3$, embora não existam fundamentos teóricos sobre estes valores. Uma vez realizada a divisão, é induzido um classificador a partir do conjunto de treinamento e são calculadas as medidas de eficiência sobre os documentos de teste. A desvantagem desse método é que ele possui apenas uma iteração, dependendo muito da qualidade da partição escolhida.

2.10.3 Amostragem aleatória

Na amostragem aleatória, são criadas *K* partições do conjunto de todos os documentos. Cada partição é criada selecionando de forma aleatória e sem reposição um número fixo de documentos pra treinamento. São realizados *K* experimentos e a medida de eficiência é a média das medidas de eficiência obtidas em cada experimento. A amostragem aleatória pode produzir melhores estimativas de erro que o método holdout.

2.10.4 K-Fold Cross Validation

Nesse método, os documentos são aleatoriamente divididos em *K* partições mutuamente exclusivas ("folds") de tamanho aproximadamente igual a n/K, onde *n* é o tamanho do conjunto de documentos. Então, são realizados *K* experimentos, onde, em cada experimento, uma partição diferente é escolhida para o teste e as $K - 1$ partições restantes são escolhidas para o treinamento. A medida de eficiência é a média das medidas de eficiência calculadas para cada uma das partições.

A grande vantagem desse método comparado ao anterior, é que todos os documentos são usados tanto para treinamento quanto para teste.

2.10.5 Leave-One-Out

O método leave-one-out é um caso especial de cross-validation. Possui uma complexidade computacional elevada e, portanto, é mais usado em amostras pequenas. Para um conjunto de *n* documentos, um exemplo é usado para teste e *n*

– 1 documentos são usados para treinamento. Este processo é repetido *n* vezes, cada vez escolhendo um exemplo diferente para teste.

A medida de eficiência deste método é a média das medidas de eficiência dos *n* experimentos realizados.

2.10.6 Bootstrap

No método bootstrap, o conjunto de treinamento possui o mesmo tamanho do conjunto de todos os documentos e é constituído de documentos selecionados aleatoriamente com reposição a partir de tal conjunto.

Desta forma, para um mesmo conjunto de treinamento, alguns documentos podem não estar incluídos, enquanto outros podem aparecer mais de uma vez. Os documentos que não aparecem no conjunto de treinamento são usados como conjunto de teste.

Geralmente, o processo de bootstrap é repetido inúmeras vezes, sendo que a medida de eficiência estimada é a média das medidas de eficiência obtidas em cada experimento.

3
Algoritmos propostos

Nesse trabalho foram desenvolvidos dois algoritmos que permitem classificar documentos em categorias de forma automática, com treinamento feito por usuários.

Tais algoritmos podem ser integrados em um ambiente completo de mineração de textos composto dos seguintes passos:

1. O usuário informa um conjunto de categorias de interesse e um conjunto de fontes de notícias na web. Por exemplo: categorias como soja, juros e violência das fontes de notícia Reuters, o Globo.
2. O programa retorna um conjunto de notícias das respectivas fontes especificadas pelo usuário.
3. O usuário classifica as notícias nas categorias anteriormente especificadas, tendo a possibilidade de classificar uma notícia em mais de uma categoria.
4. O programa retorna um novo conjunto de notícias classificadas nas categorias, podendo uma notícia estar classificada em mais de uma categoria.
5. O usuário avalia a classificação executada pelo programa e pode corrigir os falsos positivos e falsos negativos.
6. Volta para o passo 4.

A fase de treinamento e a fase de classificação ocorrem em conjunto com a realimentação do usuário, diferente do método clássico de treinamento em que primeiramente o algoritmo é treinado com os documentos para posterior classificação dos novos documentos nas categorias especificadas.

Nesse trabalho, são propostos dois algoritmos para resolver o problema da classificação automática de textos. Os algoritmos propostos são baseados no

método de classificação automática de textos naive Bayes utilizando o modelo multinomial e amortização de Laplace, descritos na seção 2.8.

3.1 Classificador pseudo-multirótulo

O primeiro algoritmo, chamado de pseudo-multirótulo, transforma o problema multirótulo em um problema unirótulo. Nesse algoritmo, o conjunto de categorias é estendido para suportar as combinações de categorias. Portanto, uma combinação de categorias também é considerada uma categoria.

Considere um conjunto de categorias $C = \{c_1, c_2,..., c_{|C|}\}$ e um conjunto de documentos $D = \{d_1, d_2,..., d_{|D|}\}$. Cada documento d_i está associado a uma combinação das categorias em C, que pode ser representado por um vetor binário $\vec{l} \in \{0,1\}^{|C|}$, onde a posição i está associada à categoria c_i, um valor 1 na posição i representa que o documento está associado à categoria c_i e um valor 0 representa que o documento não está associado à categoria c_i.

O conjunto de rótulos do algoritmo é definido como um conjunto $L \subseteq \{0,1\}^{|C|}$ de vetores binários $\vec{l}_j$, onde cada rótulo define um conjunto de documentos $\{d \in D \mid \forall i(\Phi(d, c_i) = \vec{l}[i])\}$. Por exemplo, o rótulo (0, 1, 1) representa o conjunto de documentos associados exatamente às categorias c_2 e c_3. Uma representação mais intuitiva do rótulo (0, 1, 1) seria $(\overline{c_1}, c_2, c_3)$.

Na fase de treinamento, dado um documento d_i associado ao rótulo $\vec{l}_j$, verifica-se se já foi realizado algum treinamento com o rótulo $\vec{l}_j$. Caso não tenha ocorrido, é criada uma nova categoria representada pelo rótulo $\vec{l}_j$. As palavras do documento d_i e suas freqüências são incluídas na categoria.

Já na fase de classificação, dado um documento d_i não conhecido, deve-se calcular a probabilidade $P(\vec{l}_j \mid d_i)$ do documento pertencer a cada uma das combinações de categorias representadas pelos vetores binários em $\vec{l} \in \{0,1\}^{|C|}$ e escolher a combinação com maior probabilidade. Adaptando-se a fórmula (22) para contemplar combinações de categorias, representadas por vetores, calcula-se:

$$\vec{l}^{*}(d_i) = \arg\max_{\vec{l}_j} \left(\log P(\vec{l}_j) + \sum_{k=1}^{|V|} w_{ki} \log P(t_k \mid \vec{l}_j) \right) \quad (31)$$

Como exemplo da fase de treinamento, considere o conjunto de treinamento composto pelos documentos $\{d_1, d_2, d_3, d_4\}$ pelo conjunto de categorias simples $\{c_1, c_2, c_3\}$ e pelas associações $(d_1, \{c_1, c_2\})$, $(d_2, \{c_2\})$, $(d_3, \{c_2\})$ e $(d_4, \{c_1, c_2, c_3\})$.

Para o documento d_1, é verificado se já foi realizado algum treinamento com a categoria representada pelo rótulo (1, 1, 0) ou $(c_1, c_2, \overline{c}_3)$. Como não foi realizado nenhum treinamento, é criada uma categoria representada pelo rótulo $(c_1, c_2, \overline{c}_3)$. Essa categoria é composta por todos os documentos que estão associados à exatamente c_1 e à c_2. Após a criação da categoria, são retiradas as "stopwords" de d_1, todas as palavras são transformadas em minúsculas e são calculadas as freqüências de cada palavra no texto de d_1. Finalmente, as palavras e suas respectivas freqüências são incluídas na categoria $(c_1, c_2, \overline{c}_3)$.

No treinamento a partir do documento d_2, é criada uma categoria representada pelo rótulo $(\overline{c}_1, c_2, \overline{c}_3)$. Essa categoria é composta por todos os documentos que estão associados à exatamente c_2. Então, as palavras de d_2 e suas respectivas freqüências são incluídas na categoria $(\overline{c}_1, c_2, \overline{c}_3)$.

Já no treinamento a partir do documento d_3 não é criada nenhuma categoria, pois a categoria associada ao documento d_3 já foi criada anteriormente.

Na fase de classificação, dado um documento d_5 não conhecido pelo sistema, o algoritmo calcula $\vec{l}^*(d_5)$, onde $\vec{l}_j \in \{(c_1, c_2, \overline{c}_3), (\overline{c}_1, c_2, \overline{c}_3), (c_1, c_2, c_3)\}$.

O problema de tal técnica é que são necessários exemplos de documentos para cada combinação de categorias, o que nem sempre é possível.

Além disso, um documento exemplo de uma combinação de n categorias (por exemplo, um documento pertencente à combinação "milho, grão, agricultura") não seria também tratado como exemplo das 2^n-2 subcategorias (por exemplo, "grão e agricultura", "milho e agricultura", "milho e grão", "milho", "grão", "agricultura").

A complexidade teórica da fase de treinamento do algoritmo é $O(|T_r||V|)$. Desconsiderando a fase de contagem da freqüência das palavras em cada documento, para cada documento do conjunto T_r, são incluídas todas as suas palavras na combinação de categorias associada, o que custa no pior caso, $O(|V|)$.

Já a fase de classificação tem a complexidade teórica de O ($|T_e||V||T_r|$). Para cada documento, é necessário selecionar a combinação de categoria com maior probabilidade do conjunto L de combinações de categorias criadas na fase de treinamento. Para selecionar a combinação de categoria com maior probabilidade se gasta O($|V||T_r|$), uma vez que, no pior caso, existem $|T_r|$ combinações de categorias, ou seja, um documento de treinamento para cada combinação.

3.2 Classificador multirótulo

Nesse algoritmo, um rótulo é uma combinação de categorias em C e é representado por um vetor $\vec{l}$, onde cada posição corresponde a uma categoria e pode assumir os valores: "0", "1" e "?". Um valor "1" na posição i do vetor $\vec{l}$ representa que o rótulo é composto de documentos associados à categoria c_i, um valor "0" representa a ausência de associações entre os documentos do rótulo e a categoria c_i, e um valor "?" representa que o rótulo pode ser composto tanto por documentos associados à c_i quanto a documentos não associados à c_i.

Formalmente, o rótulo $\vec{l}$ define um conjunto de documentos $\{d \in D \mid \forall i(\vec{l}[i] \neq ?) \rightarrow \vec{l}[i] = \Phi(d, c_i)\}$. Como exemplo, considere o rótulo (?, 1, 1). Tal rótulo define o conjunto de documentos que estão associados pelo menos às categorias c_2 e c_3. Uma forma mais intuitiva de representar o rótulo seria (c_2, c_3).

Na fase de treinamento, dado um conjunto D de documentos de treino, e um conjunto de categorias C, o algoritmo primeiramente cria dois conjuntos de rótulos $L \subseteq \{0,1\}^{|C|}$ e $L' \subseteq \{1,?\}^{|C|}$.

Para isso, para cada documento, é verificada a combinação de rótulos associada. Dado um documento d_i de treino e uma combinação de rótulos associada representada pelo vetor $\vec{l}_j$, é verificado se já foi criada uma categoria representada pelo vetor $\vec{l}_j$. Caso não tenha sido criada, são criadas duas categorias $\vec{l}_j \in \{0,1\}^{|C|}$ e $\vec{l}'_j \in \{1,?\}^{|C|}$, onde $\vec{l}_j$ é a categoria composta de documentos associados exatamente à combinação de rótulos de d_i e $\vec{l}'_j$ é a categoria composta de documentos associados pelo menos à combinação de rótulos de d_i.

Uma vez criadas todas as categorias, para cada documento d_i são calculadas as freqüências de cada palavra no texto de d_i. Verifica-se, então, a combinação $\vec{l}_j$ associada ao documento d_i e são incluídas as palavras do documento e suas freqüências na combinação de categorias representada pelo rótulo $\vec{l}_j$ e em todas as subcombinações $\vec{l}'_k \in L'$ onde $\forall i(\vec{l}'_k[i]=1) \rightarrow (\vec{l}_j[i]=1)$, ou seja, a combinação de categorias $\vec{l}'_k$ está contida na combinação $\vec{l}_j$. Por exemplo, se o documento d_i está associado ao rótulo $(c_1, c_2, \overline{c}_3, c_4)$, ele será incluído nos rótulos $(c_1, c_2, \overline{c}_3, c_4)$, (c_1, c_2, c_4), (c_1, c_2), (c_1, c_4), (c_2, c_4), (c_1), (c_2), (c_4).

Na fase de classificação, considere os conjuntos de rótulos criados pelo treinamento $L = \{\vec{l}_1, \vec{l}_2, ..., \vec{l}_{|L|}\} \in \{0,1\}^{|C|}$, $L' = \{\vec{l}'_1, \vec{l}'_2, ..., \vec{l}'_{|L'|}\} \in \{1,?\}^{|C|}$ e um documento d_i não conhecido pelo classificador.

Primeiramente, deve-se calcular a probabilidade $P(\vec{l}'_j \mid d_i)$ do documento pertencer a cada uma das combinações $\vec{l}'_j \in L'$ compostas apenas de uma categoria e escolher a combinação com maior probabilidade, representada por $\vec{l}'^{(1)}$.

Porém, não é necessário calcular exatamente $P(\vec{l}_j \mid d_i)$, uma vez que $P(d_i)$ é constante. Assim, através da equação (31), é escolhida a combinação $\vec{l}'^{(1)}$, que possui a maior probabilidade.

Posteriormente, escolhe-se a combinação com maior probabilidade $\vec{l}'^{(2)}$, do conjunto de combinações $\vec{l}'_j \in L'$ compostas por duas categorias, onde uma das categorias da combinação pertence à combinação $\vec{l}'^{(1)}$, escolhida anteriormente.

Então, se compara a probabilidade $P(\vec{l}'^{(2)} \mid d_i)$ com a probabilidade $P(\vec{l}^{(1)} \mid d_i)$, onde $\vec{l}^{(1)} \in L$ é a combinação tal que $\forall k(\vec{l}'^{(1)}[k]=1) \rightarrow (\vec{l}^{(1)}[k]=1)$ e $\forall k(\vec{l}'^{(1)}[k]=?) \rightarrow (\vec{l}^{(1)}[k]=0)$. Por exemplo, se $C = \{c_1, c_2, c_3\}$ e $\vec{l}'^{(1)} = (c_1)$, então $\vec{l}^{(1)} = (c_1, \overline{c}_2, \overline{c}_3)$.

Para ser realizada a comparação das probabilidades, calcula-se $\arg\max_{\vec{r}} \left\{ P(\vec{r} \mid d_i), \vec{r} \in \{\vec{l}'^{(2)}, \vec{l}^{(1)}\} \right\}$. Se $P(\vec{l}'^{(2)} \mid d_i) < P(\vec{l}^{(1)} \mid d_i)$, então o documento d_i é classificado em $\vec{l}^{(1)}$. Caso contrário, o algoritmo continua aumentando a quantidade de categorias em cada combinação, até que a condição de parada

$P(\vec{l}'^{(k)} \mid d_i) < P(\vec{l}^{(k-1)} \mid d_i)$ seja satisfeita, ou a quantidade de categorias em cada combinação seja |*C*|. Caso a quantidade de categorias em cada combinação seja |*C*| e a condição de parada não seja satisfeita o documento d_i é classificado na combinação de categorias $\vec{l}^{(k)}$, que representa a combinação composta por todas as categorias em *C*.

Como exemplo, considere um documento d_1, um conjunto *C* de categorias $\{c_1, c_2, c_3\}$ e um conjunto de rótulos $L' = \{(c_1), (c_2), (c_1, c_2), (c_1, c_3)\}$. Inicialmente, o algoritmo induz uma associação entre o documento d_1 e uma das categorias em *C*, calculando a probabilidade do documento d_1 pertencer a cada uma das categorias em *C*. Caso a categoria induzida tenha sido c_1, são analisadas todas as combinações de categorias presentes no conjunto de rótulos *L´*, onde uma das categorias é c_1, ou seja, (c_1, c_2) e (c_1, c_3). Então, é escolhido o rótulo com maior probabilidade. Caso o rótulo com maior probabilidade seja (c_1, c_2), a regra de decisão deverá induzir se o documento pertence exatamente a c_1 ou se pertence pelo menos a c_1 e c_2. A regra de decisão pode ser visualizada pela figura 2. O algoritmo induz se o documento pertence à região marcada na figura com linhas diagonais, da região marcada com linhas horizontais.

$$P(c_1, c_2 \mid d_1) > P(c_1, \overline{c}_2, \overline{c}_3 \mid d_1)?$$

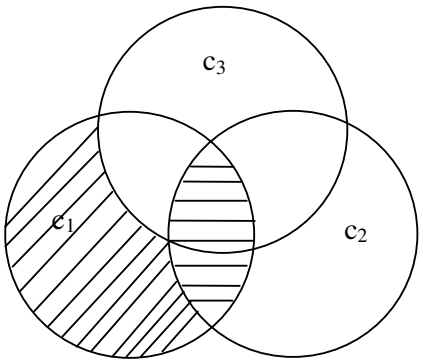

Figura 2 - Exemplo da regra de decisão do algoritmo multilrótulo proposto.

A complexidade teórica da fase de treinamento do algoritmo é $O(|T_r||V||T_e|)$. Para cada documento pertencente ao conjunto de treinamento T_r, no pior caso, existem T_r combinações de categorias pertencentes ao conjunto de combinações de categorias.

A complexidade teórica da fase de classificação do algoritmo é $O(|T_r||V||T_e|)$. No primeiro passo, é selecionada a combinação de categorias composta de apenas 1 categoria. Uma vez escolhida a combinação de categorias composta de apenas 1 categoria, seleciona-se a combinação de categorias compostas de 2 categorias, onde uma das categorias pertence à combinação de categorias escolhida no primeiro passo. Assim, no pior caso, compara-se todas as combinações de categorias existentes na base de treinamento.

Como exemplo da fase de treinamento, considere o conjunto de treinamento composto pelo conjunto de documentos $D = \{d_1, d_2, d_3, d_4, d_5\}$, pelo conjunto de categorias $C = \{c_1, c_2, c_3, c_4\}$ e pelas associações $(d_1, \{c_1, c_2\})$, $(d_2, \{c_2\})$, $(d_3, \{c_1, c_2, c_3\})$, $(d_4, \{c_1, c_3\})$ e $(d_5, \{c_1\})$.

Primeiramente, são criados os conjuntos de rótulos L e L'. Para isso, são verificados para cada documento as respectivas categorias associadas. Assim, para o documento d_1, são criados os rótulos (c_1, c_2) e $(c_1, c_2, \overline{c}_3, \overline{c}_4)$, para d_2, (c_2) e $(\overline{c}_1, c_2, \overline{c}_3, \overline{c}_4)$, para d_3, (c_1, c_2, c_3) e $(c_1, c_2, c_3, \overline{c}_4)$, para d_4, (c_1, c_3) e $(c_1, \overline{c}_2, c_3, \overline{c}_4)$ e d_5, (c_1) e $(c_1, \overline{c}_2, \overline{c}_3, \overline{c}_4)$.

Desta forma os conjuntos de rótulos são $L' = \{(c_1), (c_2), (c_1, c_2), (c_1, c_3), (c_1, c_2, c_3)\}$ e $L = \{(c_1, \overline{c}_2, \overline{c}_3, \overline{c}_4), (\overline{c}_1, c_2, \overline{c}_3, \overline{c}_4), (c_1, c_2, \overline{c}_3, \overline{c}_4), (c_1, \overline{c}_2, c_3, \overline{c}_4), (c_1, c_2, c_3, \overline{c}_4)\}$.

Depois de criados os conjuntos de rótulos, para cada documento são analisadas suas palavras, são retiradas as "stopwords", calculadas suas freqüências no texto e inseridas nos rótulos ao qual o documento pertence.

Para o documento d_1, suas palavras e freqüências são incluídas nos rótulos $(c_1, c_2, \overline{c}_3, \overline{c}_4)$, (c_1, c_2), (c_1) e (c_2).

Para o documento d_2, suas palavras e freqüências são incluídas nos rótulos $(\overline{c}_1, c_2, \overline{c}_3, \overline{c}_4)$ e (c_2).

Para o documento d_3, suas palavras e freqüências são incluídas nos rótulos $(c_1, c_2, c_3, \overline{c}_4)$, (c_1, c_2, c_3), (c_1, c_2), (c_1, c_3) e (c_2).

Já no caso do documento d_4, suas palavras e freqüências são incluídas nos rótulos $(c_1, \overline{c}_2, c_3, \overline{c}_4)$, (c_1, c_3) e (c_1).

Finalmente para o documento d_5, suas palavras são incluídas nos rótulos $(c_1, \overline{c}_2, \overline{c}_3, \overline{c}_4)$ e (c_1).

Como exemplo de classificação, considere o documento d_6 não conhecido pelo classificador.

Primeiramente, o classificador escolhe o rótulo com uma categoria com maior probabilidade. Assim, o classificador deve calcular $\arg\max_{\vec{r}} \left\{P(\vec{r} \mid d_i), \vec{r} \in \{(c_1),(c_2)\}\right\}$.

Supondo que o rótulo escolhido tenha sido (c_1), então o classificador analisa todos os rótulos com duas categorias, onde uma das categorias é c_1, ou seja, os rótulos (c_1, c_2) e (c_1, c_3). Caso o rótulo com maior probabilidade seja (c_1, c_3), então o classificador compara $P(c_1, c_3 \mid d_6)$ com $P(c_1, \overline{c}_2, \overline{c}_3, \overline{c}_4 \mid d_6)$. Se $P(c_1, c_3 \mid d_6) < P(c_1, \overline{c}_2, \overline{c}_3, \overline{c}_4 \mid d_6)$, o documento d_6 é classificado no rótulo $(c_1, \overline{c}_2, \overline{c}_3, \overline{c}_4)$.

4
Experimentos

4.1
Introdução

Foram realizados experimentos com os dois algoritmos propostos no capítulo anterior em duas bases de documentos, Reuters-21578 e Ohsumed. Primeiramente serão descritas cada uma das bases de documentos. Então, serão apresentados os resultados obtidos nas duas bases.

4.2
Bases de dados

Nessa seção serão descritas as duas bases de documentos onde foram aplicados experimentos com a finalidade de comparar os dois algoritmos propostos no capítulo anterior. As duas bases de documentos são amplamente mencionadas na literatura de máquina de aprendizado.

4.2.1
Reuters-21578

A base de testes Reuters-21578 vem sendo amplamente utilizada, nos últimos dez anos, para testar algoritmos de classificação automática de textos. A base é composta de um conjunto de 21.578 notícias que foram publicadas na rede de notícias Reuters, em 1987, e classificadas de acordo com 135 categorias, a maioria sobre economia e negócios. A base foi originalmente categorizada pelo Carnegie Group, Inc. and Reuters, Ltd. na criação do sistema de classificação automática CONSTRUE [Hayes & Weinstein, 1990] e posteriormente foi coletada e formatada por David Lewis.

Esta coleção possui muitas características importantes que a tornam uma base de dados interessante para os experimentos da classificação automática de textos:

- A base de dados é multirótulo, ou seja, um documento pode estar associado a mais de uma categoria, sendo uma situação muito mais realista.
- A distribuição de documentos pelas 135 categorias não é uniforme.
- Existem muitas relações semânticas implícitas entre as categorias da base (por exemplo, as categorias “wheat” e “grain” estão claramente associadas, pois sempre que um documento pertence à “wheat” também pertence à “grain”).

A base de testes Reuters-21578 certamente é uma base desafiadora para sistemas de classificação automática de textos baseados nas técnicas de aprendizagem de máquina, uma vez que muitas categorias possuem muito poucos exemplos de treino, tornando a construção indutiva de um classificador uma tarefa difícil.

Os documentos realmente utilizados na maioria dos experimentos de classificação automática de textos são apenas 12.902, uma vez que 8.676 documentos não foram considerados na categorização da base.

Com a finalidade de tornar os resultados experimentais comparáveis, partições compostas do conjunto de treinamento e do conjunto de teste foram definidas pelos criadores da base de 12.902 documentos. Salvo algumas exceções, os pesquisadores têm usado a partição ModApté, no qual 9.603 documentos foram selecionados para o conjunto de documentos de treinamento, enquanto 3.299 foram selecionados para compor o conjunto de documentos de teste.

Dentre os 12.902 documentos, alguns documentos não possuem categoria associada, porém, diferentemente dos 8.676 que não foram considerados, tais documentos não estão associados a nenhuma categoria, porque na tarefa de categorização da base não foi encontrada nenhuma categoria que pudesse associá-los.

Dentre as 135 categorias, 20 categorias na partição ModApté não possuem nenhum documento de treino. Portanto, tais categorias nunca foram consideradas em nenhum experimento de classificação automática de textos.

Uma vez que as 115 categorias da partição ModApté possuem pelo menos um documento de treinamento, em princípio, podem ser usadas em qualquer

experimento. Porém, muitos pesquisadores têm preferido executar seus experimentos em diferentes subconjuntos das 115 categorias.

Dentre os subconjuntos, os mais populares são:

- O conjunto das 10 categorias que possuem a maior quantidade de documentos de treinamento, chamado de R(10). Esse conjunto foi utilizado em [Bennett et. al, 2002; McCallum & Nigam, 1998; Nigam et al., 2000; Tong & Koller, 2001].
- O conjunto das 90 categorias que possuem pelo menos um documento de treinamento e um documento de teste, chamado de R(90). Esse conjunto, que está presente na maioria dos experimentos já relatados, foi utilizado em [Chai et al., 2002; Joachims, 1998; Lam & Lai, 2001; Moschitti, 2003a; Sebastiani et al., 2000; Yang & Liu, 1999].
- O conjunto das 115 categorias que possuem pelo menos um documento de treinamento, chamado de R(115). Esse conjunto foi utilizado em [Caropreso et al., 2001; Dumais et al., 1998; Galavotti et al., 2000].

Vale notar que $R(10) \subset R(90) \subset R(115)$.

4.2.2 Ohsumed

A coleção de teste Ohsumed é um subconjunto de 348.566 documentos da base MEDLINE (uma base on-line de textos sobre medicina), compilada por William Hersh e proveniente de 270 jornais médicos por um período de cinco anos (1987- 1991).

Todos os documentos possuem títulos, porém apenas 233.445 possuem abstracts. Os documentos foram manualmente categorizados em 14.321 categorias definidas no thesaurus Mesh (Medical Subject Heading).

A base Ohsumed é considerada uma base mais difícil uma vez que não existe uma relação bem definida entre as palavras e categorias. No caso da Reuters, o próprio nome da categoria é uma palavra muito freqüente no conjunto de documentos definido pela categoria.

A base de documentos médicos Ohsumed foi utilizada nos trabalhos [Joachims, 1998; Moschitti, 2003a].

4.3 Experimentos Realizados

Os algoritmos propostos na seção anterior foram desenvolvidos na linguagem C# e rodados em uma máquina AMD Semprom 2.2 GHz com 512 MB de memória RAM com Windows XP e Microsoft .NET Framework 2.0. Cada documento foi representado como uma tabela de palavras, definida por um conjunto de tuplas, relacionando cada palavra do documento com sua respectiva freqüência. Além disso, todas as palavras foram transformadas em minúsculas e foram retiradas as "stopwords" definidas no conjunto de 571 "stopwords" da língua inglesa SMARTLIST desenvolvida inicialmente para o sistema SMART [Salton, 1971].

Todos os documentos considerados nos experimentos possuem pelo menos uma categoria associada.

4.3.1 Experimentos com a base Reuters R(10)

No conjunto R(10), primeiramente foi realizado um experimento utilizando a partição ModApté, com 6.490 documentos de treinamento, 2.545 documentos de teste, vocabulário de 21955 palavras e 39 combinações das 10 categorias da base.

As tabelas 3, 4 e 5 apresentam os resultados do algoritmo pseudo-multirótulo. Já as tabelas 6, 7 e 8 apresentam os resultados do algoritmo mulirótulo.

Categoria	Recall	Precision	F1
acq	97,36	97,63	97,49
corn	71,43	72,73	72,07
crude	94,71	90,40	92,51
earn	98,16	98,61	98,39
grain	81,88	99,19	89,71
interest	58,78	87,50	70,32
money-fx	94,41	78,24	85,57
ship	71,91	91,43	80,50
trade	91,45	78,10	84,25
wheat	76,06	77,14	76,60

Tabela 3 – Resultados do algoritmo pseudo-multirótulo na base R(10) para as 10 categorias que compõem a base.

Micro Recall	92,54
Micro Precision	93,58
Micro F1	93,05
Macro Recall	83,61
Macro Precision	87,10
Macro F1	84,74

Tabela 4 – Resultados globais do algoritmo pseudo-multirótulo na base R(10).

Classe	Segundos
Treinamento	3,812
Classificação	7,203

Tabela 5 – Tempo de execução da fase de treinamento e da fase de classificação do algoritmo pseudo-multirótulo na base R(10).

Categoria	Recall	Precision	F1
acq	96,80	97,62	97,21
corn	87,50	62,03	72,59
crude	96,30	88,78	92,39
earn	97,98	98,61	98,29
grain	89,26	97,08	93,01
interest	74,05	86,61	79,84
money-fx	94,41	75,78	84,08
ship	79,78	85,54	82,56
trade	90,60	84,80	87,60
wheat	83,10	62,11	71,08

Tabela 6 – Resultados do algoritmo multirótulo na base R(10) para as 10 categorias que compõem a base.

Micro Recall	94,26
Micro Precision	92,11
Micro F1	93,17
Macro Recall	88,98
Macro Precision	83,90
Macro F1	85,86

Tabela 7 – Resultados globais do algoritmo multirótulo na base R(10).

Classe	Segundos
Treinamento	4,125
Classificação	3,218

Tabela 8 – Tempo de execução da fase de treinamento e da fase de classificação do algoritmo multirótulo na base R(10).

Após o experimento utilizando a partição ModeApté, foi realizado um experimento para verificar como se comportam os dois algoritmos em função da quantidade de documentos de treinamento.

Para isso, para cada um dos algoritmos, foram gerados 10 partições aleatórias, onde 6.490 documentos eram de treinamento e 2.545 documentos eram de teste.

Para cada partição rodou-se o algoritmo 11 vezes, mantendo-se os documentos de teste fixos (2.545 documentos) e variando a quantidade de documentos de treinamento pertencentes ao conjunto de 6.490 documentos, iniciando com 6.490 e dividindo por 2 até a quantidade de documentos de treinamento chegar a 6.

Uma vez gerada a curva para cada uma das 10 partições calculou-se a média das 10 curvas.

As figuras abaixo apresentam os resultados do experimento para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo.

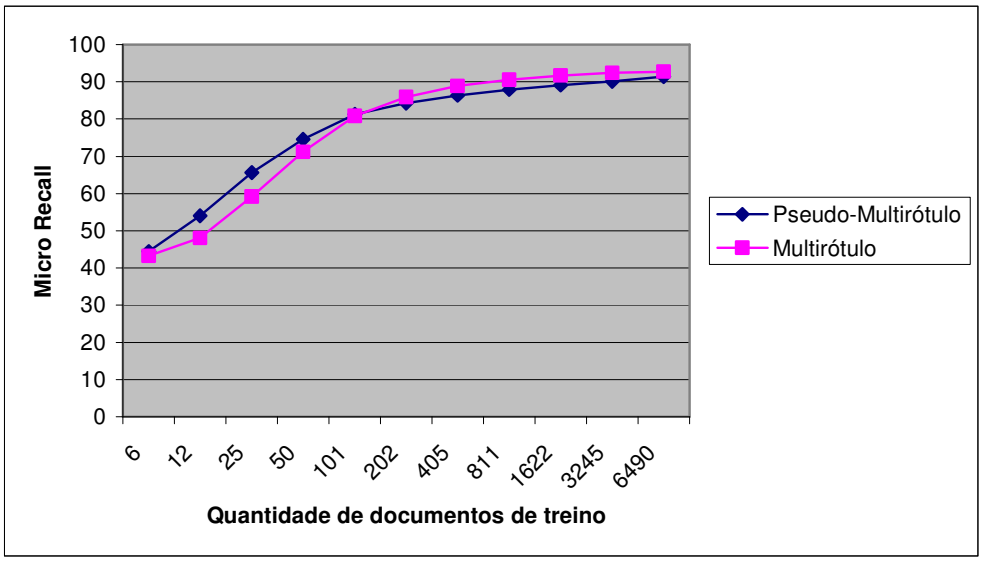

Figura 3 - Resultados Micro Recal para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(10).

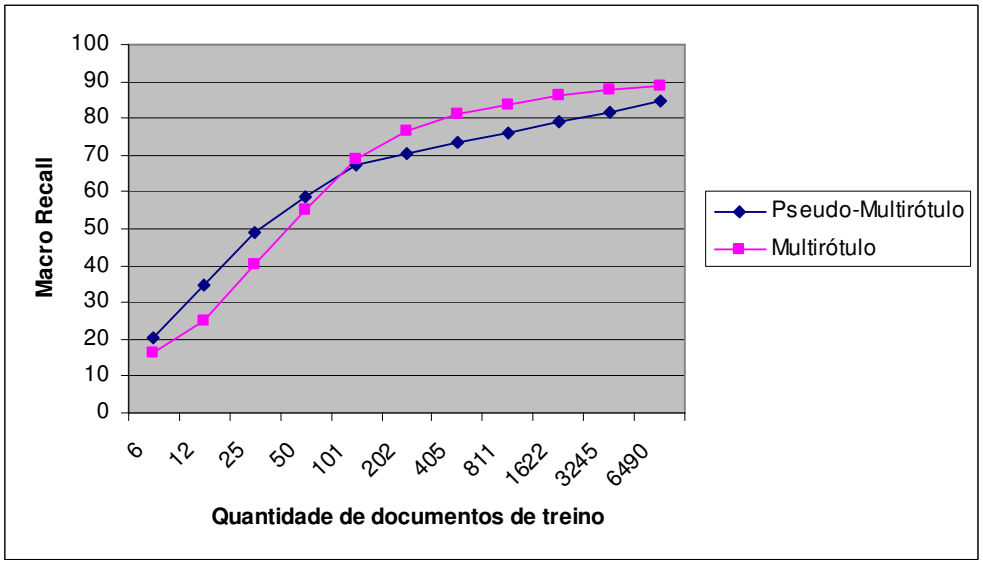

Figura 4 - Resultados Macro Recall para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(10).

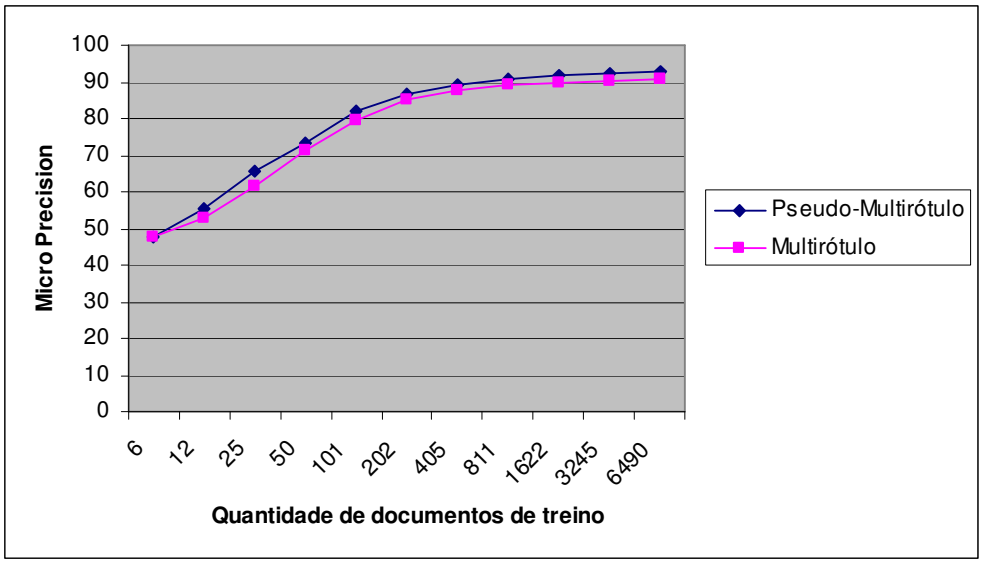

Figura 5 - Resultados Micro Precision para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(10).

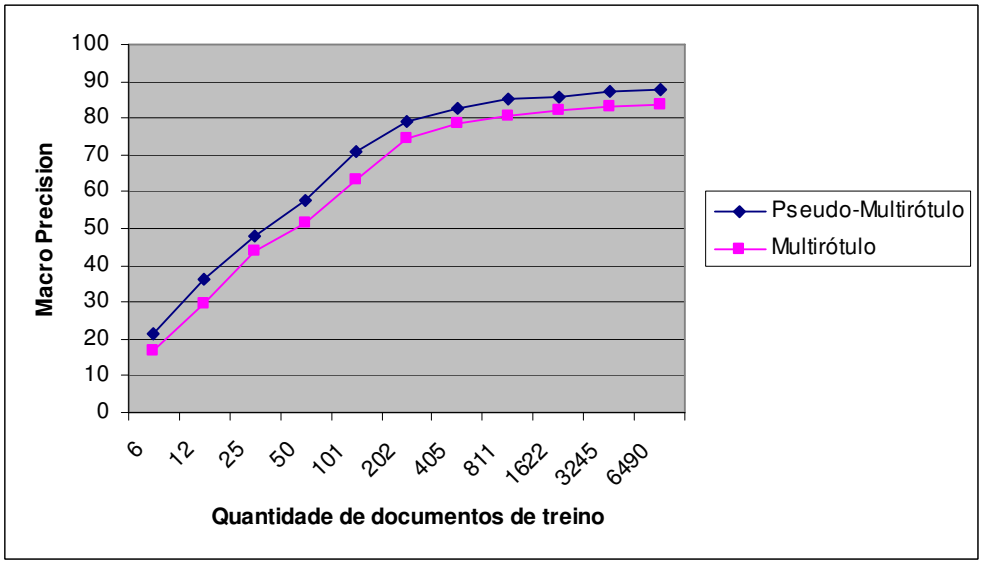

Figura 6 - Resultados Macro Precision para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(10).

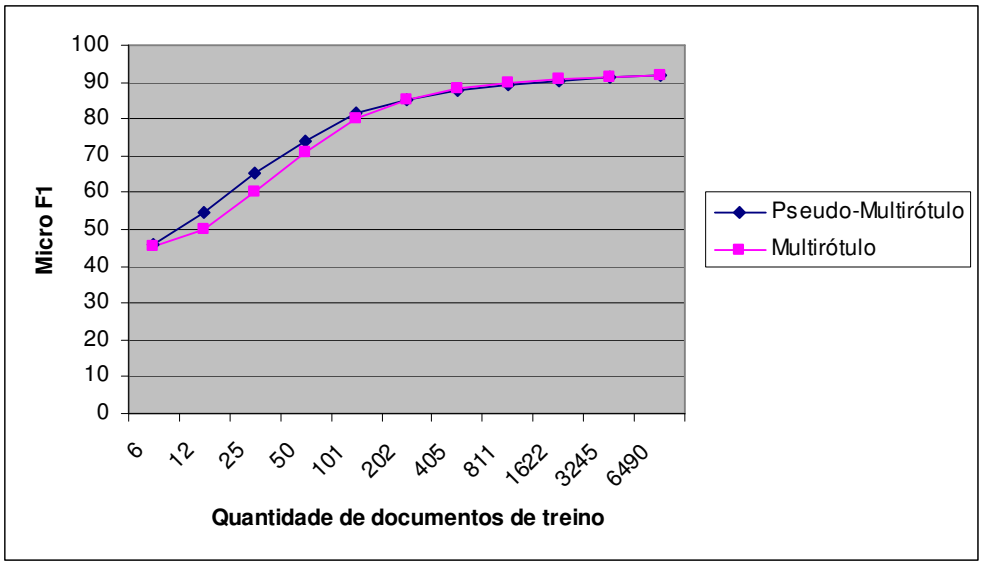

Figura 7 - Resultados Micro F1 para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(10).

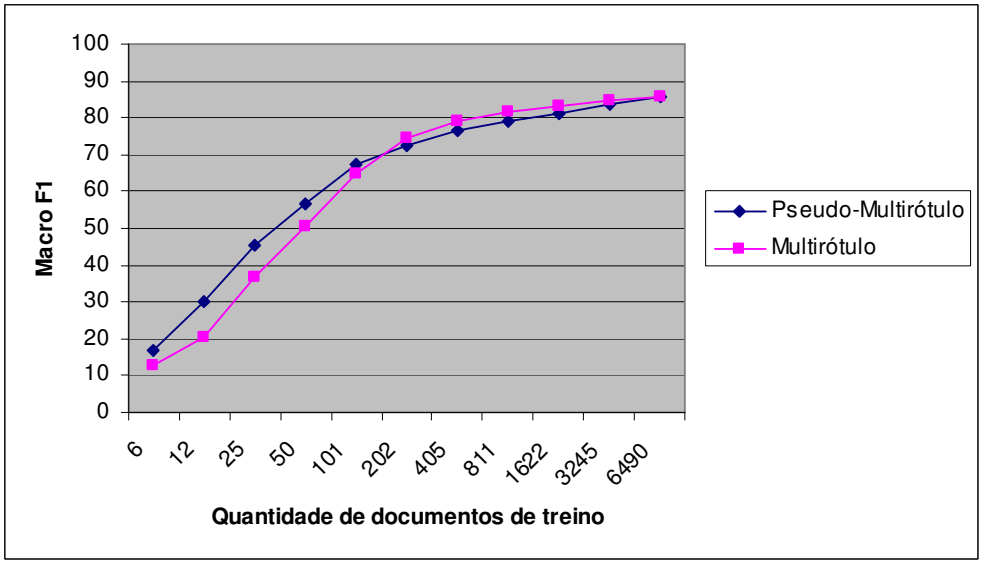

Figura 8 - Resultados Macro F1 para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(10).

4.3.2
Experimentos com a base Reuters R(90)

Conforme concluído nos experimentos de Sebastiani & Debole [2004], a construção indutiva de classificadores na base R(10) é menos árdua que na base

R(90). Por isso, foram realizados experimentos para os dois algoritmos propostos na base R(90).

Assim como no conjunto R(10), primeiramente foi realizado um experimento utilizando a partição ModApté, com 7.770 documentos de treinamento, 3.019 documentos de teste, vocabulário de 24.244 palavras e 365 combinações das 90 categorias da base.

As tabelas 7, 8 e 9 apresentam os resultados do algoritmo pseudo-multirótulo. Já as tabelas 10, 11 e 12 apresentam os resultados do algoritmo mulirótulo.

Categoria	Recall	Precision	F1
acq	97,64	92,73	95,12
corn	35,71	74,07	48,19
crude	93,12	67,69	78,40
earn	98,25	95,36	96,78
grain	69,80	84,55	76,47
interest	64,12	85,71	73,36
money-fx	91,62	68,91	78,66
ship	67,42	88,24	76,43
trade	88,03	47,03	61,31
wheat	64,12	85,71	73,36

Tabela 9 – Resultados do algoritmo pseudo-multirótulo na base R(90) para as 10 categorias com maior quantidade de documentos de treinamento.

Micro Recall	72,58
Micro Precision	83,14
Micro F1	77,50
Macro Recall	19,13
Macro Precision	38,93
Macro F1	21,92

Tabela 10 – Resultados globais do algoritmo pseudo-multirótulo na base R(90).

Classe	Segundos
Treinamento	5,750
Classificação	56,344

Tabela 11 – Tempo de execução da fase de treinamento e da fase de classificação do algoritmo pseudo-multirótulo na base R(90).

Categoria	Recall	Precision	F1
acq	96,80	97,62	97,21
corn	87,50	62,03	72,59
crude	96,30	88,78	92,39
earn	97,98	98,61	98,29
grain	89,26	97,08	93,01
interest	74,05	86,61	79,84
money-fx	94,41	75,78	84,08
ship	79,78	85,54	82,56
trade	90,60	84,80	87,60
wheat	83,10	62,11	71,08

Tabela 12 – Resultados do algoritmo multirótulo na base R(90) para as 10 categorias com maior quantidade de documentos de treinamento.

Micro Recall	79,68
Micro Precision	77,70
Micro F1	78,68
Macro Recall	29,07
Macro Precision	41,99
Macro F1	30,73

Tabela 13 – Resultados globais do algoritmo multirótulo na base R(90).

Classe	Segundos
Treinamento	18,594
Classificação	20,344

Tabela 14 – Tempo de execução da fase de treinamento e da fase de classificação do algoritmo multirótulo na base R(90).

Após o experimento utilizando a partição ModeApté, foi realizado um experimento para verificar como se comportam os dois algoritmos em função da quantidade de documentos de treinamento.

Para isso, para cada um dos algoritmos, foram geradas 10 partições aleatórias, onde 7.770 documentos eram de treinamento e 3.019 documentos eram de teste.

Para cada partição rodou-se o algoritmo 11 vezes, mantendo-se os documentos de teste fixos (3.019 documentos) e variando a quantidade de documentos de treinamento pertencentes ao conjunto de 7.770 documentos, iniciando com 7.770 e dividindo por 2 até a quantidade de documentos de treinamento chegar a 7.

Uma vez gerada a curva para cada um das 10 partições calculou-se a média das 10 curvas.

As figuras abaixo apresentam os resultados do experimento para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo.

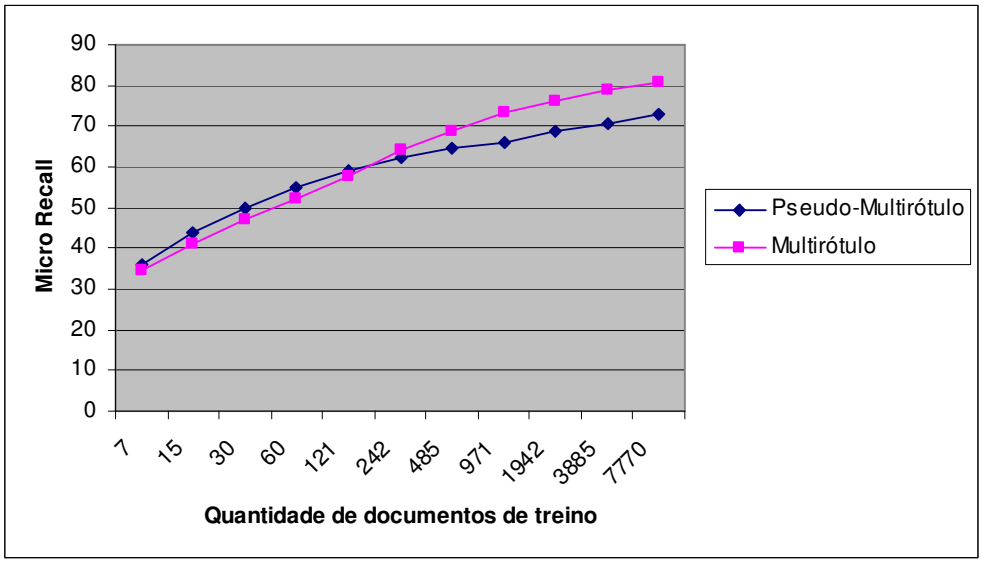

Figura 9 – Resultados Micro Recall para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(90).

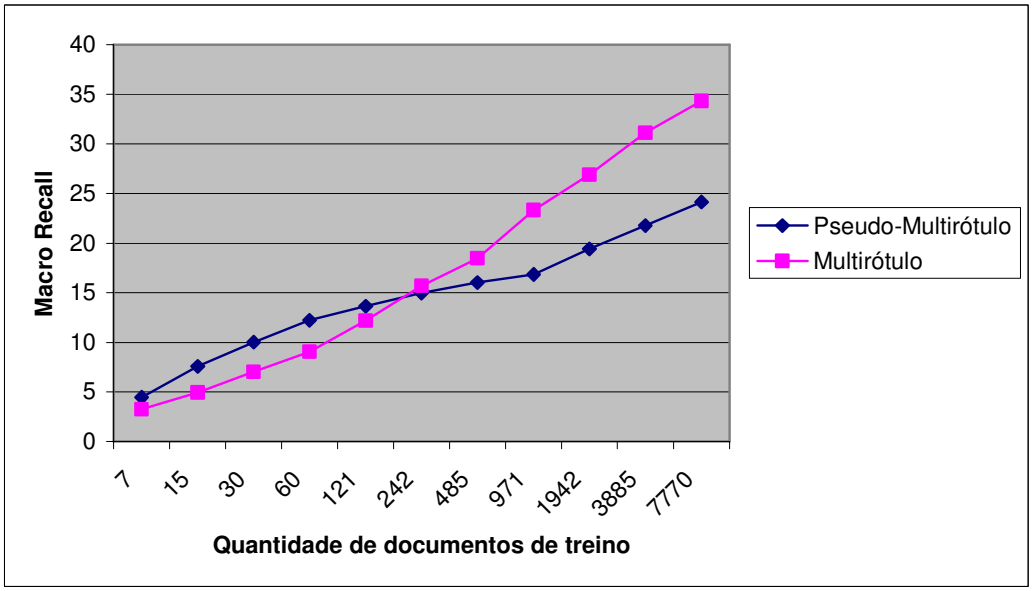

Figura 10 – Resultados Macro Recall para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(90).

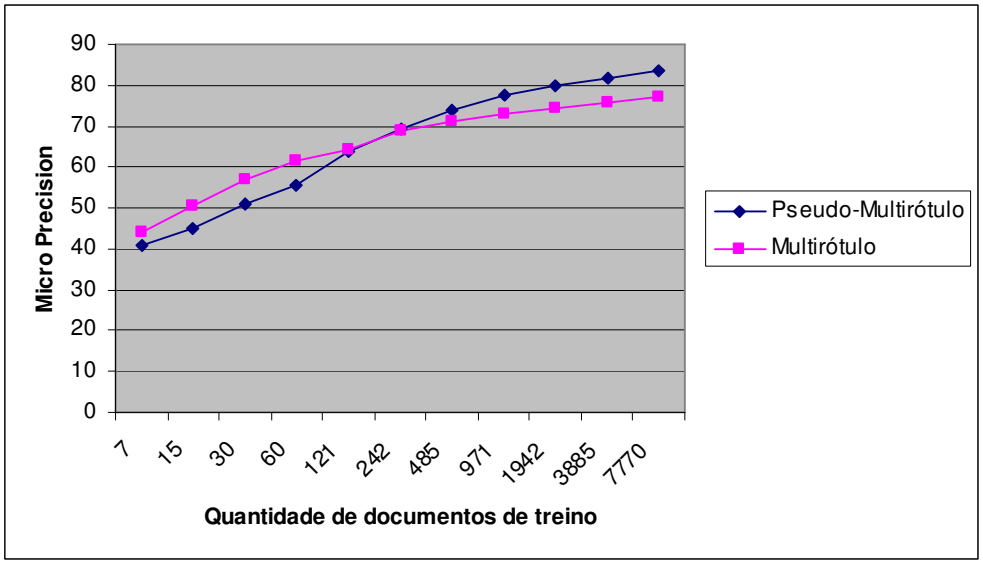

Figura 11 – Resultados Micro Precision para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(90).

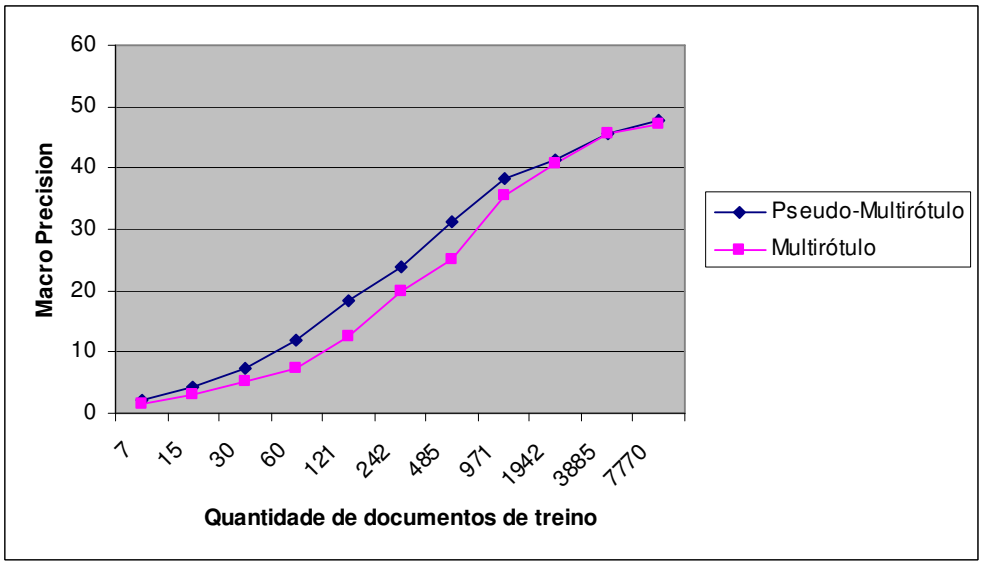

Figura 12 – Resultados Macro Precision para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(90).

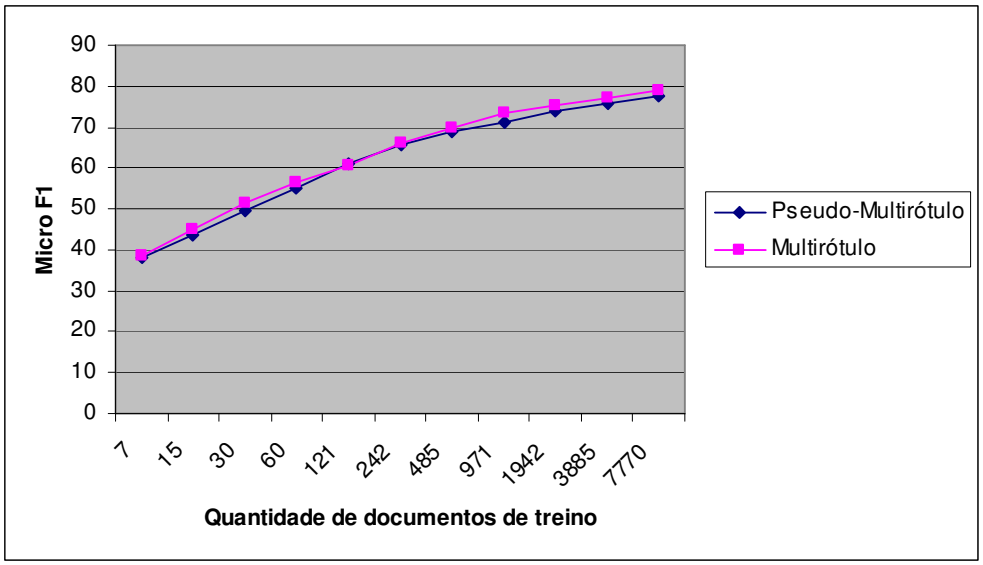

Figura 13 – Resultados Micro F1 para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(90).

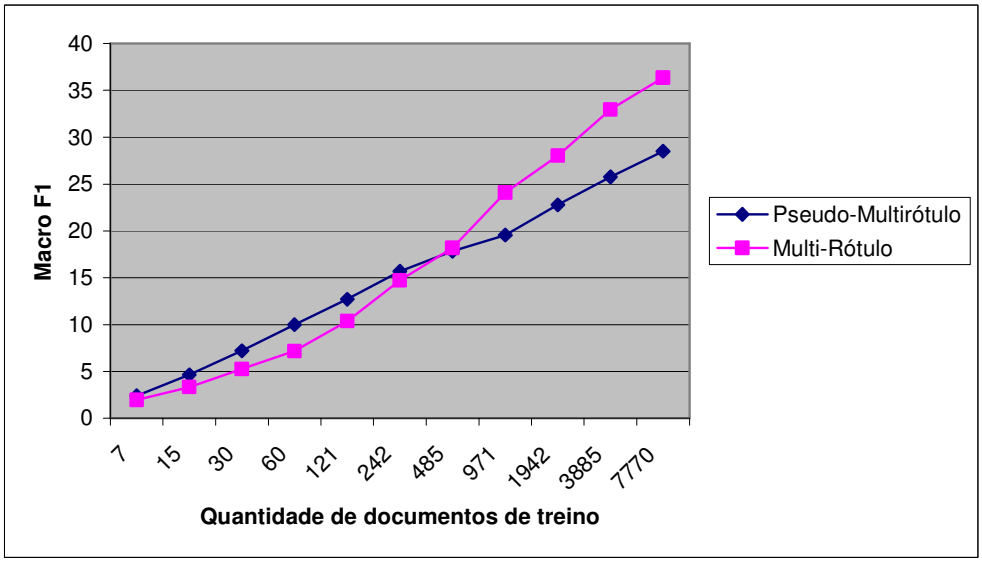

Figura 14 – Resultados Macro F1 para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base R(90).

4.3.3
Conclusões dos experimentos com as bases da Reuters

Com base nos experimentos realizados na base R(10) e base R(90) usando a partição fixa ModApté, pode-se observar que o algoritmo multirótulo mostrou-se mais eficiente, uma vez que possui valores maiores para Micro F1 e Macro F1.

Vale ressaltar também que o algoritmo multirótulo apresenta valores maiores para Micro Recall e Macro Recall, porém apresenta valores menores para Micro Precision e Macro Precision, o que sugere que o algoritmo multirótulo possui maior recall em detrimento de uma menor precision.

Com relação aos experimentos com partição aleatória, observa-se que conforme se aumenta a quantidade de documentos de treinamento, as medidas Micro Recall e Macro Recall dos algoritmos crescem até chegar a um ponto em que as medidas Micro Recall e Macro Recall do algoritmo multirótulo ultrapassam as medidas Micro Recall e Macro Recall do algoritmo pseudo-multirótulo.

4.3.4
Experimentos com a base Ohsumed

Nos experimentos foi utilizado um subconjunto da base Ohsumed definido pelos primeiros 20.000 documentos de 1991 que possuem abstracts, classificados em 23 categorias sobre doenças.

Primeiramente, foi realizado um experimento com uma partição fixa definida por 6.286 documentos de treinamento e 7.643 documentos de teste, um vocabulário de 27.871 palavras e 756 combinações das 23 categorias da base.

As tabelas 15, 16 e 17 apresentam os resultados do algoritmo pseudo-multirótulo. Já as tabelas 18, 19 e 20 apresentam os resultados do algoritmo mulirótulo.

Categoria	Recall	Precision	F1
Pathology	28,15	51,53	36,41
Cardiovasc	82,63	58,71	68,65
Immunolog	50,65	60,27	55,04
Neoplasms	84,46	62,23	71,66
Dig.Syst.	39,56	79,87	52,91

Tabela 15 – Resultados do algoritmo pseudo-multirótulo na base Ohsumed para 5 categorias.

Micro Recall	38,73
Micro Precision	62,53
Micro F1	47,83
Macro Recall	23,70
Macro Precision	71,13
Macro F1	28,48

Tabela 16 – Resultados globais do algoritmo pseudo-multirótulo na base Ohsumed.

Classe	Segundos
Treinamento	10,672
Classificação	727,516

Tabela 17 – Tempo de execução da fase de treinamento e da fase de classificação do algoritmo pseudo-multirótulo na base Ohsumed.

Categoria	Recall	Precision	F1
Pathology	59,96	43,81	50,63
Cardiovasc	79,63	68,07	73,40
Immunolog	55,83	56,07	55,95
Neoplasms	77,51	73,21	75,30
Dig.Syst.	65,03	55,09	59,65

Tabela 18 – Resultados do algoritmo multirótulo na base Ohsumed para 5 categorias.

Micro Recall	52,75
Micro Precision	60,04
Micro F1	56,16
Macro Recall	37,63
Macro Precision	65,40
Macro F1	43,27

Tabela 19 – Resultados globais do algoritmo multirótulo na base Ohsumed.

Classe	Segundos
Treinamento	40,281
Classificação	97,857

Tabela 20 – Tempo de execução da fase de treinamento e da fase de classificação do algoritmo multirótulo na base Ohsumed

Após o experimento utilizando a partição fixa, foi realizado um experimento para verificar como se comportam os dois algoritmos em função da quantidade de documentos de treinamento.

Para isso, para cada um dos algoritmos, foram gerados 10 partições aleatórias, onde 6.286 documentos eram de treinamento e 7.643 documentos eram de teste.

Para cada partição rodou-se o algoritmo 11 vezes, mantendo-se os documentos de teste fixos (7.643 documentos) e variando a quantidade de documentos de treinamento pertencentes ao conjunto de 6.286 documentos, iniciando com 6.286 e dividindo por 2 até a quantidade de documentos de treinamento chegar a 6.

Uma vez gerada a curva para cada uma dos 10 partições calculou-se a média das 10 curvas.

As figuras abaixo apresentam os resultados do experimento para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo.

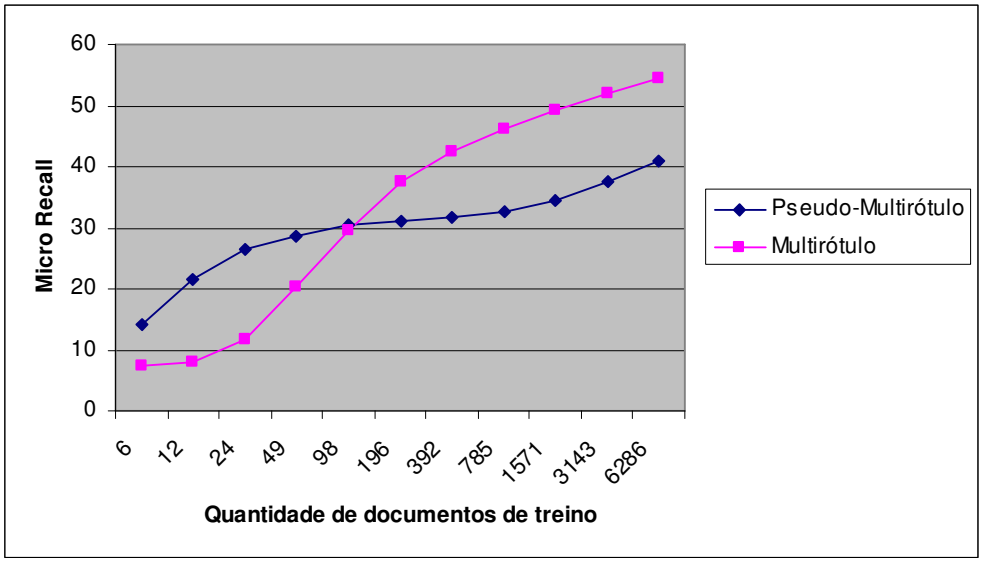

Figura 15 – Resultados Micro Recall para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base Ohsumed.

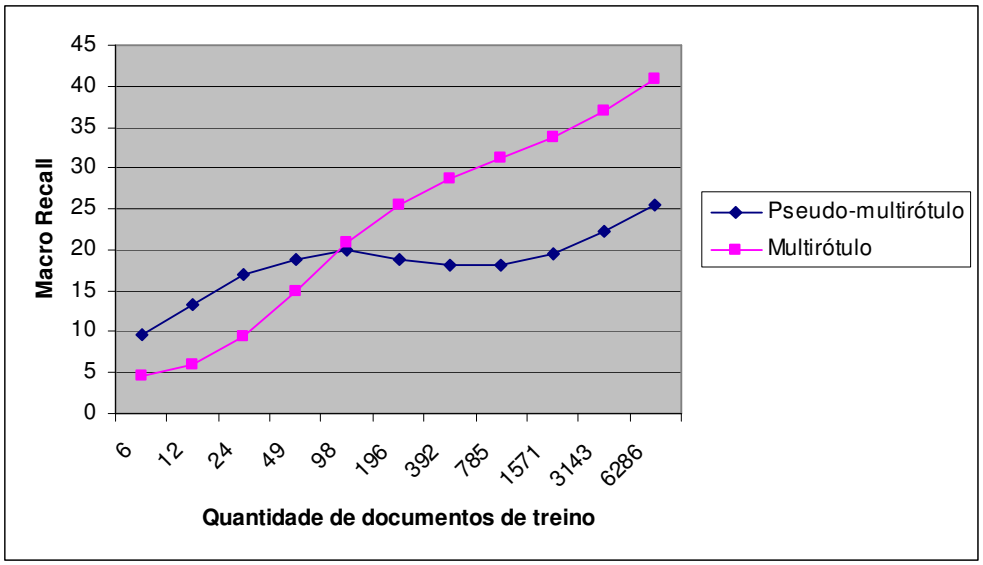

Figura 16 – Resultados Macro Recall para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base Ohsumed.

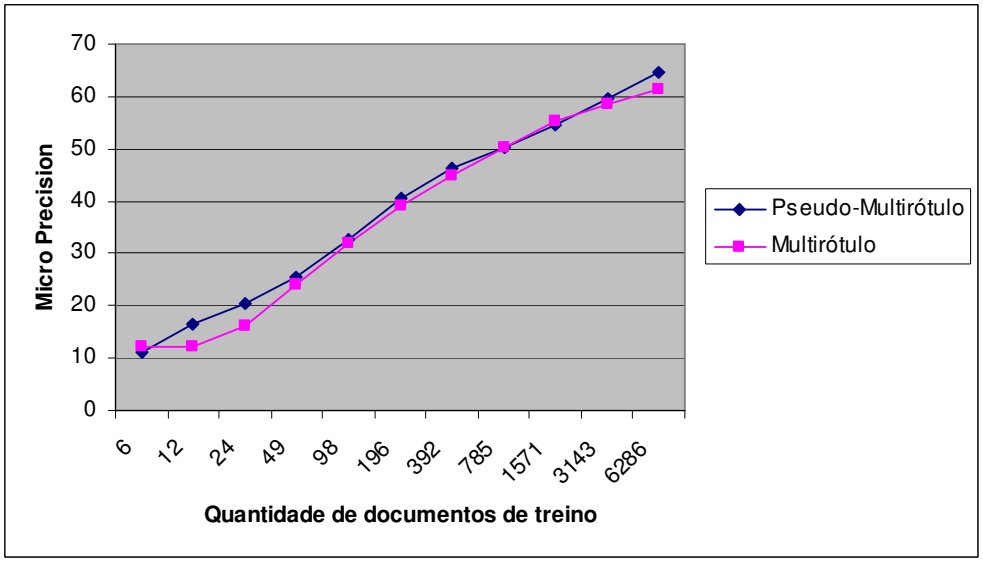

Figura 17 – Resultados Micro Precision para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base Ohsumed.

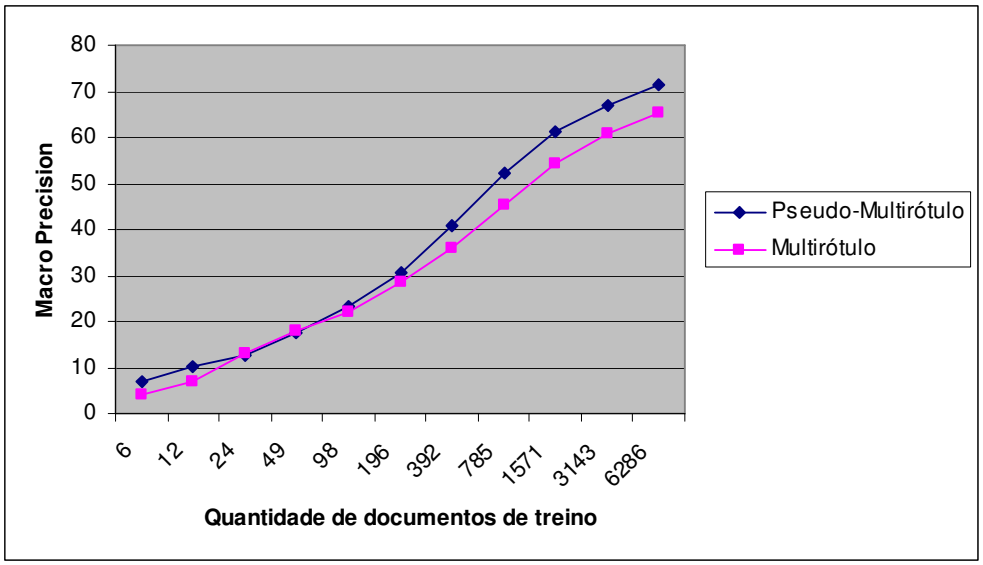

Figura 18 – Resultados Macro Precision para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base Ohsumed.

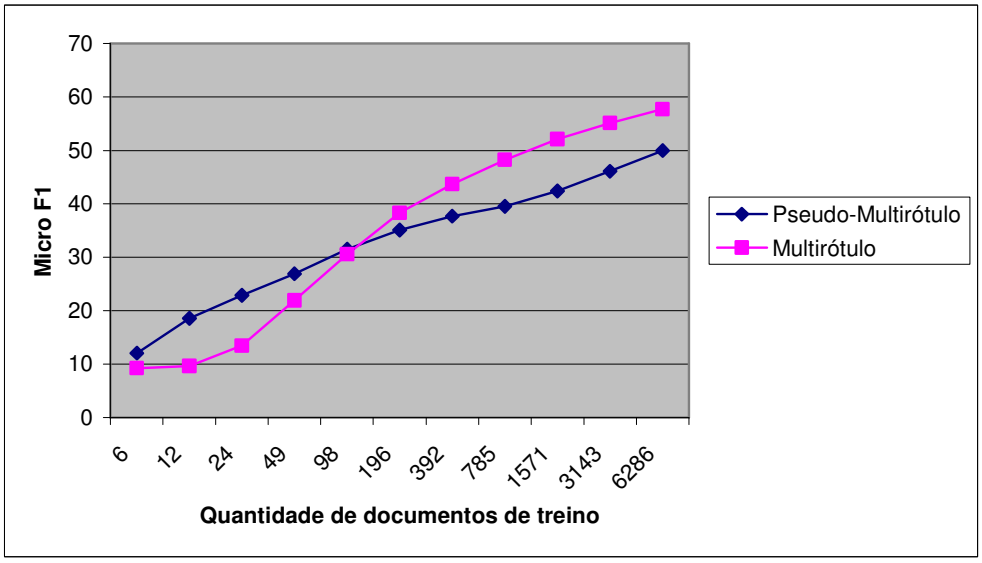

Figura 19 – Resultados Micro F1 para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base Ohsumed

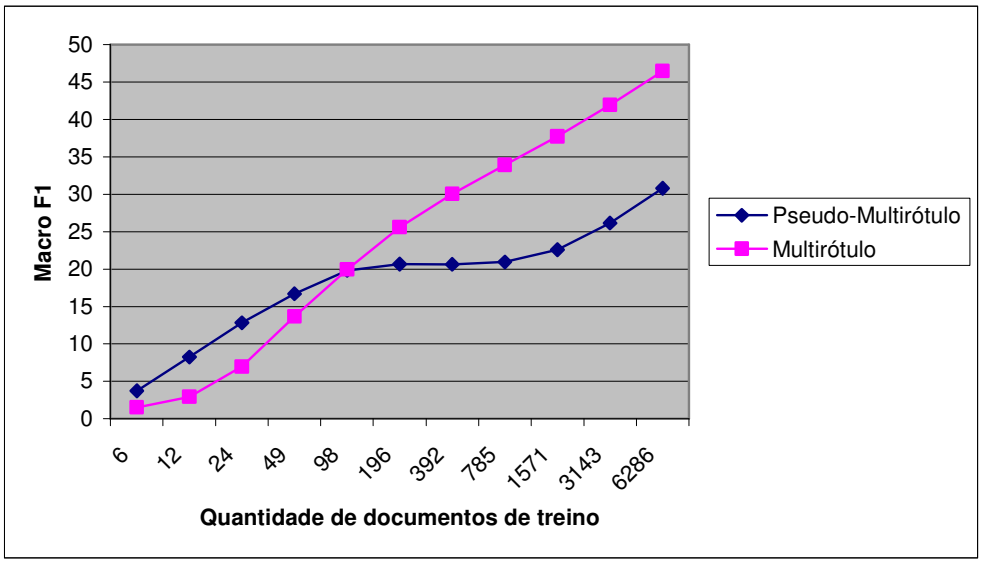

Figura 20 – Resultados Macro F1 para o algoritmo pseudo-multirótulo e para o algoritmo multirótulo na base Ohsumed

4.3.5
Conclusões dos experimentos com a base Ohsumed

Com base nos resultados da partição fixa, o algoritmo multirótulo obteve resultados melhores que o algoritmo pseudo-multirótulo, chegando a uma diferença de 51% entre os valores Macro F1, explicado por um aumento de 58% da medida Macro Recall.

Como na base da Reuters, o algoritmo multirótulo apresenta um aumento nos valores da medida recall e uma diminuição no valor da medida precision, porém o valor menor da medida precision não é suficiente para afetar o valor da medida F1, que em todos os experimentos se mostrou maior para o algoritmo multirótulo.

Nos experimentos com partições aleatórias verificou-se também que o algoritmo multirótulo obteve resultados piores que o algoritmo pseudo-multirótulo para poucos documentos de treinamento, porém, quando o número de documentos de treinamento chega a 98, os valores de Micro F1 e Macro F1 do algoritmo multirótulo ultrapassam os valores do algoritmo pseudo-multirótulo, chegando a uma diferença de 66%.

4.3.6
Comparação com outros trabalhos

Nesta seção, serão comparados os resultados obtidos pelo algoritmo multirótulo, que apresentou melhores resultados, com resultados de outros trabalhos na literatura de classificação de textos.

A grande dificuldade encontrada para realizar a comparação é a divergência de medidas utilizadas para avaliar o desempenho dos algoritmos propostos. Muitos trabalhos, por exemplo, apresentaram seus resultados através da medida "breakeven point", que só faz sentido em classificadores que usam limiares.

4.3.6.1.
Reuters

Bennett et al. [2002] propuseram um método de combinação de classificadores e testou esse método na base de dados Reuters R(10), utilizando quatro classificadores conhecidos na literatura: árvore de decisão, support vector machine, naive Bayes binário e naive Bayes multinomial. A seguir serão apresentados os resultados individuais dos classificadores, o resultado gerado pela combinação dos quatro classificadores (Strive-S) e por último os resultados dos algoritmos propostos neste trabalho:

Método	Macro F1
Árvore de decisão	78,46
Support vector machine	84,80
Naive Bayes binário	65,74
Naive Bayes multinomial	76,45
Strive-S (norm)	87,49
Naive Bayes pseudo-multirótulo	**84,74**
Naive Bayes multirótulo	**85,86**

Tabela 21 – Resultados de Bennett et al. [2002] na base R(10)

Já Sebastiani & Debole [2004] comparam a dificuldade de três subconjuntos da base de dados da Reuters, R(10), R(90) e R(115) comumente utilizados na literatura. Para realizar a comparação foram realizados testes através dos classificadores Rochhio, *K*-NN (K-nearest neighbor) e Support vector machine. A média dos resultados obtidos pelos três classificadores para a base R(10) e R(90) são apresentados na tabela 22 e tabela 23, assim como os resultados obtidos neste trabalho:

Método	Micro F1	Macro F1
Rochhio, *K*-NN e Support vector machine	85,223540	72,393364
Naive Bayes pseudo-multirótulo	**93,05**	**84,74**
Naive Bayes multirótulo	**93,17**	**85,86**

Tabela 22 – Resultados de Sebastiani & Debole [2004] na base de dados R(10)

Método	Micro F1	Macro F1
Rochhio, *K*-NN e Support vector machine	78,707075	52,659655
Naive Bayes pseudo-multirótulo	**77,50**	**21,92**
Naive Bayes multirótulo	**78,68**	**30,73**

Tabela 23 – Resultados de Sebastiani & Debole [2004] na base de dados R(90)

No trabalho de Yang & Liu [1999] é comparado o desempenho de cinco classificadores de texto na base de dados R(90): SVM (Support vector machine), *K*-NN (K-nearest neighbor), LLSF (Linear Least Squares Fit), NB (Naive bayes baseado em um modelo misto multinomial, proposto por McCallum [1999]) e NNet (Rede neural). Os resultados desse experimento serão apresentados na tabela 24, assim como os resultados dos algoritmos naive Bayes pseudo-multirótulo (NBPM) e naive Bayes multirótulo (NBM) propostos neste trabalho.

Método	Micro Recall	Micro Precision	Micro F1	Macro F1
SVM	81,20	91,37	85,99	52,51
KNN	83,39	88,07	85,67	52,42
LLSF	85,07	84,89	84,98	50,08
NNet	78,42	87,85	82,87	37,65
NB	76,88	82,45	79,56	38,86
NBPM	**72,58**	**83,14**	**77,50**	**21,92**
NBM	**79,68**	**77,70**	**78,68**	**30,73**

Tabela 24 – Resultados de Yang & Liu [1999] na base de dados R(90)

4.3.6.2. Ohsumed

Alessandro Moschitti [2003b] realizou um estudo relativo à aplicação de técnicas de processamento de linguagem natural na classificação automática de textos. Desta forma, informações sintáticas, como nomes próprios e substantivos compostos, foram contempladas na representação dos documentos.

Em um dos seus experimentos, Alessandro testou os algoritmos Parametrized Rochhio (proposto pelo autor) e Support Vector Machine, na base de documentos médicos Ohsumed, utilizando a representação básica "bag-of-words" sem nenhuma informação sintática. Os resultados desse experimento serão apresentados na tabela 25, assim como os resultados dos algoritmos propostos neste trabalho.

Outros experimentos foram realizados incluindo informação sintática na representação dos documentos, porém não são comparáveis com este trabalho, cujos experimentos se basearam apenas na representação simples "bag of words".

	PRC	SVM	**NBPM**	**NBM**
Categoria	F1	F1	**F1**	**F1**
Pathology	50.58	48.5	**36,41**	**50,63**
Cardiovasc	77.82	80.7	**68,65**	**73,40**
Immunolog	73.92	72.8	**55,04**	**55,95**
Neoplasms	79.71	80.1	**71,66**	**75,30**
Dig.Syst.	71.49	71.1	**52,91**	**59,65**

Tabela 25 – Resultados de Moschitti [2003b] em 5 categorias da base de dados Ohsumed

Método	Micro F1
PRC	65,80
SVM	68,37
NBPM	**47,83**
NBM	**56,16**

Tabela 26 – Resultados globais de Moschitti [2003b] na base de dados Ohsumed

5
Conclusão

Nesta dissertação foram propostos dois algoritmos de classificação automática de textos multirótulo.

Além disso, foi proposta a utilização de tais algoritmos em um ambiente completo de mineração de textos, onde inicialmente o usuário especifica um conjunto de categorias de interesse e um conjunto de fontes de notícias da internet. O sistema, então, retorna ao usuário um conjunto de notícias, possibilitando ao usuário selecionar para cada notícia um conjunto de categorias associadas. A partir do treinamento do usuário, um novo conjunto de notícias é retornado, onde cada notícia está associada a um conjunto de categorias. Então, o usuário pode corrigir os possíveis erros cometidos pelo classificador.

Com o propósito de apresentar o desempenho de tais algoritmos, realizaram-se experimentos em duas bases de documentos freqüentemente utilizadas na literatura, a base de notícias Reuters 21578 e a base de documentos médicos Ohsumed.

A fim de que os resultados obtidos fossem comparáveis com outros trabalhos, foram realizados experimentos utilizando partições pré-definidas, no caso da Reuters, a partição ModApté e no caso da base Ohsumed, a partição dos 20.000 primeiros documentos.

Comparando-se os resultados obtidos com outros trabalhos, pode-se concluir que os dois algoritmos possuem uma eficiência satisfatória com relação às medidas Micro Recall, Micro Precision e Micro F1.

Porém, nos experimentos realizados na base da Reuters R(90), os dois algoritmos apresentaram a medida Macro F1 consideravelmente inferior aos trabalhos anteriores. Uma vez que a base de notícias Reuters R(90) possui uma distribuição não uniforme entre os documentos de treinamento e o conjunto de categorias, pode-se concluir que tal desempenho se deve ao fato de tais algoritmos serem muito sensíveis à distribuição entre documentos e categorias. Isso pode ser uma desvantagem, uma vez que tais algoritmos aplicados ao sistema de mineração

de textos proposto necessitam de uma maior intervenção do usuário, já que deve ser apresentada uma maior quantidade de exemplos de treinamento para cada categoria.

Outra desvantagem é que as complexidades teóricas da fase de classificação do algoritmo pseudo-multirótulo e da fase de treinamento do algoritmo multirótulo são exponenciais na quantidade de categorias. Entretanto, considerando que, na prática, os documentos são associados a combinações de categorias relativamente pequenas, isso não é um problema.

As vantagens de se utilizar tais algoritmos propostos são:

- Não é necessária a manutenção dos documentos de treinamento. Uma vez realizado o treinamento, os documentos podem ser descartados.

- Os algoritmos são perfeitamente adaptáveis ao contexto de classificação on-line, uma vez que dado um novo documento de treinamento, não é necessário retreinar toda a base para contemplar o novo conhecimento.

- Os dois algoritmos não utilizam limiares, que são muito específicos para o conjunto de treinamento utilizado, não possuindo grande capacidade de generalização na aprendizagem. Desta forma, não é necessária uma fase de validação para se encontrar valores ideais para tais parâmetros.

- Os dois algoritmos consideram que existe dependência entre categorias, que, na prática, é o caso mais comum.

5.1 Contribuições

As principais contribuições deste trabalho são:

- Criação de dois algoritmos de classificação automática de textos de fácil implementação, que possibilitam resolver o problema mais genérico de classificação (multirótulo), sem a necessidade de calcular limiares e considerando que existe dependência entre as categorias.

- Proposta de criação de um sistema de mineração de textos aplicado ao contexto de notícias de jornal, com a possibilidade de realimentação de relevância por parte do usuário e treinamento on-line.

- Possibilidade de aplicação dos algoritmos propostos no sistema de mineração de textos.

5.2 Trabalhos futuros

- Implementação do sistema de mineração de textos e realização de testes para verificar a eficiência de uma aplicação prática dos algoritmos propostos neste trabalho.

- Inclusão de relações entre categorias, ou seja, permitir que o usuário apresente ao sistema relações entre categorias e o sistema agregue tais informações ao aprendizado.

- Possibilitar que o aprendizado seja semi-supervisionado, reduzindo a necessidade de o usuário associar para cada documento um conjunto de categorias, tarefa que pode ser bastante trabalhosa e suscetível a erros.

- Realização de melhorias no algoritmo multirótulo, verificando novas condições de parada na fase de classificação.

- Realização de testes com outras técnicas de amortização além da técnica utilizada neste trabalho.

6
Referências bibliográficas

BAEZA-YATES, R.; HURTADO, C.; MENDOZA, M. **Query Recommendation using Query Logs in Search Engines**. Center of Web Reseach, Dept. of Computer Science, Universidad de Chile, 2004.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrieval**. Association for Computing (ACM) and Addison Wesley Longman Limited, USA, 1999.

BAKER, D.; MCCALLUM, A. **Distributional clustering of words for text classification**. In W. Bruce Croft, Alistair Moffat, Cornelis J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, pages 96–103, Melbourne, AU, 1998. ACM Press, New York, US.

BASILI, R.; MOSCHITTI, A.; PAZIENZA, M. T. **Language sensitive text classification**. In Proceedings of 6th RIAO Conference (RIAO 2000), Content-Based Multimedia Information Access, Collge de France, Paris, France, 2000.

BENNETT, P. N.; DUMAIS, S. T.; HORVITZ, E. **Probabilistic combination of text classifiers using reliability indicators: models and results**. Proceedings of the 25th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR02), pp. 207–214, 2002.

BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. **Classification and Regression Trees**. Wadsworth Int, 1984.

CAROPRESO, M. F.; MATWIN, S.; SEBASTIANI, F. **A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization**. In Amita G. Chin, editor, Text Databases and Document Management: Theory and Practice, pages 78–102. Idea Group Publishing, Hershey, US, 2001.

CHAI, K. M.; NG, H. T.; CHIEU, H. L. **Bayesian online classifiers for text classification and filtering**. In Micheline Beaulieu, Ricardo Baeza-Yates, Sung Hyon Myaeng, and Kalervo Järvelin, editors, Proceedings of SIGIR-02, 25th ACM International Conference on Research and Development in Information Retrieval, pages 97–104, Tampere, FI, 2002. ACM Press, New York, US.

DOMINGOS, P.; PAZZANI, M. **On the optimality of the simple Bayesian classifier under zero-one loss**. Machine Learning, 29(2/3), 103, 1997.

DUMAIS, S.; PLATT, J.; HECKERMAN, D.; SAHAMI, M. **Inductive learning algorithms and representations for text categorization**. In Georges Gardarin, James C. French, Niki Pissinou, Kia Makki, and Luc Bouganim, editors, Proceedings of CIKM-98, 7th ACM InternationalConference on Information and Knowledge Management, pages 148–155, Bethesda, US, 1998. ACM Press, New York, US.

GALAVOTTI, L.; SEBASTIANI, F.; SIMI, M. **Experiments on the use of feature selection and negative evidence in automated text categorization**. In José L. Borbinha and Thomas Baker, editors, Proceedings of ECDL-00, 4th European Conference on Research and Advanced Technology for Digital Libraries, pages 59–68, Lisbon, PT, 2000. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1923.

GILES, C. L.; HAN, H.; MANAVOGLU, E.; ZHA, H.; ZHANG, Z.; FOX, E. A. **Automatic document metadata extraction using support vector machines**. In JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries, pages 37–48, Washington, DC, USA, IEEE Computer Society, 2003.

GRAHAM, P. **A plan for Spam**. Disponível em <http://www.paulgraham.com/spam.html>. Acesso em 22 jul. 2006.

HARMAN, D. **How effective is suffixing?** Journal of the Amercian Society of Information Science 42(1), 7 – 16, 1991.

HAYES, P. J.; WEINSTEIN, S. P. **Construe/Tis: a system for content-based indexing of a database of news stories**. In Alain Rappaport and Reid Smith, editors, Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence, pages 49–66. AAAI Press Menlo Park, US, 1990.

JOACHIMS, T. **Learning to classify text using support vector machines**. Kluwer Academic Publishers, 2002.

JOACHIMS, T. **Text categorization with support vector machines: learning with many relevant features**. In Machine Learning: ECML98, Tenth European Conference on Machine Learning, pages 137–142, 1998.

KOLLER, D.; SAHAMI, M. **Hierarchically classifying documents using very few words**. Proc. of the 14th International Conference on Machine Learning ICML97, pp. 170–178, 1997.

LAM, W.; LAI, K. **A meta-learning approach for text categorization**. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval, pages 303–309, New Orleans, US, 2001. ACM Press, New York, US.

LEWIS, D. **An evaluation of phrasal and clustered representations on a text categorization task**. In Proceedings of SIGIR-92, 15thACM International Conference on Research and Development in Information Retrieval, pages 37–50, Kobenhavn, DK, 1992.

LEWIS, D. **Naive Bayes at forty: The independence assumption in information retrieval**. Conference proceedings of European Conference on Machine Learning (pp. 4–15), 1998.

LEWIS, D.; RINGUETTE, M. **A comparison of two learning algorithms for text categorization**. The Third Annual Symposium on Document Analysis and Information Retrieval, pp. 81–93, 1994.

LOVINS, J. B. **Development of a stemming algorithm**. Mechanical Translation and Computacional Linguistics 11 (1–2), 22–31, 1968.

MARON, M. **Automatic indexing: an experimental inquiry**. Journal of the Association for Computing Machinery, 8(3):404–417, 1961.

MCCALLUM, A. K. **Multi-label text classification with a mixture model trained by EM**. In AAAI 99 Workshop on Text Learning, 1999.

MCCALLUM, A. K.; NIGAM, K. **A comparison of event models for naive Bayes text classification**. In Proceedings of the 1st AAAI Workshopon Learning for Text Categorization, pages 41–48, Madison, US, 1998.

MCCALLUM, A. K.; ROSENFELD, R.; MITCHELL, T., NG, A. **Improving text classification by shrinkage in a hierarchy of classes**. Machine Learning: Proceedings of the Fifteenth International Conference, pp. 359–367, 1998.

MITCHELL, T. **Machine Learning**. Boston, USA: McGraw-Hill, 1997.

MLADENIĆ, D.; GROBELNIK, M. **Word sequences as features in text-learning**. In Proceedings of ERK-98, the Seventh Electrotechnical and Computer Science Conference, pages 145–148, Ljubljana, SL, 1998.

MOSCHITTI, A. **A study on optimal parameter tuning for Rocchio text classifier**. In proceedings of the 25th European Conference on Information Retrieval Research (ECIR), Pisa, Italy, 2003a.

MOSCHITTI, A. **Natural Language Processing and Text Categorization: a study on the reciprocal benefical interactions**. PhD thesis, University of Rome Tor Vergata, Rome, Italy, 2003b.

MOSTELLER, F.; WALLACE, D. **Inference and Disputed Authorship: The Federalist**. Addison-Wesley, Reading, Massachusetts, 1964.

NIGAM, K.; MCCALLUM, A.; THRUN, S.; MITCHELL, T. **Text classification from labeled and unlabeled documents using EM**. Machine Learning, 39(2/3): 103–134, 2000.

PORTER, M. **An algorithm for suffixing stripping**. Program 14(3), 130–137, 1980.

ROBERTSON, S. E.; SPARCK JONES, K. **Relevance weighting of search terms**. Journal of the American Society for Information Science, 27(3), 129–146, 1976.

SALTON, G. **The SMART Retrieval System**. Experiment in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

SCHAPIRE, R.; SINGER, Y. **BOOSTEXTER: A boosting-based system for text categorization**. Machine Learning, Vol.39, No. 2/3, 2000.

SCOTT, S.; MATWIN, S. **Feature engineering for text classification**. In Ivan Bratko and Saso Dzeroski, editors, Proceedings of ICML-99, 16th International Conference on Machine Learning, pages 379–388, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

SEBASTIANI F.; SPERDUTI A.; VALDAMBRINI, N. **An improved boosting algorithm and its application to automated text categorization**. In Arvin Agah, Jamie Callan, and Elke Rundensteiner, editors, Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management, pages 78–85, McLean, US, 2000. ACMPress, New York, US.

SEBASTIANI, F. **Machine learning in automated text categorization**, Tech. Rep. IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Pisa, Italy, 1999.

SEBASTIANI, F.; DEBOLE, F. **An analysis of the relative hardness of Reuters-21578 subsets**. In Proceedings of LREC-04, 4th International Conference on Language Resources and Evaluation, pages 971–974, Lisbon, PT, 2004.

SHAVLIK, J.; ELIASSI-RAD, T. **Intelligent agents for web-based tasks: An advice-taking approach**. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence: Workshop on Learning for Text Categorization. Madison, WI, pp. 63–70, 1998.

TONG, S.; KOLLER, D. **Support vector machine active learning with applications to text classification**. Journal of Machine Learning Research, 2:45–66, 2001.

VAPNIK, V.N. **Statistical learning theory**. John Wiley & Sons, Inc., N.Y., 1998.

VILAR, D.; JOSÉ CASTRO, M.; SANCHIS, E. **Multi-label text classification using multinomial models**. Computer Science Department, Aachen University (Germany), Departament de Sistemes Informàtics i Computació, Universitat Politècnica de València (Spain), 2004.

YANG, Y.; LIU, X. **A re-examination of text categorization methods**. In Marti A. Hearst, Fredric Gey, and Richard Tong, editors, Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, pages 42–49, Berkeley, US, 1999. ACM Press, New York, US.