SISTEMAS DE INFORMAÇÃO
Revista de Sistemas de Informacao da FSMA n. 5 (2010) pp. 42-62 http://www.fsma.edu.br/si/sistemas.html

A Tarefa de Classificação em *Text Mining*

Eduardo Bezerra e Ronaldo Goldschmidt

***Resumo*—A tarefa de classificação consiste em criar um mapeamento de cada objeto de uma coleção (*dataset*) em um conjunto de categorias (ou classes). Esse mapeamento é também chamado de *modelo de classificação* ou *classificador*. No contexto de dados textuais os objetos usados na formação do classificador podem ser documentos da coleção, ou mesmo frases ou palavras que ocorrem naqueles documentos. Este tutorial fornece uma introdução à tarefa de classificação de documentos, um dos problemas mais conhecidos em *Text Mining*. São apresentados alguns algoritmos populares na literatura para criação de modelos de classificação. Além disso, também são algumas descritas técnicas cujo objetivo é permitir avaliar a qualidade de um modelo de classificação.**

***Index Terms*—text mining, classificação, aprendizado indutivo.**

I. Introdução

OS constantes avanços na área da Tecnologia da Informação têm viabilizado o armazenamento de grandes volumes dados. Tecnologias como a Internet, Sistemas Gerenciadores de Banco de Dados, leitores de códigos de barras, dispositivos de memória secundária de maior capacidade de armazenamento e de menor custo e Sistemas de Informação em geral são alguns exemplos de recursos que têm viabilizado a proliferação e o crescimento de inúmeras bases de dados de natureza comercial, administrativa, governamental e científica. Segundo estimativas, o mundo produz atualmente de um a dois exabytes de informação por ano, o que equivale a cerca de 250 megabytes ou 250 livros por ser humano.

A humanidade vive um cenário de grande sobrecarga de informação no qual mecanismos de busca de informação pela web tais como Google™, Altavista™, Yahoo™, entre outros, ao invés de reduzir o problema, acabam por amplificá-lo, uma vez que tornam novos documentos rapidamente disponíveis. Por exemplo, a Google possui 4,2 bilhões de documentos em seus índices e realiza diariamente cerca de 150 milhões de consultas, o que equivale a aproximadamente 2000 consultas por segundo. Uma parcela significativa da informação existente em formato digital é relativa a dados textuais. Estima-se que cerca de 80% dos dados existentes em empresas esteja nesse formato, abrangendo documentos em intranets, páginas da Web, mensagens de e-mail, biblioteca digitais, newsgroups, blogs, dentre outros.

A busca por informação a partir de grandes quantidades de documentos é inviável sem o auxílio de ferramentas computacionais apropriadas. Portanto, torna-se imprescindível o desenvolvimento de ferramentas que auxiliem o ser humano, de forma automática e inteligente, na análise e na realização de tarefas como extração de padrões, tendências, associações a partir de coleções de dados textuais.

Para atender a essas novas demandas, surgiu uma nova área de estudo denominada Mineração de Textos (**Text Mining**), que vem atraindo interesse junto às comunidades científica e industrial. *Text Mining* procura abstrair padrões, relações e regras (conhecimentos) a partir de dados textuais. Dados textuais podem se apresentar em linguagem natural (inglês, português), ou em formato semi-estruturado (por exemplo, documentos em XML, documentos de email, etc.).

A. Complicadores no processamento de dados textuais

Há diversos complicadores na análise de dados textuais, entre eles:

- A falta de estrutura dos textos a serem analisados. Diferentes estilos de escrita podem ter sido utilizados.
- A ambigüidade existente na estrutura e no significado dos textos. A natureza ambígua da linguagem natural é um grande desafio. Confusões semânticas tais como sinonímia e polissemia tornam ainda mais difícil a tarefa de compreensão da linguagem natural. A sinonímia compreende as muitas formas de representar o mesmo conceito. Por exemplo: os termos automóvel e carro referem-se ao mesmo conceito. A polissemia, por outro lado, envolve termos iguais com significados diferentes em diferentes contextos. Exemplo: A palavra manga possui significados diferentes nos contextos de fruticultura e de vestuário.

B. Tipos de processamento de dados textuais

Há três tipos de processamento voltado à Análise de Conteúdo, com níveis de sofisticação sucessivos:

- Processamento Léxico e Sintático - Envolve o reconhecimento de tokens (termos), a normalização de termos e a construção da linguagem.
- Processamento Semântico - Envolve a extração do significado inerente aos textos. Requer a extração de entidades nomeadas tais como nomes de pessoas, nomes de organizações, locais, etc...
- Processamento de Características Extra-Semânticas - Mais complexo, envolve a identificação de sentimentos nos textos analisados. Por exemplo: sarcasmo, melancolia, alegria, etc.

II. Tarefas de *Text Mining*

A seguir encontram-se relacionadas e resumidas algumas das principais principais tarefas de *Text Mining*. Estas tarefas

Eduardo Bezerra é professor do CEFET-RJ (Centro Federal de Educação Tecnológica Celso Suckow da Fonseca).
Ronaldo Goldschmidt é professor do IST-FAETEC (Instituto Superior de Tecnologia em Ciências da Computação do Rio de Janeiro - Fundação de Apoio à Escola Técnica).

estão associadas ao tipo de conhecimento a ser abstraído a partir dos dados analisados.

Descoberta de Associações: Abrange a busca por termos que freqüentemente ocorram de forma simultânea em documentos textuais. Termos simultâneos e freqüentes podem auxiliar na remoção de ambigüidades e na caracterização de contextos. Um exemplo clássico e didático da aplicação desta tarefa é na área de atendimento ao cliente: após o lançamento de um novo produto no mercado, diversos clientes utilizaram o site da empresa para relatar seu nível de satisfação com tal produto. Palavras freqüentes e simultâneas podem auxiliar na identificação de pontos fortes e fracos do produto, levando a novas ações de marketing ou até mesmo de reformulação da produção. Algoritmos tais como o Apriori, GSP, DHP, entre outros, são exemplos de ferramentas que implementam a tarefa de descoberta de associações [1].

Agrupamento: Utilizada para separar os documentos de uma base de textos em subconjuntos ou clusters, de tal forma que os documentos de um cluster compartilhem de propriedades comuns que os distingam de documentos em outros clusters. O objetivo nesta tarefa é maximizar similaridade intracluster e minimizar similaridade intercluster. Diferente da tarefa de classificação, que tem rótulos pré-definidos, a clusterização precisa automaticamente identificar os grupos de documentos aos quais o usuário deverá atribuir rótulos ([2]). Por exemplo: uma escola pode realizar um processo de clusterização de sua base de documentos de forma obter grupos de documentos que compartilhem o mesmo perfil de conteúdo. Na implementação desta tarefa podem ser utilizados algoritmos tais como: K-Means, K-Modes, K-Prototypes, K-Medoids, Kohonen, dentre outros.

Sumarização: Esta tarefa, muito comum em KDT, consiste em procurar identificar e indicar características comuns entre conjuntos de textos [3]. Como exemplo considere um banco de textos como o mencionado no exemplo da tarefa anterior. Após a clusterização, uma prática usual é utilizar um algoritmo de sumarização de textos que permita descrever de forma resumida o conteúdo dos documentos em cada cluster. Tal informação poderia ser utilizada pela equipe da escola para organizar a chegada de novos textos. Lógica Indutiva, Algoritmos Genéticos, Otimização por Nuvem de Partículas são alguns exemplos de tecnologias que podem ser aplicadas na implementação da tarefa de sumarização.

Detecção de Desvios: Esta tarefa consiste em procurar identificar documentos do banco de textos cujas características sejam divergentes de um conjunto de documentos textuais [3]. Tais documentos são denominados "outliers". A Estatística fornece recursos para a implementação desta tarefa.

Classificação: Consiste em descobrir uma função que mapeie um conjunto de textos em um conjunto de rótulos categóricos pré-definidos, denominados classes. Uma vez descoberta, a função pode ser aplicada a novos textos de forma a prever a classe em que tais textos se enquadram. Como exemplo da tarefa de classificação, considere uma empresa que deseja separar notícias em função do segmento ao qual pertençam (esportes, política, religião, etc...). Uma aplicação da tarefa de classificação consiste em descobrir uma função que mapeie corretamente os textos, a partir de seu conteúdo, em uma destas classes. Tal função, uma vez descoberta, pode ser utilizada para prever a alocação de novos textos recebidos pela empresa. Esta função pode ser incorporada a um sistema de apoio à decisão que auxilie na filtragem e catalogação de documentos textuais recebidos. Um outro exemplo se refere à classificação de e-mails em *spam* e *não spam*. De forma análoga ao exemplo anterior, um mecanismo de classificação automática de e-mails pode auxiliar na filtragem de e-mails indesejados. Redes Neurais, Algoritmos Genéticos, Lógica Indutiva são exemplos de tecnologias que podem ser aplicadas na tarefa de classificação ([4]).

III. Representação de documentos

Há diversas técnicas de pré-processamento que permitem a determinação de uma lista de *termos* $\mathcal{T}$, isto é, do conjunto de unidades que compõem o vocabulário da coleção. Por outro lado, o processamento computacional em algoritmos de *Text Mining* requer muitas vezes a representação dos documentos em um formato adequado. Portanto, uma atividade que deve realizada durante o pré-processamento de uma coleção é a escolha da *forma de utilizar* os termos do vocabulário para representar os documentos da coleção. Em outras palavras, devemos decidir que *representação* deve ser utilizada para estruturar essa coleção de documentos. Para resolver este problema, são usadas diversas técnicas para adicionar uma *dimensão numérica* aos documentos. O objetivo desta seção é justamente apresentar uma visão geral dessas técnicas.

A. O modelo de espaço Vetorial

Entre os principais modelos de representação de documentos textuais, podemos citar o modelo probabilístico [5], os modelos booleanos clássico e estendido [6] e o **modelo de espaço vetorial** (*vector-space model*, *term vector model*, VSM) [7].

O modelo probabilístico, como o próprio nome deixa transparecer, interpreta cada documento como um evento em um *espaço amostral*. O modelo booleano é baseado na Teoria dos Conjuntos e interpreta cada documento como um conjunto de termos. Por fim, o modelo de espaço vetorial interpreta cada documento utilizando conceitos e técnicas da Álgebra Linear e da Geometria Espacial. O VSM é a forma de representação e modelagem de documentos mais frequentemente utilizada no processamento de dados textuais. Nesse modelo, documentos são interpretados como objetos geométricos, mais especificamente como vetores em um espaço m-dimensional. Por conta de o VSM ser o mais utilizado na prática, o restante dessa seção descreve este modelo em maiores detalhes.

Em 1975, Gerard Salton propôs um modelo matemático para representar documentos, o VSM [7]. Curiosamente, Salton originalmente propôs o VSM para uso em um sistema de recuperação de informações (SRI), o `SMART` [8]. Em um SRI que utiliza essa representação, as próprias consultas que o sistema recebe, e que representam necessidades de informação de seus usuários, são também representadas como vetores. Desta forma, consultas e documentos podem ser manipulados indistintamente e de forma integrada. De fato, a elegância e simplicidade do VSM residem no fato de que, dado um

conjunto de documentos $\mathcal{D}$ e uma consulta q, para avaliar a relevância de cada documento em $\mathcal{D}$ relativamente a q, podemos usar técnicas simples da Álgebra Linear.

Embora a utilização original do VSM tenha sido em sistema de recuperação de informações, conforme novas técnicas de análise de dados textuais foram sendo propostas, também nesses casos, percebeu-se a utilidade e a adequabilidade do VSM como modelo de representação de uma coleção de documentos. De fato, conforme descrevemos em capítulos seguintes deste livro, diversas técnicas de MDT se baseiam nesse modelo de representação para documentos.

1) Matriz termo-documento: No VSM, cada documento da coleção é representado como uma lista ordenada de valores numéricos, isto é, como um *vetor*. A cada componente dessa lista, está associado um, e apenas um, termo do vocabulário. Definimos a **dimensionalidade** de uma coleção de documentos como a cardinalidade do conjunto de termos $\mathcal{T}$ do vocabulário, que denotamos por m. No VSM, essa é mesma *dimensionalidade* do espaço vetorial em que os documentos (interpretados como vetores) são representados. Mais formalmente, O VSM representa cada documento da coleção como um vetor cuja forma é dada pela Equação 1.

$$\vec{d_j} = (w_{1j}, w_{2j}, w_{3j}, \ldots, w_{mj}) \tag{1}$$

Cada componente do vetor $\vec{d_j}$ é um valor numérico definido sobre um dos eixos (coordenadas) no espaço vetorial e corresponde a um termo da coleção. Esse valor numérico está associado ou à importância do termo correspondente para o documento, ou ao fato de o termo ocorrer (valor numérico 1) ou não (valor numérico 0) neste documento. (A Seção III-B, na página 45, apresenta detalhes sobre vários dos possíveis procedimentos de cálculo desses valores numéricos.)

Visto que cada documento da coleção é um vetor, podemos representar o corpus como um todo através de uma matriz que denotamos por $\mathcal{M}$. As colunas dessa matriz são os vetores correspondentes aos documentos. Essa matriz é conhecida como **matriz termo-documento** (*term-document matrix*). Se n é a quantidade de documentos do corpus e m é a dimensionalidade da coleção, então $\mathcal{M}$ é de ordem $m \times n$. Nessa matriz, cada coluna corresponde a um documento do corpus, e cada linha está associada a um dos termos em $\mathcal{T}$. A Figura 1 apresenta a forma esquemática de uma matriz termo-documento.

$$\mathcal{M} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix}$$

Figura 1. Forma esquemática de uma matriz termo-documento.

Note que a quantidade de zeros na matriz termo-documento normalmente é grande. Em casos práticos, é comum a produção de matrizes de termos por documentos *esparsas*, que chegam a conter mais de 90% de zeros. A propósito, esse também é um aspecto relevante durante a implementação do VSM, onde é adequado utilizar estruturas de dados apropriadas, que armazenem apenas os valores diferentes de zero da matriz, resultando em uma significativa economia de memória necessária para armazenamento.

2) Listas invertidas: Uma estrutura de dados normalmente utilizada nessa implementação é a **lista invertida**. Para descrever a estrutura de uma lista invertida, considere, como exemplo, que tenhamos uma matriz termo-documento conforme a aprensentada na Figura 2. A lista invertida correspondente à matriz acima é apresentada na Figura 3. Nesse exemplo, que é apenas ilustrativo, a coleção é composta de 6 documentos e o vocabulário tem tamanho 5 e corresponde ao conjunto de termos $\{t_1, t_2, t_3, t_4, t_5\}$. Note entretanto, que em casos práticos, são comuuns coleções com dezenas ou centenas de milhares de documentos. É também comum encontrar coleções com vocabulários que contenham milhares ou dezenas de milhares de termos como componentes.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figura 2. Exemplo de matriz termo-documento.

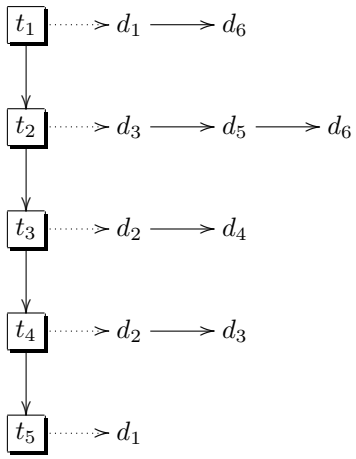

Figura 3. Lista invertida correspondente à matriz da Figura 2.

3) Bolsa de palavras: A princípio, a simplicidade do VSM pode parecer uma desvantagem, visto que ele desconsidera qualquer aspecto acerca da estrutura lingüística do texto. Em particular, as dependências entres os termos do vocabulário são ignoradas no VSM.

Por exemplo, considere dois documentos em uma coleção, cada um composto por uma das duas frases a seguir: "*As máquinas no aprendizado dos alunos*" e "*Os alunos de Aprendizado de Máquinas*". Esses dois documentos seriam considerados equivalentes no VSM, visto que ele não considera a *ordem* de ocorrência das palavras nos documentos da coleção, e considerando que as palavras "*As*", "*no*", "*dos*", "*Os*" e "*de*" seriam eliminadas durante o pré-processamento.

De fato, ambos os documentos do exemplo apresentado no parágrafo anterior seriam representados pelos termos "*alunos*",

"*aprendizado*" e "*máquinas*" do vocabulário (considerando que não foi aplicada a normalização morfológica), sem nenhuma alusão à ordem na qual esses termos ocorrem em cada documento. Por esse motivo, outro nome pelo qual a representação VSM é conhecida é a denominada *Bag-Of-Words* (BoW), nome que remete ao fato de cada documento ser representado como uma bolsa de palavras, na qual a ordem de ocorrência de cada palavra no documento *não* é considerada.

Paradoxalmente, essa característica de desconsiderar a ordem de ocorrência das palavras em cada documento não prejudica a eficiência desse modelo de representação em diversas tarefas relevantes da MDT.

B. Cálculo de pesos para os termos

Os valores numéricos $w_{ij}, i = 1..|\mathcal{T}|$ correspondentes aos elementos da matriz termo-documento $\mathcal{M}$ são chamados de **pesos**. Existem diversos procedimentos alternativos para o cálculo dos pesos de uma bolsa de palavras. Entretanto, de uma forma geral, esses procedimentos tomam como ponto de partida a frequência de ocorrência de cada termo, em cada documento e/ou na coleção como um todo. Esta Seção descreve os principais procedimentos existentes para cálculo de pesos. Entretanto, é importante notar que existem diversas variantes propostas na literatura para o cálculo dos pesos dos termos, além das que apresentamos aqui. Veja as notas bibliográficas deste Capítulo para obter referências para trabalhos que descrevem procedimentos alternativos de cálculo.

1) Medida 0/1: O procedimento de cálculo mais simples é aquele em que os pesos são binários, ou seja, cada componente w_{ij} de $\mathcal{M}$ é tal que $w_{ij} \in \{0, 1\}$. Neste procedimento, w_{ij} recebe o valor 1 quando o documento d_j contém pelo menos uma ocorrência do termo t_i, e recebe o valor 0 em caso contrário. Note que este procedimento de cálculo produz uma matriz termo-documento na qual todas as entradas são binárias (com valores 0 ou 1). Podemos então resumir o procedimento de cálculo de pesos binários através da Equação 2.

$$w_{ij} = \begin{cases} 1 & \text{se } d_j \text{ contém } t_i \\ 0 & \text{se } d_j \text{ não contém } t_i \end{cases} \quad (2)$$

A representação VSM na qual são utilizados pesos obtidos a partir do conjunto {0,1} com o uso da Equação 2 é chamada de *modelo de espaço vetorial binário* (*binary vector space model*).

Uma desvantagem desse procedimento de cálculo dos pesos é que ele não leva em consideração a intuição de que termos que aparecem mais vezes (até um certo limite; veja a discussão sobre a medida TF/IDF na página 45) são mais importantes para representar um documento do que aqueles que aparecem menos vezes. De fato, note que este procedimento atribui o valor 1 a um determinado componente, se os termo correspondente aconceu ao menos uma vez no documento em questão, independentemente da quantidade de vezes que o termo ocorre.

Os procedimentos de cálculo descritos nas próximas seções adotam a estratégia diferente de atribuir um valor entre 0 e 1 a um componente, em função da sua quantidade de ocorrências. Assim, saímos de um procedimento de atribuição de pesos binários para descrever procedimentos que atribuem pesos na faixa de valores do intervalo $[0, 1]$.

2) Medida TF: Neste procedimento para cálculo de pesos, cada peso w_{ij} é dado pela função $tf(t, d)$, que possui dois argumentos, conforme a descrição a seguir:

1) o argumento t representa um dos termos do vocabulário,
2) o argumento d representa um dos documentos do corpus.

A função $tf(t, d)$ (de *term frequency*) é definida como segue: dado um termo t_i e um documento d_j, essa função retorna a quantidade de vezes que t_i ocorre em d_j. Este procedimento de cálculo pode ser resumido através da Equação 3.

$$w_{ij} = tf(t_i, d_j) \quad (3)$$

3) Medida TF/IDF: O terceiro procedimento de cálculo de pesos é conhecido na literatura por medida **TF/IDF** (*term frequency/inverse document frequency*). Como motivação para esse procedimento de cálculo de pesos, considere os conceitos de *precisão* e *abrangência*, conhecidos na área de Recuperação de Informações. Se um termo tal como, por exemplo, "computador" ocorre com razoável frequência em alguns documentos de uma coleção, isso muito provavelmente indica que estes documentos discorrem sobre computadores. A associação do termo "computador" àqueles documentos irá então ajudar a recuperá-los em resposta a consultas apropriadas. Todavia, este procedimento não é adequado quando se quer obter não só alta taxa de abrangência, mas também uma boa precisão nos documentos recuperados. O fato é que alta precisão implica na habilidade de distinguir documentos individuais na coleção de outros irrelevantes à consulta em questão. Portanto, um termo cuja frequência seja alta é relevante somente se sua frequência de ocorrência não for igualmente alta em todos os documentos da coleção. Por exemplo, o termo "computador" pode não ser um termo relevante em uma coleção na qual todos os documentos versassem sobre computadores, e na qual virtualmente todos esses documentos contivessem aquele termo.

Uma melhor forma de calcular a relevância de um termo é dar maior importância (i.e., maior peso) a termos que ocorrem mais raramente na coleção. Isso porque esses termos estão certamente aptos a distinguir os poucos documentos nos quais eles ocorrem daqueles em que eles não ocorrem. O fato que resume toda esta discussão é que bons termos, são aqueles que ocorrem frequentemente em documentos individuais, mas raramente no restante dos documentos da coleção. Em outras palavras, se dois termos quaisquer t_a e t_b ocorrem com igual frequência em um documento, e t_a ocorrem em menos documentos do que t_b, então o peso atribuído t_a deve ser *maior* do que o associado a t_b.

A expressão matemática que reflete a intuição associada à medida TF/IDF descrita nos parágrafos anteriores usa as funções $tf(t, d)$ (definida na Seção anterior) e $idf(t)$. Esta última é o recíproco da *frequência documental* (*document frequency*) de um termo, denotada por $df(t)$ e definida como o número de documentos em uma coleção $\mathcal{D}$ nos quais o termo t ocorre ao menos uma vez. A função $idf(t)$ é dada pelo logaritmo do recíproco da frequência documental (isto é, $1/df(t)$) vezes a quantidade de documentos em $\mathcal{D}$ (denotada por $|\mathcal{D}|$). Veja a Equação 4.

$$idf(t) = \log\left(\frac{|\mathcal{D}|}{df(t)}\right) \quad (4)$$

Sendo assim, a medida TF/IDF considera dois fatores no cálculo de w_{ij}. O primeiro fator é a frequência de um termo t com relação a um documento d, conforme definido pela função $tf(t,d)$. O segundo fator é a função $idf(t)$. A expressão matemática da medida TF/IDF expressa a importância representativa de um termo em relação a um documento, e é dada pela Equação 5.

$$w_{ij} = tf(t_i,d_j) \times idf(t_i,d_j) = tf(t_i,d_j) \times \log\left(\frac{|\mathcal{D}|}{df(t_i)}\right) \quad (5)$$

Na Equação 5, $|\mathcal{D}|$ corresponde à quantidade de documentos na coleção, $tf(t_i,d_j)$ é a quantidade de vezes que o termo t_i ocorre no documento d_i, e $df(t_i)$ é a quantidade de documentos da coleção que possuem ao menos uma ocorrência de t_j.

A ideia subjacente à medida TF/IDF é a de que termos que ocorrem em uma coleção não têm igual força discriminatória para a caracterização dos documentos. De fato, note que a medida TF/IDF aumenta em função da quantidade de ocorrências do termo no documento. Isso traduz o conceito intuitivo de que, quanto mais um termo ocorre em um documento, maior é o indício de que este termo seja representativo do mesmo. Além disso, a expressão acima também reflete outro conceito intuitivo, a saber, o fato de que quanto mais um termo ocorre na coleção como um todo, menor é o poder representativo deste termo com relação a um documento específico dessa coleção. De fato, considerando a situação extrema em que um termo ocorre em todos os documentos da coleção, o segundo fator da Equação 5 se torna igual a zero, o que faz com que w_{ij} também seja igual a zero.

Outra forma de interpretar a medida TF/IDF é através dos *escopos de informação* que ela utiliza para determinar qual o peso de um termo t_i relativamente a um documento d_j. Em primeiro lugar, essa medida utiliza *informação local* da quantidade de ocorrências de t_i em d_j (através da função $tf(t_i,d_j)$). Em segundo lugar, essa medida também usa *informação global*, pois considera a quantidade de vezes que t_i ocorre na coleção de documentos como um todo (através da função $df(t_i)$).

4) Normalização dos Vetores: Outro aspecto importante a considerar no cálculo de pesos é que documentos *grandes* (ou seja, documentos que contêm muitos termos) são representados no VSM como vetores que possuem muitas coordenadas diferentes de zero. Isso faz com que o comprimento (módulo) desses vetores seja um valor grande, quando comparado aos dos demais documentos, o que pode resultar em distorções no cálculo de *medidas de similaridades* (veja o Apêndice A entre os documentos, e por fim influenciar nos resultados dos algoritmos de MDT aplicados.

De fato, já foi demonstrado experimentalmente que a normalização dos vetores ajuda a reduzir a tendência que favorece documentos que têm maior comprimento [9]. Por essa razão, uma prática normalmente utilizada é alterar os vetores representantes dos documentos de tal forma que todos tenham comprimento unitário. Isso equivale a um procedimento de normalização dos pesos, pois cada um dos componentes w_{ij} de um vetor é dividido pelo tamanho (ou módulo) desse vetor. A Figura 4 ilustra graficamente o problema com documentos de módulo relativamente grande.

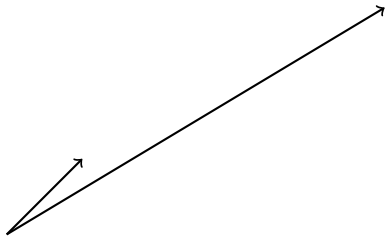

Figura 4. Vetores de módulo muito grande têm predominância sobre outros de módulo menor.

Para normalizar os vetores dos documentos, basta dividir (cada coordenada de) cada vetor $\vec{d}$ pela sua própria norma, conforme mostra a Equação 6. Nessa Equação, $||\vec{d}||$ é a norma euclideana do vetor $\vec{d}$. A aplicação dessa transformação a todos os vetores faz com que todos eles passem a ter comprimento unitário.

$$\vec{d}_{normalizado} = \frac{\vec{d}}{||\vec{d}||} \quad (6)$$

Durante o pré-processamento de um corpus, a normalização de vetores é normalmente aplicada em conjunto com algum dos procedimentos de cálculo de pesos descritos anteriormente. De fato, a combinação da medida TF/IDF com vetores normalizados é a mais comumente utilizada na prática. Nesse caso, cada peso w_{ij} é calculado pela Equação 7.

$$w_{ij} = \frac{tf(t_i,d_j) \times idf(t_i,d_j)}{\sqrt{\sum_{k=1}^{|\mathcal{T}|} \left[tf(t_i,d_j) \times idf(t_k,d_j)\right]^2}} \quad (7)$$

C. Redução da dimensionalidade da coleção

A grande dimensionalidade do espaço no qual documentos se encontram se deve ao fato do modelo de espaço vetorial considerar cada documento como um vetor em um espaço m-dimensional, onde m corresponde à quantidade de termos que ocorrem na coleção. É comum encontrar coleções de documentos contendo dezenas de milhares de termos, o que leva a uma representação de vetores em um espaço de dimensionalidade muito grande.

Vetores representativos de documentos também têm a característica intrínseca de serem esparsos (apresentarem muitas coordenadas com valor igual a zero). Isso se deve ao fato de que apenas uma pequena quantidade de termos da coleção como um todo ocorre em um determinado documento. Os demais termos da coleção que não ocorrem nesse documento têm no vetor correspondente o valor de coordenada igual a zero.

Ocorre que, mesmo após a remoção das palavras de pouco poder discriminatório e da conflação de palavras às suas raízes morfológicas, a quantidade de termos que permanecem pode ainda ser muito grande para que a coleção seja tratável computacionalmente. Isso torna necessário e adequado eliminar o

máximo possível de termos, de tal forma que os algoritmos de mineração que forem aplicados trabalhem sobre documentos representados em um espaço de dimensionalidade a menor possível. Por conta disso, outra atividade que é realizada durante o pré-processamento de documentos corresponde à aplicação de *técnicas para redução de dimensionalidade*.

Técnicas para redução de dimensionalidade se baseiam na suposição de que cada documento é representado como um conjunto de termos, onde cada termo de um documento está associado a um peso (valor numérico) que indica a importância daquela característica para o documento. Técnicas para redução de dimensionalidade podem ser divididas em dois tipos: **seleção de termos** e **extração de termos**. Independente do tipo de técnica utilizada (a extração ou seleção de termos), a redução de dimensionalidade normalmente resulta no aumento da eficiência do processamento da coleção e na diminuição do risco de o algoritmo de mineração aplicado se ajustar demasiadamente aos documentos utilizados para geração do modelo de aprendizado. As duas próximas Seções resumem as duas famílias de técnicas para redução de dimensionalidade.

1) Seleção de termos: A seleção de termos tem o objetivo de selecionar os termos mais representativos da coleção e que, por conta disso, devem ser utilizados como características dos documentos durante a execução do algoritmo de agrupamento. Através da aplicação de alguma técnica de seleção sobre o conjunto original de termos T_o, obtém-se o conjunto de termos T_f, de tal forma que $|T_o| \gg |T_f|$ e $|T_o| \supset |T_f|$.

Uma técnica de seleção de termos bastaste utilizada na prática é definir dois valores inteiros positivos, δ_{inf} e δ_{sup}, e eliminar todos os termos cuja frequência (considerando a coleção de documentos como um todo) seja menor que δ_{inf} e maior que δ_{sup}. Esses valores são chamados de *pontos de corte*.

2) Extração de termos: A extração de termos tem o objetivo de obter um conjunto T_f a partir de T_o, o conjunto original de termos da coleção. No entanto, diferentemente da seleção de termos, o conjunto de termos obtido a partir de uma técnica de extração de termos não é um subconjunto do conjunto de termos que existem naturalmente na coleção. Duas técnicas conhecidas de extração de termos são a *Indexação Semântica Latente* (LSI, *Latent Semantic Indexing*) e o *agrupamento de termos*. A estratégia geral dessas técnicas é formar termos a partir da combinação (linear ou não) dos termos originais da coleção.

A técnica LSI leva em conta o fato de existirem dependências entre os termos componentes de um documento. Essa dependência se manifesta na forma de dados redundantes. No contexto do VSM (Seção III-A), cada termo que permanece no vocabulário representa uma dimensão no espaço vetorial onde os documentos da coleção são representados. Entretanto, em virtude do fenômeno de *sinonímia* (veja a Seção I-A)), podem permanecer no vocabulário diversas palavras que possuem o mesmo significado. Quando a representação VSM é utilizada, o resultado é que mais dimensões são criadas no espaço de termos do que o mínimo necessário para representar os documentos adequadamente. Nesse contexto, seria útil que houvesse alguma forma de detectar quais termos são sinônimos e substituí-los por um único termo. Com a aplicação da técnica LSI, algumas palavras (termos) com significado similar são mapeadas (projetadas) em uma mesma dimensão. Com efeito, diversas dimensões (correspondentes às palavras com significado similar) são substituídas por uma única dimensão, que representa este significado. Essa técnica simula o comportamento humano de julgar a similaridade considerando o *significado* entre os termos.

A técnica de agrupamento de termos consiste em formar grupos de termos, de acordo com a similaridade entres estes últimos: termos similares são posicionados em um mesmo grupo; termo com baixa similaridade são alocados em grupos distintos. Para o cálculo da similaridade entre um par de termos, as métricas de similaridade apresentadas no Apêndice A podem ser usadas. Uma vez formados os grupos, o conjunto de termos $\mathcal{T}_i$ que constituem um grupo qualquer são substituídos no vocabulário por outro conjunto de termos $\mathcal{T}_i$ que representem esse grupo, de tal forma que $|T_i| < |T_f|$. O resultado desse procedimento é a diminuição do tamanho do vocabulário usado na representação da coleção. Em [10], Linden descreve diversos algoritmos de agrupamento aplicáveis a documentos (i.e., colunas da matriz termo-documento). É também perfeitamente possível a aplicação desses algoritmos para agrupar termos de uma coleção (linhas da matriz termo-documento).

IV. Visão geral da tarefa de classificação

A tarefa de *classificação* tem como objetivo produzir um modelo que permita mapear um conjunto de objetos (documentos, imagens, registros em uma tabela, etc.) em um conjunto de *categorias*. No caso específico de *Text Mining*, no entanto, os objetos utilizados na construção do modelo de classificação são documentos textuais e a tarefa recebe o nome de *Classificação de Documentos*.

A. Abordagens para construção de classificadores

Na década de 1980, a forma usual de construção de classificadores era a manual, com o uso da abordagem da *engenharia do conhecimento*. Atualmente, a abordagem utilizada é a do *aprendizado indutivo*. A seguir, apresentamos essas duas abordagens em mais detalhes.

1) Abordagem da Engenharia do Conhecimento: Nessa construção, um conjuntos de regras era definido por seres humanos. Naquela época, era comum o uso de técnicas da *Engenharia do Conhecimento* (*Knowledge Engineering*) para definição manual de regras de classificação, que eram posteriomente inseridas em *sistemas especialistas*. Essas regras de classificação eram definidas por especialistas do domínio.

Como exemplo da abordagem baseada em engenharia do conhecimento para geração de classificadores, considere a seguinte frase: "*No Brasil, há hoje 600 unidades do carro em mãos de colecionadores. O modelo T desse automóvel foi montado no país entre 1919 e 1926 com peças que vinham dos EUA.*". Ao ler esta frase, um especialista de domínio pode extrair as seguintes regras para compor o classificador:

- Regra 1: (modelo **or** carro) **and** automo* → Setor Automobilístico

- Regra 2: (avi* **or** aero*) **and** passage* $\rightarrow$ Setor de Aviação
- Regra 3 ...

A Regra 1 representa o conhecimento daquele especialista de domínio de que um documento que contenha uma ou mais ocorrências das palavras *modelo* ou *carro* e simultaneamente alguma ocorrência de alguma palavra que comece por *automo* deve ser classificado como pertencente à categoria "Setor Automobilístico". Já na Regra 2 está implícito o conhecimento de que uma ou mais ocorrências de palavras de prefixos *avi* ou *aero*, e que também contenham palavras cujo prefixo é *passage* indicam um documento que deve ser classificado como "Setor de Aviação".

Há duas principais desvantagens na abordagem baseada em Engenharia de Conhecimento para construção de classificadores:

1) Em primeiro lugar, a montagem manual das regras de classificação pelos especialistas do domínio consome tempo e é bastante trabalhosa. Por exemplo, a cada novo documento que deve ser adicionado à coleção MEDLINE (http://www.ncbi.nlm.nih.gov/PubMed) devem ser adicionados descritores provenientes da hierarquia de conceitos MeSH (que atualmente possui 20.000 descritores!). Esse processo manual demanda 2 milhões de dólares/ano [11].
2) Outro aspecto negativo da abordagem baseada em regras é que diferentes especialistas de domínio podem gerar regras inconsistentes entre si (i.e., que predizem diferentes classes para um mesmo documento). De fato, este problema da inconsistência tende a se agravar conforme aumenta o tamanho do conjunto de regras de classificação geradas.

As desvantagens descritas acima serviram de motivação para a substituição da Engenharia do Conhecimento por outra abordagem ao longo dos anos. O detalhamento dessa outra abordagem para geração de classificadores, a baseada em *aprendizado indutivo*, é feito na próxima Seção.

2) Abordagem baseada em aprendizado indutivo: Atualmente, a construção automática de classificadores é a abordagem dominante, na qual técnicas de *Aprendizado de Máquina* (*Machine Learning*) são utilizadas. Nesta abordagem, chamada de aprendizado indutivo, um conjunto de documentos de exemplo é apresentado para um algoritmo de classificação. Esse algoritmo deve então produzir uma representação ou regras de decisão para classificar futuros documentos.

Em um algoritmo cujo objetivo é a geração de um modelo de classificação de documentos, a entrada fornecida é um conjunto de documentos $\mathcal{D}$, onde cada um deles está associado a uma ou mais classes pré-definidas de um conjunto finito $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$. Mais formalmente, a entrada é apresentada ao algoritmo de classificação na forma de um conjunto de pares objetos $\langle d_i, C_i \rangle$ onde d_i é o i-ésimo documento no conjunto $\mathcal{D}$, e $\mathcal{C}_i$ é um subconjunto (possivelmente vazio) de $\mathcal{C}$ que corresponde às classes associadas a d_i.

Os documentos em $\mathcal{D}$ e suas classes correspondentes são usados pelo algoritmo de classificação na construção de um *modelo de classificação*. Uma vez construído, esse modelo pode ser usado para, dado um *novo* documento d, realizar sua classificação, isto é, identificar qual o conjunto de classes mais adequado para associar a d. Nesse sentido, uma possível interpretação da fase de geração do modelo de classificação é a de que há um *supervisor externo* que, para cada documento d_i em $\mathcal{D}$, ensina ao algoritmo qual é a classificação correta de d_i. Por esse motivo, a classificação é então uma tarefa que se encaixa na categoria de técnicas de *aprendizado supervisionado* (*supervised learning*). Sendo assim, podemos dizer que métodos de classificação possuem uma fase inicial, denominada *fase de treinamento*. Nessa fase, o algoritmo de classificação é apresentado a exemplos (documentos) corretamente classificados.

Há duas vantagens principais da abordagem baseada em aprendizado indutivo sobre a baseada em engenharia do conhecimento. Em primeiro lugar, a primeira abordagem permite criar classificadores mais precisos. Além disso, ela é também menos cara e menos demorada. Entretanto, uma desvantagem da abordagem baseada em aprendizado indutivo é que sua aplicação pressupóe a existência de uma coleção de treinamento rotulada, ou seja, de um conjunto de documentos, para cada um dos quais são conhecidas as classes correspondentes.

B. Variantes do problema

Em função das propriedades do conjunto de classes $\mathcal{C}$ utilizado, há diversas variantes do problema de classificação. Esta Seção apresenta uma taxonomia que pode ser aplicada para caracterizar determinado problema de classificação.

Em primeiro lugar, com relação à cardinalidade do conjunto $\mathcal{C}$, um problema de classificação pode ser *binário* ou *n-ário*. No primeiro caso, existem apenas duas classes no problema de classificação (i.e., $|\mathcal{C}| = 2$). No segundo caso, existem n classes em $\mathcal{C}$, $n > 2$. Por vezes, pode ser conveniente do ponto de vista prático tratar um problema de classificação n-ária como n problemas de classificação binária. Neste caso, cada tarefa de classificação binária resultante procura resolver o problema de gerar um modelo de classificação que permite classificar um documento novo como pertencente ou não a uma classe, para todas as n classes.

Outra forma de categorizar problemas de classificação é com respeito aos eventuais relacionamentos entre os elementos de $\mathcal{C}$. No primeiro caso, considera-se que não há relações entre as classes, que essas relações são desconhecidas, ou que são irrelevantes. No caso da existência de relações entre os elementos de $\mathcal{C}$, o cenário mais comum é aquele em que esses elementos formam uma *hierarquia de conceitos*, uns mais específicos, outros mais genéricos. Neste último caso, dizemos que estamos diante de um problema de *classificação hierárquica*. Como exemplo, citamos a hierarquia de conceitos denominada MeSH. Cada conceito dessa hierarquia é usado para rotular documentos da coleção PUBMED, composta de artigos da área médica. Em particular, os conceitos "Cheek", "Chin" e "Eye" são casos particulares do conceito "Face" nessa hierarquia. Em problemas de classificação desta natureza, se um documento é associado a determinada classe c, esse mesmo documento é indiretamente associado aos conceitos (classes) mais genéricos da hierarquia alcançaveis a partir de c.

Uma terceira forma de categorização de problemas de classificação diz respeito à quantidade de classes que estão associadas a cada documento em $\mathcal{D}$. Nos problemas de *classificação de única classe* (*single-class classification*), cada documento em $\mathcal{D}$ está associado a apenas uma classe. Já nos problemas de *classificação multi-classes* (*multi-class classification*), cada documento está associado a zero ou mais classes.

A seguir, resumimos a taxonomia das diferentes formas pelas quais um problema de classificação pode ser apresentado. Essa taxonomia é relativamente padronizada na literatura acerca da tarefa de classificação. A lista abaixo apresenta também os nomes originais utilizados na literatura estrangeira.

- Com relação à quantidade de elementos em $\mathcal{C}$: *binária* (*binary*) versus n*-ária* (*multi-way*).
- Com relação à existencia de relações hierárquicas entre os elementos de $\mathcal{C}$: *não hierárquica* (*flat*) versus *hierárquica* (*hierarchical*).
- Com relação à quantidade de classes associadas a cada documento em $\mathcal{D}$: *única classe* (*single-category*) versus *multi-classes* (*multi-category*)

As diversas variantes do problema de classificação descritas acima podem ser mescladas. Como exemplos, considere duas coleções de documentos que devem ser usadas para geração de modelos de classificação. A primeira coleção corresponde a diversas mensagens de correio eletrônico, onde cada documento está associado a um e apenas um elemento do conjunto de classes $C = \{\text{spam, não-spam}\}$. Esse é portanto um problema de classificação binária de única classe. Outro exemplo é uma coleção de documentos em uma agência de notícias, onde cada um deles está associado a uma ou mais classes no conjunto $C = \{\text{esportes, política, internacional}\}$. Esse é portanto um problema de classificação ternária multi-classes.

C. Passos da tarefa

A tarefa de classificação pode ser dividida em diversos passos. A seguir, resumimos os passos típicos que devem ser realizados para aplicação da tarefa de classificação sobre uma coleção de documentos. Vários desses passos são descritos em maiores detalhes nas seções restantes deste Capítulo.

1) Definir o conjunto de classes e os potenciais relacionamentos entre elas. Esse passo envolve definir o conjunto $\mathcal{C}$ de classes. Envolve também definir eventuais relacionamentos entre elementos desse conjunto, conforme mencionado na Seção IV-B. Esse passo é dependente do domínio da aplicação.
2) Rotular os textos (documentos). Este passo corresponde a associar a cada documento em $\mathcal{D}$ um subconjunto de $\mathcal{C}$. Normalmente, este passo é normalmente realizado por especialistas (anotadores) com relação à coleção de documentos $\mathcal{D}$. Se for feito manualmente, esse passo é de difícil realização, consome tempo, além de haver o potencial de inconsistência entre as decisões dos anotadores. Entretanto, é também possível que os documentos em $\mathcal{D}$ sejam previamente *agrupados*, para facilitar a atribuição de classes. Em [12], o leitor pode encontra uma descrição detalhada da tarefa de agrupamento de documentos.
3) Selecionar/Extrair características a ser utilizadas para representar os documentos da coleção. Esse passo corresponde a aplicar técnicas de redução de dimensionalidade, conforme definido na Seção III-C. Na verdade, existem técnicas de redução de dimensionalidade específicas para a tarefa de classificação. Veja [13] para detalhes.
4) Selecionar um método de classificação para treinar o classificador. O objetivo da fase de treinamento na tarefa de classificação produzir um modelo de classificação através da observação de exemplos. Esse modelo de classificação, posteriormente, deve permitir classificar corretamente documentos que não foram usados como exemplos. Na Seção V, descrevemos diversos algoritmos que podem ser usados para produzir modelos de classificação.
5) Avaliar o classificador. O objetivo de um algoritmo de classificação é portanto inferir um modelo de classificação que permita associar documentos a zero ou mais das classes do conjunto $\mathcal{C}$. Um aspecto importante a notar é que esse modelo de classificação idealmente deve mapear de forma correta documentos não contidos em $\mathcal{D}$. Desta forma, para um documento novo d, o classificador deve *predizer* (com um certo grau de certeza) a classificação correta para d. Nesse contexto, é importante averiguar a *qualidade* do modelo de classificação gerado por um algoritmo, com o objetivo de ter uma noção do quão efetivo será este modelo quando for apresentado a documentos que não foram vistos durante a fase de treinamento. Usualmente, o modelo de classificação resultante da aplicação do algoritmo deve ser validado com o uso de um conjunto de documentos que não foi utilizado na fase de aprendizado (treinamento). O objetivo de validar esse modelo é averiguar sua capacidade preditiva sobre documentos não utilizados durante sua geração. Na Seção VI, descrevemos diferentes abordagens para avaliar a qualidade de um classificador.
6) Usar o classificador para para classificar novos documentos. Um vez construído, o classificador ou modelo de classificação pode ser usado para predizer a classe de novos documentos. Inclusive, esse modelo de classificação pode ser incorporado em um sistema de escopo mais amplo. Por exemplo, é comum em sistemas de correio eletrônico a existência de uma funcionalidade para filtrar os chamados *spams*, mensagens de correio eletrônico que normalmente correspondem a conteúdo de propaganda indesejável.

V. Algoritmos de classificação

Um algoritmo de classificação toma como entrada um mapeamento da forma $f : \mathcal{D} \rightarrow \mathcal{C}$. Essa função é apresentada explicitamente como um conjunto de documentos $\mathcal{D}$ e suas respectivas classes retiradas de um conjunto $\mathcal{C}$. A partir desse conjunto, o algoritmo então constrói um **modelo preditivo** (também chamado de *modelo de classificação* ou *classificador*). De forma geral, um classificador é construído

através de um procedimento de **treinamento** no qual f é usada para inferir o modelo de classificação. Por esse motivo, denominamos f de *conjunto de treinamento*.

Uma vez construído, um modelo de classificação pode ser usado na predição da classes (ou das classes) de novos documentos. A literatura sobre a tarefa de classificação é bastante rica em termos de algoritmos para geração desses modelos. Nas próximas seções, descrevemos três deles: Rocchio, k-NN, Classificador Bayesiano. Por simplicidade, nessa descrição, consideramos que cada documento em $\mathcal{D}$ está associado a apenas uma classe. Consideramos também que o conjunto de classes é formado por m elementos, isto é, $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$. Além disso, usamos a notação $c(d_j)$ para denotar um valor inteiro positivo correspondente à classe (i.e., o elemento de $\mathcal{C}$) associada ao documento d_j. Sendo assim, $1 \leq c(d_j) \leq m$. Outra notação utilizada é usar $\vec{d}$ para denotar a *representação vetorial* (no sentido do modelo de espaço vetorial; veja a Seção III-A) do documento $d \in \mathcal{D}$.

A. Algoritmo Rocchio

Esse algoritmo interpreta cada documento do conjunto de treinamento como um vetor no espaço n-dimensional. Mais especificamente, a entrada para este algoritmo é uma *matriz de termos por documentos*, cuja construção é descrita na Seção III-A1. Dada essa matriz, o algoritmo Rocchio constrói vetores representativos de cada uma das m classes definidas no conjunto de treinamento.

O vetor representativo dos documentos associados à classe c_i é da forma $\vec{p_k} = (p_{k1}, p_{k2}, \ldots, p_{kn})$, $1 \leq k \leq m$. Cada $\vec{p_k}$ é chamado de **vetor protótipo** (*prototype vector*), e é definido como a média dos vetores correspondentes aos documentos da classe c_k. Sendo assim, o modelo de classificação produzido por esse algoritmo corresponde a m vetores protótipos. O Algoritmo 1 apresenta o procedimento de treinamento correspondente ao algoritmo Rocchio.

```
1:  Entrada: conjunto de treinamento.
2:  Saida: {p_k}, o conjunto de protótipos, 1 ≤ k ≤ |C|.
3:  m ← |C|
4:  for k = 1 to m do
5:      p_k ← (0, 0, ..., 0)
6:  end for
7:  for all d_j ∈ D do
8:      for k = 1 to m do
9:          if c(d_j) = k then
10:             p_k ← p_k + d_j
11:         end if
12:     end for
13: end for
```

Algoritmo 1: Rocchio - Fase de Treinamento

Uma vez criado um modelo de classificação através do Algoritmo 1, podemos utilizá-lo para classificar um novo documento d. A determinação da classe de um documento d é feita **similaridade** entre sua representação vetorial e os vetores protótipos. Mais especificamente, a similaridade entre $\vec{d}$ e cada protótipo $\vec{p_k}$ é calculada, e o protótipo mais similar é determinado. Finalmente, a classe desse protótipo mais similar é usada para classificar o documento d. O Algoritmo 2 apresenta o algoritmo de classificação utilizado pelo método Rocchio.

```
1:  Entrada: d, documento a ser classificado.
2:  Saída: c, a classe inferida para d.
3:  m ← |C|
4:  s_max ← −∞
5:  for k = 1 to m do
6:      s ← Similaridade(d, p_i)
7:      if s > s_max then
8:          s_max ← s
9:          c ← c_k
10:     end if
11: end for
12: Retorne c
```

Algoritmo 2: Rocchio - Fase de Classificação

No Algoritmo 2, a linha 6 faz uso de uma *função de similaridade* denominada *Similaridade*. Essa função toma dois vetores de mesma dimensionalidade e retorna um valor numérico que indica quanto esses vetores são similares. Note que o conceito de similaridade usado aqui é o mesmo definido no Apêndice A, no qual apresentamos diferentes expressões para cálculo de similaridades entre objetos.

B. Método k-NN: k vizinhos mais próximos

O algoritmo k-NN (k *Nearest Neighbors*) possui esse nome porque ele determina a classe de um documento d com base nas classes dos documentos do conjunto de treinamento que são **vizinhos** a d. O conceito de *vizinhança* entre documentos, fundamental para o funcionamento do k-NN, é definido através de uma *função de similaridade*, assim como no algoritmo Rocchio.

Para classificar um documento d, esse método produz uma *ordem total* sobre os documentos de $\mathcal{D}$. Essa ordem total permite enumerar os documentos em $\mathcal{D}$ de acordo com a similaridade de cada um em relação a d. A seguir, dado um número inteiro $k \geq 1$ fornecido como entrada, o algoritmo pode determinar quais são os k documentos em $\mathcal{D}$ mais similares a d. Por fim, as classes desses k vizinhos mais similares são usadas para predizer a classe de d.

Para esclarecer a idéia básica do método k-NN, considere o exemplo descrito a seguir. A Figura 5 apresenta 16 pontos localizados em um espaço bidimensional. Neste exemplo, considere que esses pontos correspondem aos documentos do conjunto de treinamento. (Em uma situação real, entretanto, a dimensionalidade do espaço seria 3 ou 4 ordens de grandeza maior, assim como também seria maior a quantidade de documentos envolvidos.) Note que, nessa figura, os pontos ou estão em *branco* ou em *preto*, o que indica as classes existentes no conjunto de treinamento.

Agora, considere a Figura 6, que apresenta o mesmo conjunto de treinamento da Figura 5 e, adicionalmente, um novo ponto cuja classe desejamos determinar. Considerando que o valor fornecido para k seja igual a 3, o k-NN toma esse novo

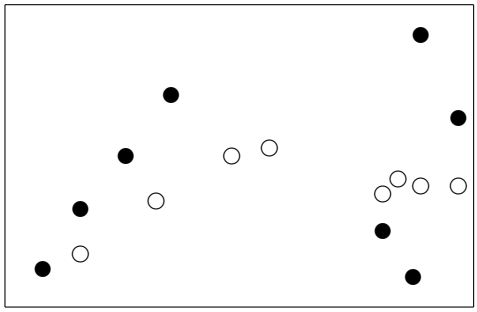

Figura 5. Conjunto de treinamento fictício de 16 elementos.

1: **Entrada**: $\mathcal{D}$, conjunto de documentos; d, documento a ser classificado.
2: **Saída**: $\mathcal{K}$, a k-vizinhança de d.
3: **for all** $d_j \in \mathcal{D}$ **do**
4: $\quad s_j \leftarrow Similaridade(d_j, d)$
5: **end for**
6: $\mathcal{D}_{Sort} \leftarrow$ lista de documentos em $\mathcal{D}$ ordenada por valores decrescentes de s_j
7: $\mathcal{K} \leftarrow$ conjunto dos primeiros k documentos em $\mathcal{D}_{Sort}$
8: Retorne $\mathcal{K}$.

Algoritmo 3: k-NN - Definição da k-vizinhança de d

ponto e o utiliza como centro de uma circunferência que, por conta da distribuição dos objetos no espaço bidimensional, delimita 3 elementos do conjunto de treinamento. Neste exemplo, esses elementos são os *vizinhos* do ponto a classificar. A seguir, o k-NN contabiliza as quantidades das diferentes classes dos pontos *internos* à circunferência para determinar a classe do novo ponto. No exemplo, o k-NN classificaria o novo ponto como *preto*, visto que esta é a classe majoritária dentre os vizinhos.

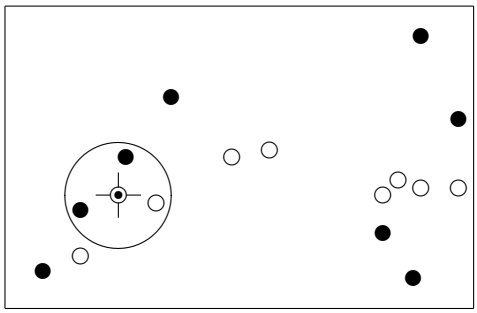

Figura 6. Na versão mais simples do k-NN, a classe do novo ponto é determinada pela classe majoritária dos pontos internos à circunferência definida pelo valor de $k = 3$.

O exemplo apresentado acima ilustra o funcionamento da variante mais simples do k-NN, na qual os vizinhos são considerados independentemente de sua proximidade (similaridade) maior ou menor em relação a d. Note, entretanto, que há outras versões do k-NN que usam os vizinhos de d de forma diferente. Outra versão do k-NN é a que leva em consideração não só os vizinhos em si, mas também o quanto eles estão próximo de d. Para isso, a influência de cada vizinho sobre a classe prevista para d é ponderada pelas similaridades eles. A motivação para essa variante do k-NN é que, quanto mais similar o vizinho, mais influência ele deve ter na determinação da classe de d. A descrição e os algoritmos que seguem são relativos a essa segunda versão do k-NN.

Agora vamos formalizar o método k-NN na forma de um algoritmo que determine a classe de um documento novo d. Em primeiro lugar, vamos definir a *k-vizinhança* (*k-neighborhood*) de d como os k vizinhos mais próximos de d. Sendo assim, o Algoritmo 3 permite determinar a *k-vizinhança* de um documento d fornecido como entrada. Note mais uma vez o uso da função *Similaridade*, que já tinha sido usada no Algoritmo 2.

k-vizinhança

Uma vez determinado $\mathcal{K}$, o conjunto correspondente à *k-vizinhança* de um documento d, a classe desse documento pode ser determinada. Com esse objetivo, para cada $c_i \in \mathcal{C}$, devemos produzir *estimativas de probabilidades condicionais* $\Pr(c_i|d)$, isto é, dado um documento d a ser classificado, o quão provável é de esse documento pertencer à classe c_i.

Vamos então descrever como determinar essas estimativas de probabilidades utilizando a conjunto $\mathcal{K}$. Primeiramente note que a cardinalidade de $\mathcal{K}$ é k, por definição. Considere também que $q(c_i)$ é um número inteiro não-negativo que corresponde à quantidade de documentos de $\mathcal{D}$ que são da classe c_i e que estão em $\mathcal{K}$. Para calcular a estimativa para $\Pr(c_i|d)$, basta então utilizar a Equação 8, definida a seguir.

$$\Pr(c_i|d) \approx \frac{q(c_i)}{k} \tag{8}$$

O passo final do método k-NN é utilizar as estimativas de probabilidades obtidas com a aplicação da Equação 8 para inferir a classe do documento d. Isso é feito pela escolha da classe c cuja estimativa de probabilidade é a maior dentre todas. Essa é a chamada *classe majoritária* (*majority class*) e é determinada pela expressão dada pela Equação 9.

$$c = \arg\max_{c_i \in \mathcal{C}} \Pr(c_i|d) \tag{9}$$

Para entender a Equação 9, devemos primeiramente compreender o operador $\arg\max$. De forma geral, esse operador toma um conjunto A qualquer e aplica uma função $f : A \rightarrow B$ a cada elemento desse conjunto, onde $B \subseteq \Re$. O resultado produzido pelo operador $\arg\max$ é o elemento de A cujo valor é máximo com relação a f. Em particular, se $A = \mathcal{C}$, $B = [0, 1]$ e $f(c_i) = P(c_i|d)$, então o operador $\arg\max$ retorna a classe mais provável dado o documento d a ser classificado.

operador $\arg\max$

O Algoritmo 4 resume a discussão dos parágrafos anteriores e apresenta o procedimento de classificação de um documento d com o uso de método k-NN. Note que esse algoritmo utiliza o procedimento auxiliar de determinação das *k-vizinhanças*, dado pelo Algoritmo 3.

1: **Entrada**: d, documento a ser classificado; $\mathcal{D}$, documentos do conjunto de treinamento.
2: **Saída**: c, a classe inferida para d.
3: $\mathcal{K} \leftarrow k$-vizinhança de d, de acordo com o Algoritmo 3.
4: **for all** $c_i \in \mathcal{C}$ **do**
5: $q(c_i) \leftarrow$ quantidade de documentos em $\mathcal{K}$ que pertencem à classe c_i.
6: $\Pr(c_i|d) \leftarrow \frac{q(c_i)}{k}$ (Equação 8)
7: **end for**
8: $c \leftarrow \arg\max_{c_i \in \mathcal{C}} \Pr(c_i|d)$ (Equação 9)
9: Retorne c

Algoritmo 4: k-NN - Fase de Classificação

Um aspecto importante acerca do k-NN é que não há uma fase de treinamento explícita, na qual um *modelo de classificação* é gerado, conforme vimos no método Rocchio (Seção V-A). No Rocchio, os documentos em $\mathcal{D}$ são usados na fase de treinamento para a determinação dos prótotipos de cada classe e não são necessários na fase de classificação. Já no método k-NN, todos os documentos de $\mathcal{D}$ são mantidos para realizar a classificação de um documento novo. Outra forma de interpretar essa característica é pensar que o modelo de classificação no k-NN corresponde a todo o conjunto $\mathcal{D}$. Isto é, a fase de treinamento do k-NN consiste apenas em armazenar as representações vetoriais dos documentos em $\mathcal{D}$. Por esse motivo é que se diz que o k-NN é um método de *aprendizado tardio* (*lazy learning*).

aprendizado tardio

1) Valor do parâmetro k: Um parâmetro que deve ser definido no método k-NN é justamente o valor de k, que determina a quantidade de vizinhos a considerar na determinação da classe de um documentos. Usar $k = 1$ é uma estratégia sujeita a erros. Isso porque o único vizinho escolhido tem o potencial de ser um exemplo atípico, o que pode acontecer em caso em que há erros no conjunto de documentos usado no treinamento.

Uma estratégia mais robusta é utilizar um valor de $k > 1$ exemplos mais similares e retornar a classe mais provável destes k exemplos. Nesse caso, tipicamente o valor escolhido em problemas de classificação binária (veja a Seção IV-B) é ímpar (para evitar empates durante a determinação da classe majoritária). Valores comumente utilizados na prática são $k = 3$ ou $k = 5$.

C. Algoritmo C4.5

O C4.5 é um dos algoritmos mais tradicionais na tarefa de classificação. Esse método C4.5 procura abstrair uma **árvore de decisão** (*decision tree*) a partir de uma abordagem recursiva de particionamento da coleção $\mathcal{D}$. Utiliza, para tanto, conceitos e medidas da *Teoria da Informação*.

árvore de decisão

A fim de descrever o funcionamento do algoritmo C4.5, consideremos sua aplicação em um conjunto de documentos $\mathcal{D}$ representados de acordo com o modelo de espaço vetorial, e que o conjunto $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ contém elementos que são usados como classes dos documentos.

Uma árvore de decisão é um *modelo de conhecimento* (mais especificamente, um modelo de classificação) em que cada nó não folha da árvore representa uma decisão sobre um atributo que determina como os dados estão particionados pelos seus nós filhos. Inicialmente, a raiz da árvore representa toda a coleção $\mathcal{D}$, com exemplos misturados das várias classes. Um *predicado*, denominado ponto de separação, é escolhido como sendo a condição que melhor separa ou discrimina as classes. Tal predicado envolve exatamente um dos atributos do problema e particiona o conjunto $\mathcal{D}$ em dois ou mais subconjuntos, que são associados cada um a um nó filho. Cada novo nó abrange, portanto, uma partição de $\mathcal{D}$ que, por sua vez, é recursivamente separada, até que o conjunto de documentos associado a cada nó folha consista inteiramente ou predominantemente de elementos de uma mesma classe.

Para ilustrar o funcionamento do algoritmo c4.5, considere a Tabela I, que apresenta uma coleção fictícia de documentos representada em um espaço vetorial booleano. Observe que este problema de classificação possui três classes: *Política*, *Moda* e *Economia*.

Partido	Legenda	PIB	Real	Classe
1	1	0	0	Política
1	0	0	1	Política
0	0	0	1	Moda
0	0	1	1	Economia
0	0	1	1	Política

Tabela I
EXEMPLO FICTÍCIO DE UMA COLEÇÃO DE DOCUMENTOS REPRESENTADA EM UM ESPAÇO VETORIAL BOOLEANO.

A Figura 7 apresenta um esquema gráfico de uma árvore de decisão associada à coleção fictícia de documentos representada pela Tabela I. Nessa figura, observe os predicados que indicam os critérios de separação dos dados em todos os nós não folha. Associado a cada nó folha encontra-se um subconjunto do corpus cujos documentos satisfazem a todos os predicados pertencentes ao caminho que parte do nó raiz até o nó folha correspondente. Os documentos associados a cada nó folha devem pertencer em maioria a uma única classe de documentos. Quanto menor a diversidade de classes de documentos associados a um nó folha, maior a pureza do referido nó. Existem medidas voltadas especificamente à aferição do grau de pureza/impureza de cada nó em árvores de decisão como, por exemplo, o índice gini.

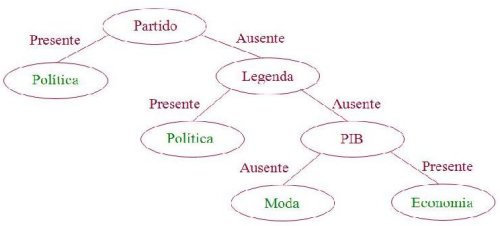

Figura 7. Exemplo de árvore de decisão em um corpus fictício sobre notícias.

Na fase de construção da Árvore de Decisão, uma árvore é gerada pelo particionamento recursivo dos dados de treinamento. O conjunto de treinamento é separado em duas ou mais

partições usando restrições sobre os conjuntos de valores de cada atributo. O processo é repetido recursivamente até que todos ou a maioria dos exemplos em cada partição pertençam a uma classe. A árvore gerada abrange todo o conjunto de treinamento e é construída em profundidade.

Há duas operações principais durante o processo de construção da árvore:

(a) a avaliação dos pontos de separação de cada nó interno da árvore e a identificação de qual o melhor ponto de separação.
(b) a criação das partições usando o melhor ponto de separação identificado para os casos pertencentes a cada nó

Uma vez determinado o melhor ponto de separação de cada nó, as partições podem ser criadas pela simples aplicação do critério de separação identificado.

Para avaliação dos pontos de separação de cada nó interno da árvore, as seguintes medidas devem ser calculadas:

- Ganho de informação considerando a partição da coleção de documentos associada ao nó em análise. Observa-se que, para o nó raiz, a coleção de documentos corresponde a $\mathcal{D}$. Para este cálculo utiliza-se a fórmula abaixo que representa a **entropia** (ou complexidade) do conjunto de documentos considerando o atributo de classificação:

$$info(S) = -\sum_{j=1}^{k} \frac{freq(C_j, S)}{|S|} \log_2 \frac{freq(C_j, S)}{|S|} \text{ bits} \tag{10}$$

  Onde:
  - S representa a partição da base de dados;
  - $freq(C_j, S)$ representa o número de vezes que a classe C_j acontece em S;
  - $|S|$ denota o número de casos do conjunto S;
  - k indica o número de classes distintas.

- Ganho de informação de cada atributo considerando a partição da base de dados associada ao nó em análise. Observa-se que, para o nó raiz, todos os atributos, com exceção do atributo de classificação, devem ser analisados. Para este cálculo utilizam-se as fórmulas abaixo sobre cada atributo:

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} info(T_i) \text{ bits} \tag{11}$$

  Onde:
  - T representa a quantidade de ocorrências na partição em análise;
  - T_i representa a quantidade de ocorrências de uma classe contidas no conjunto T;
  - n é o número de valores distintos do atributo X.

  O cálculo do ganho de informação é expresso por:

$$gain(X) = info(S) - info_X(T) \tag{12}$$

Deve então ser selecionado para construção do nó da árvore, o atributo com maior ganho de informação obtido sobre a partição em análise.

Partido	Legenda	PIB	Real	Classe do Documento
5	1	0	2	Política
4	0	1	0	Política
0	0	1	3	Moda
1	1	3	7	Economia
6	2	0	2	Política

Tabela II
EXEMPLO FICTÍCIO DE UMA BASE DE NOTÍCIAS REPRESENTADA EM UM ESPAÇO VETORIAL CLÁSSICO.

É importante ressaltar que o processo de avaliação de pontos de separação depende do domínio de cada atributo, que pode ser numérico ou categórico. No caso de classificação de documentos textuais representados em um espaço de vetores, todos os atributos são numéricos, uma vez que indicam a freqüência com que cada termo ocorre no documento. Nos casos em que os documentos são representados por vetores binários, os atributos podem ser considerados categóricos para efeito de separação, uma vez que os valores possíveis são as categorias *termo presente* e *termo ausente*, conforme ilustrado pela Tabela I.

O processo de avaliação dos pontos de separação de atributos numéricos baseia-se em testes dicotômicos da forma $A \leq v$, onde A é um atributo e v é um número real. Este processo requer a ordenação dos exemplos de treinamento baseado nos valores do atributo em análise. Por exemplo, sejam $v_1, v_2, ..., v_n$, valores ordenados de um atributo numérico A. Como qualquer valor entre v_i e v_{i+1} divide o conjunto nos mesmos dois subconjuntos, apenas $(n-1)$ possibilidades de separação precisam ser analisadas. Tipicamente, o ponto médio entre v_i e v_{i+1} é escolhido como ponto de separação. Pode ser observado, portanto, que o custo de avaliação das separações para um atributo numérico é dominado pelo custo de ordenação dos valores.

O processo de avaliação dos pontos de separação de atributos categóricos baseia-se em testes sobre cada atributo individualmente.

Em ambos os casos (atributos categóricos ou numéricos), os testes consistem em calcular o ganho de informação associado ao atributo correspondente. Este cálculo no caso de atributos categóricos foi ilustrado acima. Consideremos agora um exemplo de cálculo de ganho de informação para um atributo numérico. Suponhamos uma base de documentos similar àquela da tabela I, representada na Tabela II. Esta base foi adaptada para atributos numéricos. Conforme já mencionado, nesta representação de espaço vetorial, cada valor indica o número de vezes que o termo correspondente ocorre no documento em questão.

Consideremos o cálculo do ganho de informação para o atributo referente ao termo *Partido*. Como este atributo é numérico, é necessário a avaliação dos pontos de separação a fim de selecionar aquele que melhor particiona o conjunto de documentos. Para tanto, os valores são ordenados e os pontos médios calculados, conforme mostra a tabela III. Esta mesma tabela mostra a condição associada a cada ponto médio, assim como o resultado do cálculo do ganho de informação em cada situação. Assim, o melhor ponto de separação do

Pares	Ponto Médio	Predicado	Ganho de Inf.
0 e 1	0.5	*Partido* ≤ 0.5	0.721928
1 e 4	2.5	*Partido* ≤ 2.5	0.970951
4 e 5	4.5	*Partido* ≤ 4.5	0.419973
5 e 6	5.5	*Partido* ≤ 5.5	0.170951

Tabela III
PONTOS MÉDIOS DOS VALORES DO ATRIBUTO *Partido* E OS RESPECTIVOS GANHOS DE INFORMAÇÃO.

atributo *Partido*, ou seja, o predicado que leva ao melhor ganho de informação para o atributo *Partido* é *Partido* ≤ 2.5. Obviamente, um raciocínio análogo deve ser aplicado aos demais atributos a fim de escolher qual deverá ser o atributo e a condição de separação a ser imposta na criação da árvore de decisão. A árvore de decisão parcialmente representada na figura 8 ilustra o processo de construção da estrutura caso o maior ganho de informação estivesse associado ao atributo *Partido*.

Figura 8. Exemplo Parcial de Árvore de Decisão em um Corpus Fictício sobre Notícias

A seguir encontra-se uma versão simplificada do Algoritmo C4.5 para a fase de construção de árvores de decisão. Ela é recursiva, realizada em profundidade, e considera que cada nó da árvore gerada possui três informações:

- O nome do atributo associado
- A sub-base correspondente
- Uma lista de filhos

Cada nó da lista de filhos associada a um nó da árvore possui, por sua vez, duas informações:

- A raiz da sub-árvore associada
- Um predicado envolvendo o atributo em questão e que especifica a condição de seleção dos registros, defindo a sub-base associada a sub-árvore

Convém ressaltar que o nó raiz da árvore recebe, no início do processamento, a base de dados completa como subárvore. A versão do C4.5 descrita a seguir encontra-se subdividida em dois procedimentos (Algoritmo 5 e Algoritmo 6) e mostra o processamento a partir do nó raiz da árvore.

```
1: Entrada: 𝒟, o conjunto de treinamento.
2: Saida: Árvore de decisão 𝒜.
3: Raiz ← CriaNohArvore(D)
4: ProcessaNohArvore(Raiz)
5: ExibeArvore(Raiz)
```

Algoritmo 5: C4.5 - Fase de construção da árvore de decisão - Procedimento Principal

Quando a base de dados possui atributos ainda não processados, a função *BaseImpura()* do passo 2 do Algoritmo 6 retorna verdadeiro ou falso dependendo do valor apurado a partir de algum índice que calcule o grau de impureza da base. Quando a base de dados não tem mais atributos diferentes do atributo objetivo a serem processados, a função retorna falso. Neste caso, conforme pode ser observado no passo 21, o algoritmo considera como classe o valor prevalente na referida base.

O procedimento do passo 5 do mesmo algoritmo executa o cálculo do ganho de informação apropriado em função do tipo do atributo (categórico ou numérico). Ainda com relação ao Algoritmo 6, no passo 11, o procedimento particiona a base de dados em função dos predicados formados a partir dos valores do atributo com maior ganho de informação. Os elementos da lista de bases particionadas no passo 11 possuem, além das sub-bases, os predicados que levaram ao particionamento.

Conforme mostra o passo 21 do Algoritmo 6, caso a base de dados seja considerada suficientemente pura, ou não tenha mais atributos a serem processados, o nó corrente é considerado um nó folha e o valor prevalente do atributo objetivo nessa base é obtido e indicado para ser o rótulo da classe correspondente.

```
1: Entrada: Nó raiz da subárvore Raiz.
2: if BaseImpura(Raiz.SubBase) then
3:    MaiorGanho ← −∞
4:    for all Atb ∈ Raiz.SubBase do
5:       Gain ← CalcularGanhoInfo(Raiz.SubBase,Atb)
6:       if Gain > MaiorGanho then
7:          MaiorGanho ← Gain
8:          MelhorAtb ← Atb
9:       end if
10:   end for
11:   Raiz.Atributo ← MelhorAtb
12:   lBases ← ParticionarBase(MelhorAtb, Raiz.SubBase)
13:   for all B ∈ lBases do
14:      Filho ← CriaNohLista()
15:      Filho.Predicado ← MontaPredicado(MelhorAtb, B)
16:      Filho.SubArvore = CriaNohArvore(B)
17:      Inclui(Raiz.lFilhos, Filho)
18:      ProcessaNohArvore(Filho.SubArvore)
19:   end for
20: else
21:   Raiz.Atributo ← ObterValorAtbObj(Raiz.SubBase)
22: end if
```

Algoritmo 6: C4.5 - Fase de Construção da Árvore de Decisão - *ProcessaNohArvore()*

O processamento da árvore de decisão sobre um novo documento a ser classificado consiste em percorrer a árvore, partindo do nó raiz em direção a um nó folha que estabeleça a qual classe tal documento pertence. O caminho entre o nó raiz e o nó folha é estabelecido na medida em que os predicados associados aos nós não folha vão sendo satisfeitos pelos atributos do documento a ser classificado.

Cabe ressaltar ainda a possibilidade de conversão de uma árvore de decisão para um conjunto de regras de classificação. Uma regra de classificação é uma regra de produção em que o

consequente estabelece uma classe a qual um novo documento deverá pertencer, caso os atributos deste documento satisfaçam aos predicados estabelecidos no antecedente da regra.

D. Classificador Ingênuo de Bayes

Teorema de Bayes

O *Classificador Ingênuo de Bayes* (*Naïve Bayes Classifier*), CIB, é um método de classificação baseado na Teoria das Probabilidades. Mais especificamente, o **Teorema de Bayes** (*Bayes Theorem*) desempenha um papel crítico nesse método. Dado um novo documento d a ser classificado, o CIB associa a d a classe mais provável dentre todas as classes possíveis. Para realizar essa associação, o CIB segue um procedimento composto dos dois passos abaixo:

1) Gera uma estimativa da distribuição de *probabilidades posteriores* para cada classe.
2) Associa a d a classe mais provável, com base nessa distribuição de probabilidades.

Vamos agora formalizar o procedimento realizado pelo método CIB. Seja o conjunto de categorias $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$. Seja d o documento a ser classificado. Para classificar d, o CIB calcula e avalia $\Pr(c_i|d)$, para cada classe $c_i \in \mathcal{C}$. O valor de $\Pr(c_i|d)$ corresponde à probabilidade de o documento d pertencer à classe c_i. Dessa forma, temos que $\sum_{c_i \in \mathcal{C}} \Pr(c_i|d) = 1$, por definição. Se utilizarmos a definição de probabilidade concicional, obtemos uma expressão para o valor $\Pr(c_i|d)$, dada pela Equação 13.

$$\Pr(c_i|d) = \frac{\Pr(c_i, d)}{\Pr(d)} \tag{13}$$

Podemos utilizar o Teorema de Bayes para transformar a expressão da Equação 13. (Deixamos como exercício para o leitor a realização do algebrismo envolvido.) Quando fazemos isso, obtemos a expressão apresentada na Equação 14.

$$\Pr(c_i|d) = \frac{\Pr(c_i) \times \Pr(d|c_i)}{\Pr(d)} \tag{14}$$

No CIB, a Equação 14 é calculada para cada classe $c_i, i = 1 \ldots m$. Após esse cálculo, o método pode determinar a classe mais provável para d. A classe mais provável de d, c_{map}, é dada por:

$$\begin{aligned} c_{map} &= \arg\max_{c_i \in \mathcal{C}} \Pr(c_i|d) \\ &= \arg\max_{c_i \in \mathcal{C}} \frac{\Pr(c_i) \times \Pr(d|c_i)}{\Pr(d)} \\ &= \arg\max_{c_i \in \mathcal{C}} \Pr(c_i) \times \Pr(d|c_i) \end{aligned} \tag{15}$$

Há duas transformações relevantes na Equação 15. A primeira é a eliminação do denominador $\Pr(d)$. Essa transformação simplifica os cálculos necessários e somente é possível porque $\Pr(d)$ é um valor constante e portanto o resultado retornado pelo operador $\arg\max$ depende apenas de $\Pr(c_i) \times \Pr(d|c_i)$. A segunda transformação realizada na Equação 15 utiliza o Teorema de Bayes, conforme a Equação 14.

De toda a discussão feita até aqui sobre o CIB, podemos concluir que esse método precisa determinar *estimativas* para as distribuições de probabilidades $\Pr(c_i)$ e $\Pr(d|c_i)$, a partir dos documentos do conjunto de treinamento. Portanto, vamos agora descrever de que forma essas estimativas podem ser obtidas.

probabilidade anterior

Primeiramente, vamos descrever o procedimento para obtenção da estimativa para $\Pr(c_i)$, denominada **probabilidade anterior** (*prior probability*). É importante entender o significado do valor $\Pr(c_i)$ para uma certa classe c_i. Esse valor é a probabilidade de a classe de um documento escolhido ao acaso ser da classe c_i. Sendo assim, se n_i documentos em $\mathcal{D}$ são da classe c_i, então podemos obter uma estimativa $\hat{\Pr}(c_i)$ para $\Pr(c_i)$ através da Equação 16.

$$\Pr(c_i) \approx \hat{\Pr}(c_i) = \frac{n_i}{|\mathcal{D}|} \tag{16}$$

O motivo pelo qual $\Pr(c_i)$ é denominada probabilidade anterior está relacionado ao fato de que, se não soubermos mais nada além da distribuição de probabilidade $\Pr(c_i), i = 1 \ldots m$, podemos usar esses valores para determinar a classe mais provável para d.

Agora vamos descrever a forma de produzir uma estimativa para a probabilidade $\Pr(d|c_i), i = 1 \ldots m$. O significado desta probabilidade é o seguinte: dentre todos os documentos da classe c_i, $P(d|c_i)$ corresponde à probabilidade de selecionar ao acaso um documento com as mesmas características de d, o documento que desejamos classificar. Nesse contexto, o método CIB interpreta um documento d a ser classificado como uma conjunção de $|\mathcal{T}|$ eventos binários, onde $\mathcal{T}$ é o conjunto de termos que compõe o *léxico* extraído de $\mathcal{D}$. O k-ésimo evento binário corresponde à ocorrência ou não do termo t_k no documento d. Dessa forma, podemos considerar d como um evento composto do $|\mathcal{T}|$ eventos binários, conforme a Equação 17.

$$d = t_1 \wedge t_2 \wedge \ldots \wedge t_{|\mathcal{T}|} \tag{17}$$

De acordo com a Teoria das Probabilidades, e usando a interpretação de d como um evento conjunto conforme a Equação 17, podemos escrever a Equação 18 para obter $\Pr(d|c_i)$.

$$\begin{aligned} \Pr(d|c_i) = {} & \Pr(c_i) \times \\ & \Pr(t_1|c_i) \times \\ & \Pr(t_2|t_1 \wedge c_i) \times \\ & \Pr(t_3|t_1 \wedge t_2 \wedge c_i) \times \\ & \times \ldots \times \\ & \Pr(t_{|\mathcal{T}|}|t_1 \wedge t_2 \wedge \ldots \wedge t_{|\mathcal{T}|-1} \wedge c_i) \end{aligned} \tag{18}$$

De acordo com a Equação 18, $\Pr(d|c_i)$ pode ser calculado pelo produtório de $|\mathcal{T}|+1$ fatores. Portanto, para calcular uma estimativa para $\Pr(d|c_i)$, devemos produzir estimativas para todos os $|\mathcal{T}| + 1$ fatores envolvidos. Esse aspecto é um complicador, se considerarmos o custo computacional necessário para o cálculo desses fatores. Além disso, a quantidade de documento no conjunto de treinamento deve ser suficiente para que estimativas confiáveis possam ser produzidas. Nesse ponto, o método CIB faz uma suposição sobre a dependência

existente entre os eventos $e(t_1), e(t_2), e(t_3), \ldots, e(t_1)$. Essa suposição facilita o cálculo da estimativa para $\Pr(d|c_i)$, conforme descrevemos a seguir.

A suposição que o CIB utiliza é considerar que os termos de um documento são **condicionalmente independentes, dada a classe** c_i. Isso quer dizer que, de acordo com essa suposição, o fato de um documento d conter um determinado termo t não diz nada acerca da probabilidade de d conter também outro termo $t^{'}$. Dessa forma, a Equação pode ser significativamente simplificada, o que resulta na Equação 19.

$$\Pr(d|c_i) = \Pr(t_1 \wedge t_2 \wedge \ldots \wedge t_{|\mathcal{T}|}|c_i) = \Pr(c_i) \times \prod_{j=1}^{|\mathcal{T}|} \Pr(t_j|c_i) \tag{19}$$

Note que a Equação 19 ainda contém $|\mathcal{T}|+1$ fatores, assim como na Equação 18. Entretanto, note também que a primeira é uma simplificação da segunda, visto que, do terceiro ao último fator da Equação 18, os termos foram removidos do *condicionante*. Essa simplificação facilita substancialmente os cálculos das estimativas de probabilidades, conforme descrevemos a seguir.

Uma vez adotada a suposição de independência condicional entre os termos, podemos descrever o procedimento para obter estimativas para os fatores correspondentes às probabilidades condicionais da forma $P(t_j|c_i)$. Suponha que há q_i ocorrências de termos nos documentos de $\mathcal{D}$ pertencentes à classe c_i. Considere ainda que q_{ij} corresponde à quantidade de ocorrências do termo t_j entre as n_i ocorrências anteriores. Então a estimativa $\hat{p}(t_j|c_i)$ pode ser obtida pela Equação 20.

$$\Pr(t_j|c_i) \approx \hat{\Pr}(t_j|c_i) = \frac{q_{ij}}{q_i} \tag{20}$$

De posse da Equação 20, podemos definir uma expressão para obter uma estimativa para $\Pr(d|c_i)$, que denotamos por $\hat{\Pr}(d|c_i)$. Essa expressão é apresentada na Equação 21.

$$\Pr(d|c_i) \approx \hat{\Pr}(d|c_i) = \hat{\Pr}(c_i) \times \prod_{j=1}^{|\mathcal{T}|} \hat{\Pr}(t_j|c_i) \tag{21}$$

A suposição de independência condicional entre os termos de um documento certamente não condiz com a realidade. Por exemplo, em uma coleção de documentos, se sabemos que um dos documentos contém a palavra *Hong*, isso aumenta nossa expectativa de encontrar a palavra *Kong*. Entretanto, apesar de a suposição adotada pelo CIB não refletir o que acontece em coleções de documentos reais, experimentos mostram que esse método é efetivo na prática. Em [14], há uma explicação para os bons resultados práticos obtidos com o método CIB.

*1) Suavização de Laplace (*Laplace Smoothing*):* Sabemos até aqui que o cálculo das estimativas para as probabilidades no método CIB é baseado em contagens de *frequências* sobre a coleção de treinamento $\mathcal{D}$. Por exemplo, para obter a estimativa para as probabilidades anteriores $\hat{p}(c_i)$, precisamos determinar com que frequência encontramos documentos em $\mathcal{D}$ que pertecem à classe c_i. Esse valor pode ser diretamente obtido pela Equação 16.

Entretanto, no cálculo das estimativas para as probabilidades condicionais, há um complicador adicional: frequências iguais a zero fazem com que a estimativa da probabilidade condicional seja igual a zero. Para ententer isso, perceba que basta que um dos fatores da Equação 21 seja igual a zero para que todo o produtório seja também igual a zero. Em particular, a freqüência $\frac{q_{ij}}{q_i}$ é igual a zero quando o termo t_j não ocorre nos documentos rotulados com classe c_i (porque, neste caso, $q_{ij} = 0$ na Equação 20). Para prevenir a ocorrências de frequencias iguais a zero, devemos *suavizar* as estimativas.

De forma geral, o procedimento de *suavizar* uma estimativa de probabilidade e significa adicionar a ela um pequeno valor positivo δ, de tal forma que a nova estimativa é $e + \delta$. O resultado disso é que estimativas de probabilidade que são iguais a zero se tornam maiores que zero.

Uma das técnicas usadas para suavizar estimativas de probabilidades é a **Suavização de Laplace** (*Laplace Smoothing*). Quando aplicada à Equação 20, essa técnica permite reescrevê-la, conforme apresentado na Equação 22

Suavização de Laplace

$$\hat{\Pr}(t_j|c_i) = \frac{q_{ij} + 1}{q_i + |\mathcal{T}|} \tag{22}$$

Essa técnica pressupõe a observação de $|a_k|$ exemplos *virtuais*, onde $|a_k|$ é a *aridade* (i.e., a quantidade de valores possíveis) do atributo previsor a_k.

2) Transformação de produtório em somatório: Além da Suavização de Laplace, outro artifício de implementação é normalmente utilizado no CIB. Considere novamente a Equação 21. Repare que essa equação apresenta um produtório sobre os fatores $\hat{\Pr}(t_j|c_i)$. Do ponto de vista computacional, esse produtório representa um complicador para o cálculo envolvido na Equação 21. Isso porque os valores $\hat{\Pr}(t_j|c_i)$ são normalmente muito próximos de 0, o que faz com que o seu produtório seja ainda mais próximo de zero. Considerando que computadores possuem uma capacidade finita para representação de números reais, isso pode levar a erros de aproximação do cálculo desejado.

Para contornar esse problema, utilizamos uma propriedade do operador $\arg\max$ e da função logarítmica $f(x) = \log(x)$, conforme descrito a seguir.

Primeiramente, note que a função $\log(x)$ é *monotonicamente crescente*, o que significa que, se $x_1 \geq x_2$, então $\log(x_1) \geq \log(x_2)$. Sendo assim, se aplicarmos a função logarítmica a cada elemento da lista passada como argumento para o operador $\arg\max$, o resultado produzido por esse operador permanece o mesmo. Ou seja:

$$\arg\max_{c_i \in \mathcal{C}} f(c_i) \quad = \quad \arg\max_{c_i \in \mathcal{C}} \log\left[f(c_i))\right] \tag{23}$$

Note também que podemos usar uma propriedade da função logarítmica segundo a qual o logaritmo de um produtório é igual ao somatório dos logaritmos. Como resultado, transformamos o produtório da expressão original de c_{map} em um somatório. Isso é vantajoso do ponto de vista computacional, visto que somas são menos sujeitas a erros de aproximação numérica do que produtos. Esse desenvolvimento é apresentado na Equação 24.

$$
\begin{aligned}
c_{map} &= \arg\max_{c_i \in \mathcal{C}} \left[\hat{\Pr}(c_i) \times \prod_j \hat{\Pr}(t_j|c_i) \right] \\
&= \arg\max_{c_i \in \mathcal{C}} \log \left[\hat{\Pr}(c_i) \times \prod_j \hat{\Pr}(t_j|c_i) \right] \\
&= \arg\max_{c_i \in \mathcal{C}} \left[\log \hat{\Pr}(c_i) + \log \prod_j \hat{\Pr}(t_j|c_i) \right] \\
&= \arg\max_{c_i \in \mathcal{C}} \left[\log \hat{\Pr}(c_i) + \sum_j \log \hat{\Pr}(t_j|c_i) \right]
\end{aligned} \tag{24}
$$

Pelo que foi descrito até aqui, podemos concluir que, de posse das estimativas $\hat{\Pr}(c_i)$ e $\hat{\Pr}(t_j|c_i)$, a classe mais provável de um documento d pode ser determinada pela Equação 25.

$$
c_{map} = \arg\max_{c_i \in \mathcal{C}} \left[\log \hat{\Pr}(c_i) + \sum_j \log \hat{\Pr}(t_j|c_i) \right] \tag{25}
$$

3) Algoritmos: Estamos agora em condições de apresentar os algoritmos envolvidos no método CIB. O Algoritmo 7 apresenta o procedimento de treinamento do CIB. Observe que as linhas 6 e 11 correspondem a aplicações das Equações 16 e 20, respectivamente.

1: **Entrada**: Conjunto de treinamento $\mathcal{D}$.
2: **Saída**: Estimativas para $\Pr(c_i)$ e $\Pr(t_j|c_i)$, $i = 1 \ldots |\mathcal{C}|$, $j = 1 \ldots |\mathcal{T}|$.
3: **for all** $c_i \in \mathcal{C}$ **do**
4: $\quad \mathcal{D}_i \leftarrow$ documentos em $\mathcal{D}$ pertencentes à classe c_i
5: $\quad n_i \leftarrow |\mathcal{D}_i|$
6: $\quad \hat{\Pr}(c_i) \leftarrow n_i/|\mathcal{D}|$
7: $\quad \mathcal{T}_i \leftarrow$ união de todos os termos em $\mathcal{D}_i$
8: $\quad q_i \leftarrow$ quantidade de ocorrências de termos em $\mathcal{D}_i$
9: $\quad$ **for all** $t_j \in \mathcal{T}_i$ **do**
10: $\quad\quad q_{ij} \leftarrow$ quantidade de ocorrências de t_j em $\mathcal{D}_i$
11: $\quad\quad \hat{\Pr}(t_j|c_i) \leftarrow (q_{ij} + 1)/(q_i + |\mathcal{T}|)$
12: $\quad$ **end for**
13: **end for**

Algoritmo 7: CIB - Treinamento

Para classificar um documento com o uso do CIB, aplicamos o Algoritmo 8.

1: **Entrada**: d, o documento a ser classificado.
2: $\mathcal{T}_d \leftarrow$ conjunto de termos do léxico que ocorrem em d.
3: Retorne c_{map}, tal que
$c_{map} = \arg\max_{c_i \in \mathcal{C}} \left[\log \hat{\Pr}(c_i) + \sum_{t_j \in \mathcal{T}_d} \log \hat{\Pr}(t_j|c_i) \right]$

Algoritmo 8: CIB - Classificação

Documento	**Conteúdo**
d_1	Human machine interface for PARC computer applications
d_2	A survey of user opinion of computer system response time
d_3	The EPS user interface management system
d_4	System and human system engineering testing of EPS
d_5	Relation of user perceived response time to error measurement
d_6	The generation of random, binary, ordered trees
d_7	The intersection graph of paths in trees
d_8	Graph minors IV: Widths of trees and well-quasi-ordering
d_9	Graph minors: A survey

Tabela IV
COLEÇÃO DE DOCUMENTOS DE EXEMPLO. ESSA COLEÇÃO ESTÁ DIVIDIDA EM DUAS CLASSES. OS PRIMEIROS 5 DOCUMENTOS PERTENCEM À CLASSE "*Human Machine Interaction*" E OS ÚLTIMOS 4 PERTENCEM À CLASSE "*Graphs*".

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
HUMAN	1	0	0	1	0	0	0	0	0
INTERFACE	1	0	1	0	0	0	0	0	0
COMPUTER	1	1	0	0	0	0	0	0	0
USER	0	1	1	0	1	0	0	0	0
SYSTEM	0	1	1	2	0	0	0	0	0
RESPONSE	0	1	0	0	1	0	0	0	0
TIME	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
SURVEY	0	1	0	0	0	0	0	0	1
TREES	0	0	0	0	0	1	1	1	0
GRAPH	0	0	0	0	0	0	1	1	1
MINORS	0	0	0	0	0	0	0	1	1
Classe	c_1	c_1	c_1	c_1	c_1	c_2	c_2	c_2	c_2

Tabela V
BOLSA DE PALAVRAS CORRESPONDENTE À COLEÇÃO DA TABELA IV.

4) Exemplo Ilustrativo: Para ilustrar a aplicação do método CIB, consideremos a coleção de documentos apresentada na Tabela IV. Para manter o exemplo em um nível de simplicidade adequado, considere que cada documento é representado apenas pelo seu título. Essa coleção possui nove documentos. Desses, os primeiros 5 são rotulados com a classe "Human Machine Interaction". Os restantes 4 documentos são rotulados com a classe "Graphs".

Considere que a fase de pré-processamento da coleção apresentada na Tabela IV tenha produzido a matriz de termos por documentos da Tabela V. Repare também que a fase de pré-processamento produziu $\mathcal{T}$, o léxico da coleção, ou seja, os termos utilizados na representação dos documentos. Esses termos são apresentados na primeira coluna da Tabela V. Note que, neste exemplo ilustrativo, $|\mathcal{T}| = 12$.

Outro detalhe a notar é que última linha da Tabela Tabela V apresenta a *classe* de cada documento, e que esta possui dois valores possíveis, c_1 (para denotar a classe "*Human Machine Interaction*") e c_2 (para denotar a classe "*Graphs*").

Uma vez realizado o pré-processamento da coleção de documentos original, considere que é apresentado um novo documento d, cujo título é "*Intelligent User Interfaces and*

Termo (t_j)	$P(t_j\|c_1)$	$t_j\|c_2)$
HUMAN	1	0
INTERFACE	1	0
COMPUTER	1	1
USER	0	1
SYSTEM	0	1
RESPONSE	0	1
TIME	0	1
EPS	0	0
SURVEY	0	1
TREES	0	0
GRAPH	0	0
MINORS	0	0

Tabela VI
BOLSA DE PALAVRAS CORRESPONDENTE À COLEÇÃO DA TABELA IV.

User Experience". A representação vetorial de d é dada pela Equação 26. Diante disso, pergunta-se em qual categoria enquadrar d.

$$\vec{d} = (0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)^T \quad (26)$$

Para a determinação da classe de d através do método CIB, primeiramente note que $\hat{\Pr}(c_1) = \frac{5}{9}$ e que $\hat{\Pr}(c_2) = \frac{4}{9}$. Essas são as estimativas para a distribuição de probabilidades anteriores. Os valores $\hat{\Pr}(t_j|c_i)$ (veja Equação 22) também podem ser calculados a partir da Tabela V. A Tabela VI apresenta esses valores, para cada um dos termos do léxico.

De posse das probabilidades anteriores e das probabilidades condicionais para os termos relevantes a d, podemos finalmente aplicar a Equação 25, para determinar c_{map}, a classe mais provável para d. Para isso, calculamos $\hat{\Pr}(d|c_1)$ e $\hat{\Pr}(d|c_2)$, conforme a seguir.

$$\begin{aligned}\hat{\Pr}(d|c_1) &= \log \Pr(USER|c_1) + \\ &\quad \log \Pr(INTERFACE|c_1) + \\ &\quad \log \Pr(c_1) \\ &= 0.0053\end{aligned}$$

$$\begin{aligned}\hat{\Pr}(d|c_2) &= \log \Pr(USER|c_2) + \\ &\quad \log \Pr(INTERFACE|c_2) + \\ &\quad \log \Pr(c_2) \\ &= 0.0206\end{aligned}$$

Portanto, uma vez que $\hat{\Pr}(d|c_1) > \hat{\Pr}(d|c_2)$, determinamos que $c_{map} = c_1$ e que a classe do documento é "*Human Machine Interaction*".

VI. AVALIAÇÃO DE CLASSIFICADORES

Usualmente, um classificador é definido por um conjunto de *parâmetros*. A tarefa de treinamento corresponde ao procedimento de encontrar um conjunto *adequado* de parâmetros para o classificador. Por exemplo, no *Classificador Ingênuo de Bayes* (Seção V-D) os parâmetros são as estimativas de probabilidades determinadas através dos cálculos de frequencias definidos por esse método. Já em classificadores gerados pelo método Rocchio (Seção V-A), os parâmetros são as coordenadas dos vetores protótipos determinados após a aplicação do métodos. A qualidade maior ou menor de um classificador (i.e., sua capacidade preditiva quando aplicado a novos objetos) depende diretamente da adequabilidade do conjunto de parâmetros que foi definido.

Após a escolha dos parâmetros (i.e., do treinamento) precisamos verificar a *qualidade* do classificador. Na verdade, a qualidade de um classificador pode ser medida em diferentes perspectivas. A seguir, enumeramos algumas delas:

- **Acurácia**. Corresponde à taxa de acertos do classificador, quando medida sobre vários conjuntos de documentos. Dado um conjunto de classificadores e um conjunto de documentos $\mathcal{D}$ a classificar, o classificador de maior acurácia é aquele que tem a maior taxa de acertos média quando aplicado aos documentos de $\mathcal{D}$. A acurácia é uma medida do quão efetivo este é na predição da classe de **novos exemplos**, isto é, de objetos que não foram usados para a determinação dos parâmetros do classificador.
- **Robustez**. A robustez do classificador através de diferentes conjuntos de documentos. Se um classificador é robusto, sua qualidade de classificação (considerando alguma das demais perspectivas) permanece praticamente a mesma quando este é treinado e avaliado em diferentes conjuntos de documentos.
- **Tempo de processamento**. De acordo com essa perspectiva de qualidade, podemos averiguar se o tempo de processamento permanece aceitável com o crescimento do conjunto de documentos. Aqui, há na verdade dois tempos de processamento relevantes, o de treinamento e o de classificação. O primeiro é o tempo necessário para produzir os parâmetros do classificador. Já o segundo diz respeito à quantidade de tempo necessária ao classificador (já treinado) para classificar um novo documento.

Uma vez fixada a perspectiva que deve ser considerada, faz sentido comparar a qualidade de dois ou mais classificadores. Embora existam diferentes perspectivas, na discussão desta Seção, damos mais ênfase à perspectiva da acurácia. Outra simplificação que fazemos nesta Seção é considerar apenas a avaliação de classificadores *binários* (Seção IV-B).

A adequabilidade de um conjunto de parâmetros relativamente à acurácia é determinada por algum *critério de otimização*. Um critério normalmente usado é a **taxa de erro na classificação** (*misclassification rate*), que é definida pela proporção de acertos do classificador quando aplicado a um conjunto de documentos, para cada um dos quais ele deve determinar a classe. Para o cálculo dessa taxa de erro, um passo inicial é construir uma **matriz de contingência**.

A. Tabelas de contingência

As tabelas de contingência são usadas para registar e analisar o relacionamento entre duas ou mais variáveis, normalmente de escala nominal. Como exemplo, considere que, em uma pesquisa realizada com 150 funcionários de uma empresa, estes tiveram de assinalar o sexo (variável s), masculino ("M") ou feminino ("F"), e o estado civil (variável ec) - solteiro, casado, viúvo ou divorciado. A seguinte tabela de contingência resume a informação contida na amostra:

	solteiro	casado	viúvo	divorciado	**total**
s=F	38	36	1	7	82
s=M	40	14	4	10	68
total	78	50	5	17	150

Note que as células da última linha e da última coluna da tabela (à exceção da célula na extrema direita inferior) apresentam os totais por coluna e por linha, respectivamente. Note também que a célula do canto inferior direito apresenta o total de indivíduos da amostra. Da análise da tabela acima, podemos, por exemplo, tirar as seguintes conclusões:

1) O número de indivíduos do sexo masculino e solteiros é 40
2) O número de indivíduos do sexo masculino é 68
3) O número de indivíduos viúvos é 5

No contexto da tarefa de classificação, tabelas de contingência são usadas para registar as quantidades de *falsos positivos*, *falsos negativos*, *verdadeiros positivos* e *verdadeiros negativos* durante o uso de um classificador. De forma geral, considere que conjunto de documentos $\mathcal{D}$ foi usado para construir um classificador binário. Suponha, além disso, que os valores das classes sejam 0 e 1. Podemos montar uma tabela de contingência da seguinte forma:

	$e=1$	$e=0$
$c=1$	N_{11}	N_{01}
$c=0$	N_{10}	N_{00}

Note que esta tabela é composta de quatro entradas denotadas por N_{00}, N_{01}, N_{10} e N_{11}, que correspondem a valores inteiros. Note também que $N_{00} + N_{01} + N_{10} + N_{11} = |\mathcal{D}|$. As quantidades N_{00}, N_{01}, N_{10} e N_{11} possuem os significados descritos a seguir:

- N_{00} é a quantidade de registros do conjunto que foi classificado como pertencentes à classe 0 e que realmente pertence a essa classe. Isto é, N_{00} é a quantidade de *verdadeiros negativos*.
- N_{01} é a quantidade de registros do conjunto que foi classificado como pertencentes à classe 0 e que na verdade pertencem à classe 0. Isto é, N_{00} é a quantidade de *falsos negativos*.
- N_{10} é a quantidade de registros do conjunto que foi classificado como pertencentes à classe 1 e que na verdade pertencem à classe 0. Isto é, N_{00} é a quantidade de *falsos positivos*.
- N_{11} é a quantidade de registros do conjunto que foi classificado como pertencentes à classe 1 e que realmente pertence a essa classe. Isto é, N_{11} é a quantidade de *verdadeiros positivos*.

O cálculo da acurácia de um classificador pode ser realizado através dos valores da tabela de contingência, conforme descrito na próxima Seção.

B. Acurácia e taxa de erro

No contexto da tarefa de classificação, a avaliação mais básica possível é calcular a acurácia do classificador através da fórmula apresentada na Equação 27.

$$\text{Acurácia} = \frac{N_{00} + N_{11}}{N_{11} + N_{01} + N_{10} + N_{00}} \tag{27}$$

A **taxa de erro** (*mis-classification rate*) de um classificador é o complemento da acurácia e mede o número de predições incorretamente feitas. A taxa de erro é dada pela Equação 28.

$$\text{Taxa de Erro} = 1 - \text{Acurácia} = \frac{N_{01} + N_{10}}{N_{11} + N_{01} + N_{10} + N_{00}} \tag{28}$$

Há três tipos de taxa de erro, conforme a descrição abaixo.

- A **taxa erro verdadeira** (*True Error Rate*, TrER) é definida como a taxa de erro do classificador quando aplicado a um conjunto de dados assintoticamente grande (i.e., que converge, no limite, para a distribuição de valores real da variáveis envolvidas) de exemplos não usados durante o treinamento.
- A **taxa de erro aparente** (*Apparent Error Rate*, AER) é a calculada considerando-se o conjunto de treinamento.
- A **taxa de erro de teste** (*Teste Error Rate*, TER) é a calculada considerando-se o conjunto de treinamento.

Na prática, a TrER não pode ser calculada, já que não dispomos de um conjunto de documentos infinitamente grande. De fato, a AER e a TER podem ser vistas como estimativas do valor teórico TrER. Já a AER, embora possa ser calculada facilmente, não tem utilidade prática, pois é obtida considerando documentos que foram usados para gerar o classificador e, sendo assim, é uma aproximação otimista da TrER. Em situações práticas, a TER é o valor usado para determinar o poder de predição do classificador. Resultados práticos mostram que a TER é uma boa estimativa para a TrER quando o número de elementos em $\mathcal{D}_{teste}$ é da ordem de 10^3 ou maior. (Veja as notas bibliográficas ao final deste capítulo.)

C. Métodos de particionamento

Uma prática normalmente seguida na tarefa de classificação é partir o conjunto de documentos original $\mathcal{D}$ em dois conjuntos disjuntos, $\mathcal{D}_{treino}$ e $\mathcal{D}_{teste}$. O primeiro é o *conjunto de treinamento* e corresponde aos documentos usados para gerar o classificador. O segundo é o *conjunto de teste* e é usado na fase de validação do classificador. Este conjunto de teste é formado pelos elementos de $\mathcal{D}$ não utilizados durante a geração do classificador. Uma vez gerado, o classificador é aplicado a cada elemento de $\mathcal{D}_{teste}$ e uma matriz de contingência é montada com as quantidades de falsos positivos, verdadeiros positivos, falsos negativos e verdadeiros negativos. Por fim, essa tabela de contingência é usada para o cálculo da taxa de erro. É comum usar o termo *experimento* para denominar a atividade de cálculo da taxa de erro sobre um conjunto de documentos.

Uma questão importante diz respeito ao procedimento utilizado para dividir o conjunto de dados original em **conjunto de teste** e **conjunto de treinamento**. Os principais métodos alternativos usados no particionamento (divisão) de um conjunto de documentos original são:

- **Amostragem Aleatória Simples** (*Hold Out Method* ou *Simple Random Sampling*),
- **Validação Cruzada** (*Cross Validation*),
- **Bootstrap**.

Em [15], Ron kohavi apresenta um estudo comparativo dos métodos *Validação Cruzada* (Seção VI-C2) e *Boostrap* (Seção VI-C3) na determinação da acurácia de classificadores. Neste mesmo trabalho, o autor apresenta resultados de experimentos computacionais que o método mais adequado para determinar a acurácia de um classificador é a Validação Cruzada com valor de $k = 10$, mesmo em casos nos quais haja poder computacional para usar valores maiores desse parâmetro. A seguir, descrevemos apresentamos mais detalhes sobre cada um desse métodos de particionamento.

1) Amostragem Aleatória Simples: Na aplicação deste método, mantemos parte do conjunto de documentos original em separado para ser usado na validação (conjunto de teste) e utilizamos a parte restante para treinar o classificador (conjunto de treinamento). Note portanto que esse método gera um único conjunto de teste (e um único conjunto de treinamento correspondente) para avaliar a acurácia do classificador.

A proporção normalmente utilizada na prática é usar $2/3$ dos documentos em $\mathcal{D}$ para formar $\mathcal{D}_{treino}$ e os restantes $1/3$ para formar $\mathcal{D}_{teste}$. De forma geral, a escolha da partição à qual um documento de $\mathcal{D}$ deve pertencer é feita de forma aleatória, seguindo as probabilidades p (de o documento pertencer a $\mathcal{D}_{treino}$) e $1 - p$ (de o documento pertencer a $\mathcal{D}_{teste}$).

Uma desvantagem desse método de particionamento é que ele reduz sobremaneira a quantidade de exemplos que se pode utilizar durante o treinamento do classificador. Isso normalmente é um problema se a quantidade de documentos no conjunto original for pequena. Nesse caso, os parâmetros do classificador resultante podem estar muito distantes da configuração ótima, o que compromete sua aplicabilidade.

2) Validação Cruzada: Diferentemente da *Amostragem Aleatória Simples*, a estratégia do método *Validação Cruzada* é gerar t ($t > 1$) conjuntos de teste e, para cada um deles, calcular a taxa de erro correspondente. A seguir, a **taxa de erro média** (*average error rate*) de todos os t experimentos é calculada.

Há duas variantes do método Validação Cruzada, conforme descrevemos a seguir.

1) A primeira variante é denominada ***leave-one-out cross validation***. Essa variante usa um único exemplo para teste a cada experimento, e os demais para treinamento. Sendo assim, a quantidade de experimentos realizados é igual à cardinalidade do conjunto $\mathcal{D}$.
2) A segunda variante é o método ***k-fold cross validation***. Nessa variante, a estratégia usada é dividir (partir) o conjunto $\mathcal{D}$ em k partes aproximadamente iguais. A seguir, são realizados k experimentos. Cada experimento usa uma parte como conjunto de teste e as demais partes $k - 1$ para formar o conjunto de treinamento.

Note que a variante *k-fold cross validation* é equivalente à *leave-one-out cross validation* quando o valor de k é igual à quantidade de itens em $\mathcal{D}$. Quando a Validação Cruzada é aplicada, um valor normalmente usado para o parâmetro k é 10.

A vantagem do método *Validação Cruzada* é que ele produz valores mais confiáveis para a taxa de erro de um classificador. Entretanto, a quantidade de tempo necessária para realização dos experimentos é maior que a necessária na *Amostragem Aleatória Simples*.

3) Bootstrap: Tanto na *Amostragem Aleatória Simples* quanto na *Validação Cruzada*, os itens que devem formar o conjunto de teste são selecionados aleatoriamente a partir do conjunto de documentos original. Além disso, o processo de seleção aleatória é *sem reposição*. Isso quer dizer que, quando um documento é selecionado para compor o conjunto de teste, ele não pode ser selecionado outra vez.

No método Bootstrap, cada conjunto de treinamento é construído através da *seleção aleatória com reposição* a partir do conjunto $\mathcal{D}$. A seleção aleatória com reposição significa que, se um item do conjunto $\mathcal{D}$ é selecionado para compor $\mathcal{D}_{treino}$, esse mesmo item pode ser novamente selecionado. É possível provar que a probabilidade de um item ser selecionado ao menos uma vez nesse processo de amostragem é $1 - (1 - 1/|\mathcal{D}|)^{|\mathcal{D}|}$. Além disso, quando $|\mathcal{D}|$ é um valor suficientemente grande, essa probabilidade se aproxima assintoticamente do valor $0,632$.

Sendo assim, cada conjunto de treinamento (também chamado de *amostra bootstrap*) possui aproximadamente 63% dos documentos do conjunto original. Note que esse conjunto de treinamento pode conter *mais de uma ocorrência* do mesmo documento obtido a partir de $\mathcal{D}$. Os documentos que não são selecionados nesse processo de amostragem formam o conjunto de teste. O classificador gerado com o conjunto de treinamento é então avaliado com o conjunto de teste para obter a acurácia na amostra.

No método Bootstrap, o procedimento de amostragem com reposição descrito acima é realizado b vezes, o que resulta em b amostras e, para cada uma, um valor para a acurácia do classificador. Esses b valores podem ser combinados de diferentes maneiras para gerar um único valor de acurácia. Uma das formas de combinação mais usadas é fazer uma média ponderada entre cada acurácia acr_i ($1 \leq i \leq b$) obtida em cada amostra e a acurácia acr obtida considerando todo o conjunto de dados original como conjnto de teste, conforme a Equação 29.

$$acr_{bootstrap} = \frac{1}{b} \sum_{i=1}^{b} (0,632 \times acr_i + 0,368 \times acr) \quad (29)$$

O valor b (isto é, a quantidade de amostras) é um parâmetro do método *Bootstrap*. De forma geral, quanto maior esse valor, mais confiável o valor da acurácia $\text{Acurácia}_{bootstrap}$. Entretanto, a quantidade de processamento necessária também deve ser levada em consideração. A quantidade de amostras necessárias também aumenta com a quantidade de classes envolvidas no problemas de classificação.

VII. Conclusões

Neste artigo, apresentamos uma introdução á tarefa de classificação em *Text Mining*. Apresentamos a terminologia usada nesta área de pesquisa. Descrevemos também diversos algoritmos que podem ser usados para a geração de um modelo de classificação. Por fim, apresentamos diversas alternativas de avaliar a qualidade dos modelos de classificação gerados.

O leitor pode encontrar mais detalehs sobre os assuntos aqui tratados nas referências.

APÊNDICE
MEDIDAS DE SIMILARIDADE

Há diversas medidas propostas para determinar a similaridade entre dois documentos quando os pesos possuem domínio contínuo. Nesse contexto, medidas geométricas de distância entre os vetores, ou do ângulo formado entre os mesmos, podem ser utilizadas para determinar a similaridade entre os documentos. O restante desta seção descreve as seguintes medidas para cálculo de similaridade: *distâncias de Minkowski*, *similaridade por cosseno* e *distâncias de Mahalanobis*.

Distâncias de Minkowski. As denominadas distâncias de Minkowski são medidas bastante utilizadas em problemas onde os objetos podem ser representados sob um perspectiva geométrica (como é o caso do modelo de espaço vetorial). A forma geral dessa família de distâncias é a seguinte:

$$L_p(\vec{x}_a, \vec{x}_b) = (\sum_{i=1}^{d} |x_{ia} - x_{ib}|^p)^{\frac{1}{p}} \quad (30)$$

A Equação 30 na verdade é uma expressão genérica a partir da qual podem ser derivada diversas méticas popularmente usadas. Primeiramente, para o valor de $p = 1$, tem-se a *distância Manhattan* (*Manhattan distance*) entre dois vetores, apresentada na Equação 31.

$$L_1(\vec{x}_a, \vec{x}_b) = \sum_{i=1}^{n} |x_{ia} - x_{ib}| \quad (31)$$

Para o valor de $p = 2$ na Equação 30, obtemos outro membro imporntante da família de distâncias de Minkowski, a *distância euclidiana* (*euclidean distance*), apresentada na Equação 32.

$$L_2(\vec{x}_a, \vec{x}_b) = \sqrt{\sum_{i=1}^{n} (x_{ia} - x_{ib})^2} \quad (32)$$

Distâncias de Mahalanobis. Da mesma forma que as distâncias de Minkowski, as distâncias de Mahalanobis também formam uma família de distâncias entre dois vetores $\vec{x_1}$ e $\vec{x_2}$ em um espaço n-dimensional. A expressão geral das métricas pertencentes à família de distâncias de Mahalanobis é dada pela Equação 33.

$$\begin{aligned} M(\vec{x_1}, \vec{x_2}) &= \sqrt{(\vec{x_1} - \vec{x_2})^t Cov^{-1} (\vec{x_1} - \vec{x_2})} \\ &= \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_{ij}(x_{1i} - x_{2i})(x_{1i} - x_{2i})} \end{aligned} \quad (33)$$

Na Equação 33, Cov é a **matriz de covariância** de x_1 e x_2. Note que a distância de Mahalanobis é equivalente à distância euclidiana se a matriz de covariância é a matriz identidade I. Note também que, quando Cov é uma matriz diagonal, a distância de Mahalanobis pode ser vista como uma generalização da distância euclidiana. Essa generalização permite considerar correlações entre as características dos objetos. Note ainda que, da mesma forma que o parâmetro p (Equação 30) parametriza uma família de distâncias de Minkowski, a matriz Cov parametriza uma família de distâncias de Mahalanobis.

Distância por cosseno. Medidas baseadas nas distâncias de Minkowski e de Mahalanobis não se aplicam adequadamente a algumas tarefas de MDT, principalmente pelo fato de a dimensionalidade do espaço de termos ser grande. Para objetos representados em um espaço de grande dimensionalidade (como é o caso de documentos de um corpus), há um padrão de fato para medir a proximidade, a **distância por cosseno** [16]. O cálculo dessa medida produz o cosseno do ângulo entre dois vetores. A similaridade por cosseno é dada pela Equação 34.

$$cos(\vec{x}_1, \vec{x}_2) = \frac{\vec{x}_1^T \cdot \vec{x}_2}{||\vec{x}_1|| \times ||\vec{x}_2||} = \frac{\sum_{i=1}^{n}(x_{i1} \times x_{i2})^2}{\sqrt{\sum_{i=1}^{n}(x_{i1})^2}\sqrt{\sum_{i=1}^{n}(x_{i2})^2}} \quad (34)$$

Na Equação 34, $\vec{x}_1^T$ representa a aplicação de transposição aplicada ao vetor $\vec{x}_1$, e $||\vec{x}_j||$ representa a magnitude (ou norma) do vetor $\vec{x}_j$. Note que, se as componentes dos vetores $\vec{x}_1$ e $\vec{x}_2$ são todas positivas (como é o caso no VSM; veja a Seção III-B), a distância por cosseno entre esses vetores pode variar na faixa dentro do conjunto $[0, 1]$. O valor 1 indica que os vetores são exatamente iguais (a menos de sua magnitude); o valor 0 indica que os vetores são perpendiculares entre si. No primeiro caso (i.e., $cos(\vec{x}_1, \vec{x}_2) = 1$), a similaridade é máxima (ou, equivalentemente, a proximidade é mínima). No segundo caso, (i.e., $cos(\vec{x}_1, \vec{x}_2) = 0$), a similaridade é mínima (ou, equivalentemente, a proximidade é máxima).

REFERÊNCIAS

[1] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, pp. 372–390, may/Jun 2000.

[2] U. Fayyad, G. Piatetsky-shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, pp. 37–54, 1996.

[3] S. M. Weiss and N. Indurkhya, *Predictive Data Mining: A Practical Guide*. San Francisco: Morgan Kaufmann, 1998.

[4] R. J. Henery, "Classification," pp. 6–16, 1994.

[5] S. Robertson and K. S. Jones, "Relevance weighting of search terms," *Journal of the American Society for Information Science*, vol. 27, no. 3, pp. 129–146, 1976.

[6] J. H. Lee, "Properties of extended boolean models in information retrieval," in *SIGIR'94: 17th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: Springer-Verlag New York, Inc., 1994, pp. 182–190.

[7] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, November 1975. [Online]. Available: http://dx.doi.org/10.1145/361219.361220

[8] G. Salton, "A document retrieval system for man-machine interaction," in *Proceedings of the 1964 19th ACM national conference*. New York, NY, USA: ACM, 1964, pp. 122.301–122.3020.

[9] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1/2, pp. 143–175, 2001.

[10] R. Linden, "Técnicas de agrupamento," *Revista de Sistemas de Informação da FSMA*, vol. 4, pp. 18–36, 2009.

[11] Y. Yang and J. Wilbur, "Using corpus statistics to remove redundant words in text categorization," *J. Am. Soc. Inf. Sci.*, vol. 47, no. 5, pp. 357–369, 1996.

[12] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[13] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.
[14] H. Zhang, "The optimality of naive bayes." in *FLAIRS Conference*, V. Barr and Z. Markov, Eds. AAAI Press, 2004. [Online]. Available: http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf
[15] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection." Morgan Kaufmann, 1995, pp. 1137–1143.
[16] G. Salton, *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. New York: Addison-Wesley Publishing Company, Inc., 1988.

Eduardo Bezerra possui grafuação em Ciência da Computação pela UFRJ (1995), mestrado em Engenharia de Sistemas e Computação pela COPPE/UFRJ (1999). Em janeiro de 2006, adquiriu o título de doutor em Engenharia de Sistemas e Computação pela COPPE/UFRJ. Atualmente, trabalha como professor do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ). É atual coordenador do Curso Superior de Tecnologia em Sistemas para Internet (http://info.cefet-rj.br/). É professor colaborador do PPTEC (Pós-Graduação Stricto Sensu em Tecnologia) no CEFET/RJ. Seus interesses de pesquisa incluem recuperação de informações, *text mining* e inteligência computacional. Pode ser encontrado no endereço ebezerra@cefet-rj.br.

Ronaldo Ribeiro Goldschmidt Ronaldo Ribeiro Goldschmidt possui graduação em Matemática pela Universidade Federal Fluminense (1988), mestrado em Sistemas e Computação pelo Instituto Militar de Engenharia (1991) e doutorado em Engenharia Elétrica - Métodos de Apoio à Decisão pela Pontifícia Universidade Católica do Rio de Janeiro (2004). Atualmente é professor titular do Instituto Superior de Tecnologia em Ciência da Computação do Rio de Janeiro (IST-Rio / FAETEC) e professor adjunto do Centro Universitário da Cidade do Rio de Janeiro. Além disso, atua como professor titular das Faculdades Integradas de Jacarepaguá, como professor pesquisador colaborador do Instituto Militar de Engenharia e como consultor na Graal Consultoria Empresarial Ltda. Tem experiência na área de Ciência da Computação, com ênfase em Inteligência Computacional, atuando principalmente nos seguintes temas: sistemas de informação, banco de dados, sistemas de apoio à decisão, mineração de dados e descoberta de conhecimento em bases de dados.