Trabalho de Formatura: Monografia

Universidade de São Paulo

Instituto de Matemática e Estatística

Bacharelado em Ciência da Computação

Recuperação de Informações por Álgebra Linear Computacional

Iniciação Científica: Julho/2003 a Dezembro/2004

Parcialmente Financiado por PIBIC/CNPq

Aluna: Ellen Hidemi Fukuda ellen at ime.usp.br

Orientador: Paulo José da Silva e Silva rsilva at ime.usp.br

Resumo

Neste trabalho abordou-se a recuperação de informações associada ao conceito de espaço vetorial. As informações são modeladas através de uma matriz e a pesquisa do usuário ao banco de dados é representada por um vetor. Deste modo, os documentos relevantes à pesquisa são identificados utilizando-se de algoritmos conhecidos da Álgebra Linear Computacional. Na primeira parte da monografia mostrou-se como os fundamentos da computação de matrizes podem ser usados para controlar e indexar grandes quantidades de texto. Na segunda parte, relacionou-se a experiência obtida no projeto de iniciação científica com o Bacharelado em Ciência da Computação.

Palavras-chaves: Fatoração QR, decomposição por valores singulares, recuperação de informações, latent semantic indexing, modelo vetorial.

Sumário

Parte I

Projeto de Iniciação Científica

Capítulo 1

Introdução

Com a evolução de bibliotecas digitais e o crescimento exponencial da quantidade de documentos disponíveis na Internet, tornaram-se necessários métodos eficazes para o armazenamento, o processamento e a recuperação de informações [BYRN99, KM00]. Tais métodos podem ser aplicados a um grande banco de dados, quando implementados em sistemas de alta performance. Um exemplo de sistema de grande escala conhecido atualmente é o Google [Goo04]. Seus usuários definem perguntas e o sistema fornece conjuntos de documentos relacionados a elas através de um processamento de dados.

Durante muitos anos, as pesquisas na área de Recuperação de Informações (IR) eram feitas por comunidade pequenas, influenciando minimamente a indústria. As primeiras instituições que adotaram um sistema de IR foram as bibliotecas. Usualmente, tais sistemas eram desenvolvidas por instituições acadêmicas interessadas em facilitar o uso de sua biblioteca. Inicialmente, os sistemas permitiam apenas pesquisas baseadas nos nomes do autores e nos títulos dos documentos, e usavam basicamente a lógica booleana [BC89].

Com o surgimento da Internet e o conseqüente interesse do meio não acadêmico nessa área, novas funcionalidades foram adicionadas e pesquisas mais complexas tornaram-se viáveis. Surgiram, assim, modelos de IR mais sofisticados, entre eles, o vetorial [BDJ99, SB88] e o probabilístico [Cro83]. Atualmente, esses modelos existentes são adaptados a fim de melhorar ao máximo a performance do sistema.

Na prática, melhorar tal performance não é simples. Problemas associados à ambigüidade da linguagem natural, aos diversos idiomas existentes e aos tipos de informações (texto, figuras, áudio e vídeo) obrigam os pesquisadores a estudarem diversas maneiras de contorná-los. Indexar grandes quantidades de dados usando recursos limitados de processamento e retornar documentos realmente relevantes às pesquisas são ainda grandes desafios.

O desenvolvimento nessa área foi consolidada pela criação, em 1992, de um evento anual internacional conhecido como *Text REtrieval Conference* (TREC) [TRE04], patrocinado por *Defense Advanced Research Projects Agency* (DARPA) e por *National Institute of Standards and Technology* (NIST) [HV96]. Os participantes do TREC competem entre si na indexação de enormes quantidades de texto e retorno de documentos mais relevantes.

Um dos sistemas de IR mais conhecidos é o SMART (*System for the Mechanical Analysis and Retrieval of Text*) [SM83], introduzido em 1983 e baseado no modelo vetorial. Tal sistema tem se sofisticado cada vez mais com a utilização de várias heurísticas [BSAS95] e continua apresentando desempenhos surpreendentes nas conferências do TREC.

O modelo vetorial, objeto de estudo desse projeto de iniciação científica, armazena informações em uma matriz, onde cada coluna representa um documento e cada linha está associada a um termo do "dicionário". A pesquisa do usuário ao banco de dados é representada por um vetor. Com isso, os documentos relevantes à pesquisa são identificados utilizando-se de conceitos simples de álgebra linear. Uma variante mais recente desse modelo é o LSI (*Latent Semantic Indexing*) [BDJ99, DDF+90], onde uma aproximação do banco de dados é usada no lugar do original. Para isso, utilizam-se algoritmos conhecidos da Álgebra Linear Computacional, em especial, a decomposição por valores singulares (SVD) [GL96].

1.1 A Realização do Trabalho

Este projeto teve como base o artigo *Matrices, Vector Spaces, and Information Retrieval* de Berry et al [BDJ99]. A partir deste, outras referências foram encontradas a fim de complementar as idéias, entre as quais destacam-se os livros-textos do Baeza-Yates e Ribeiro-Neto [BYRN99] e de Kowalski e Maybury [KM00] que permitiram um conhecimento mais sólido em Recuperação de Informações. Para um estudo mais avançado da Álgebra Linear Computacional, foi usado o livro do Golub e Van Loan [GL96], com apoio de Trefethen e Bau [TB97] e Watkins [Wat91].

Diversas outras referências foram utilizadas durante a realização da iniciação científica e serão citadas ao longo desta monografia. Nessa parte teórica do texto, indicaremos desde os conceitos mais básicos de álgebra linear até idéias e algoritmos mais sofisticados. Da mesma forma, serão mostrados os principais conceitos de Recuperação de Informações, bem como técnicas mais modernas envolvendo o modelo vetorial. Ao final, indicaremos os principais resultados dos estudos e experimentos realizados ao longo dos semestres. Tais experimentos foram realizados com o sistema SMART já mencionado, além de programas implementados em Octave [Eat02].

Capítulo 2

Conceitos Básicos e Notações

Neste capítulo teremos uma exibição de alguns conceitos de Álgebra Linear Computacional necessários para a compreensão dos próximos tópicos, bem como a indicação das notações utilizadas. Para se obter maiores detalhes em relação aos conceitos, indicamos os livros do Golub e Van Loan [GL96], Trefethen e Bau [TB97] e Watkins [Wat91]. No capítulo 1 do livro de Golub e Van Loan podemos encontrar as principais notações utilizadas nesta monografia.

2.1 Matrizes e Vetores

Utilizaremos letras maiúsculas $(A, B, ...)$ para representar *matrizes* e letras minúsculas $(a, b, ...)$ para *vetores*. A dimensão de uma matriz é $m \times n$ se ela possui m linhas e n colunas. Um *vetor linha* é uma matriz com $m = 1$, enquanto que um *vetor coluna* é uma matriz com $n = 1$. Por conveniência, mencionaremos *vetor* como sendo de coluna.

A maioria dos conceitos que utilizaremos permite ter como universo os números complexos. Visto que a Recuperação de Informações não necessita desse universo amplo, trabalharemos apenas com os números reais. Assim, $A \in \mathbb{R}^{m \times n}$ se A possui dimensão $m \times n$ e seus elementos são reais. Do mesmo modo, $v \in \mathbb{R}^n$ se v possui dimensão n e seus elementos pertencem a $\mathbb{R}$.

Cada *elemento de uma matriz* A da linha i e coluna j é denotada por a_{ij}. Além disso, cada *coluna* i *de uma matriz* A é denotada por a_i e podemos escrever $A = [a_1, ..., a_n]$. Para pseudocódigos, utilizaremos uma notação semelhante a do Matlab [Mat95] e do Octave [Eat02], duas linguagens de alto nível voltadas para computação numérica. As notações para elemento e coluna de A são, respectivamente, $A(i, j)$ e $A(:, i)$. Pode-se também utilizar $A(i, :)$ para denotar a i-ésima linha da matriz A.

Ainda em relação às partes de uma matriz, utilizaremos $A(i_1{:}i_2, j_1{:}j_2)$ para representar o bloco de uma matriz A correspondente às linhas de i_1 a i_2 e às colunas de j_1 a j_2. Se o intervalo $i_1{:}i_2$ indica todas as linhas da matriz, usaremos $A(:, j_1{:}j_2)$. Analogamente,

$A(i_1{:}i_2,:)$ indica o bloco correspondente às linhas de i_1 a i_2 e todas as colunas de A.

Cada *elemento de um vetor* $x \in \mathbb{R}^n$ é indicado por $x_1, ..., x_n$ e sua notação para pseudocódigos é dado por $x(1), ..., x(n)$. Dizemos que a *superdiagonal* de uma matriz A é o conjunto dos elementos a_{ij} tais que $j = i + 1$, para $i = 1, ..., n - 1$. Uma matriz quadrada que possui todos os elementos da diagonal iguais a 1 e as demais iguais a zero, é chamada de *identidade* e é denotada por I. Podemos ainda indicar como I_m, se sua dimensão for $m \times m$. Esta matriz é usualmente definida tendo como colunas os vetores *canônicos* $e_k \in \mathbb{R}^m$, os quais possuem todos os seus elementos nulos, exceto na posição k, onde o elemento é igual a 1. Mais especificamente, $I_m = [e_1, ..., e_m]$.

A *transposta* de uma matriz $A \in \mathbb{R}^{m \times n}$ é uma matriz $B \in \mathbb{R}^{n \times m}$ onde $b_{ij} = a_{ji}$, e é denotada por A^T (i.e., $B = A^T$). A *inversa* de A é uma matriz B tal que $AB = I$, e é indicada por A^{-1}. Temos ainda que, dados A e B com dimensões compatíveis, $(AB)^T = B^T A^T$ e $(AB)^{-1} = B^{-1}A^{-1}$. Uma matriz $D \in \mathbb{R}^{m \times n}$ é chamada diagonal se ela é quadrada (i.e., $m = n$) e se $d_{ij} = 0$ para todo $i \neq j$. Se $d_1, ..., d_n$ ($d_i = d_{ii}$) são os elementos da diagonal de D, então denotamos $D = \text{diag}(d_1, ..., d_n)$. Em particular, toda matriz identidade é diagonal.

2.2 Conceitos Básicos de Álgebra Linear

Um *espaço vetorial* V é um conjunto fechado sobre as operações de adição de vetores e multiplicação por um escalar, isto é, se dois vetores u e v pertencem a V, então $u + v$ também está em V, e se o vetor u está em V e $\lambda \in \mathbb{R}$ é um escalar qualquer, então λu também pertence a V. Um conjunto S é chamado de *subspaço* de V se ele está contido em V e é um espaço vetorial. No contexto deste trabalho consideraremos que $V = \mathbb{R}^m$.

Considere $a_1, ..., a_n$ vetores em $\mathbb{R}^m$. Uma *combinação linear* desses vetores é qualquer vetor da forma $\sum_{i=1}^{n} \lambda_i a_i$, onde λ_i são escalares chamados de *coeficientes* da combinação linear. O espaço formado por todas as combinações lineares desses vetores é definido como *espaço gerado* por $a_1, ..., a_n$ e é denotado por $\langle a_1, ..., a_n \rangle$ ou $\text{span}\{a_1, ..., a_n\}$. Se esses vetores correspondem às colunas de uma matriz A, o espaço gerado por elas pode também ser denotado por $\langle A \rangle$ ou $\text{span}\{A\}$. Além disso, esse conjunto de vetores é dito *linearmente dependente* se existirem escalares λ_i, $i = 1, ..., n$, nem todos nulos, tais que $\sum_{i=1}^{n} \lambda_i a_i = 0$. Caso contrário, ele é *linearmente independente* e, para que a relação mencionada seja satisfeita, todos os escalares λ_i devem ser iguais a zero.

Considere S como sendo o subspaço de $\mathbb{R}^m$ e os vetores $a_i \in \mathbb{R}^m$ mencionados acima. Definimos uma *base* como sendo um conjunto gerado por esses vetores que é linearmente independente. Todas as bases de S têm o mesmo número de elementos, o qual é definido como sendo sua dimensão e denotado por $\dim(S)$.

Dois importantes subspaços associados a uma matriz $A \in \mathbb{R}^{m \times n}$ são a *imagem* de A e o *espaço nulo* de A. Definimos estes subspaços, respectivamente, por $\text{Im}(A) \doteq \{y \in \mathbb{R}^m | y = Ax \text{ para algum } x \in \mathbb{R}^n\}$ e $\text{Nulo}(A) \doteq \{x \in \mathbb{R}^n | Ax = 0\}$. Observe que a $\text{Im}(A)$ é igual ao espaço gerado pelas colunas de A, i.e., $\text{Im}(A) = \langle a_1, ..., a_n \rangle$. O *posto* de A, por sua vez, é denotado por $\text{posto}(A)$ e é igual à dimensão da imagem de A. Dizemos também que o posto é *completo* ou *máximo* se $\text{posto}(A) = \min\{m, n\}$. Uma matriz é dita *não-singular* ou *inversível* se ela é quadrada e possui posto completo. Caso contrário, a matriz é dita *singular*. Pode-se mostrar também que para toda matriz A inversível, $\det(A) \neq 0$, onde $\det(A)$ é o determinante de A.

2.3 Normas, Projetores e Ortogonalidade

A *norma-2 de um vetor* $x \in \mathbb{R}^m$, conhecida também como *norma euclidiana*, é dada por $\|x\|_2 \doteq \left(\sum_{i=1}^m |x_i|^2\right)^{\frac{1}{2}}$. Se x e y são dois vetores em $\mathbb{R}^m$, então o produto interno desses dois vetores é dado por $x^T y = y^T x = \sum_{i=1}^m x_i y_i$. Note que $\|x\|_2 = \sqrt{x^T x}$. O *cosseno do ângulo* α entre x e y pode ser escrito da seguinte forma: $\cos(\alpha) = \frac{x^T y}{\|x\|_2 \|y\|_2}$. Além disso, a *norma-2 de uma matriz* A é definida como $\sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2 = 1} \|Ax\|_2$. Definimos ainda a norma de Frobenius de uma matriz A como $\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2} = \sqrt{\text{tr}(A^T A)}$, onde $\text{tr}(A^T A)$ é o traço de $A^T A$, ou seja, a soma dos elementos da diagonal de $A^T A$.

Dois vetores x e y em $\mathbb{R}^m$ são *ortogonais* se o produto interno entre eles for nulo. Dizemos ainda que um conjunto de vetores é ortogonal se todos os elementos são ortogonais entre si. Se, além disso, todos os vetores tiverem norma igual a 1, dizemos que esse conjunto de vetores é *ortonormal*. Uma matriz $Q \in \mathbb{R}^{m \times m}$ é ortogonal se $Q^T = Q^{-1}$, ou seja, $Q^T Q = I$. Nesse caso, as colunas de Q são ortogonais entre si e formam uma base ortonormal de $\mathbb{R}^m$. Observe também que $\|Q\|_2 = 1$ e que $\det(Q) = \pm 1$.

Ainda em relação à ortogonalidade, sabe-se que a multiplicação por uma matriz ortogonal tem o efeito de preservar a norma e o ângulo de vetores. Ou seja, dada uma matriz ortogonal Q e dois vetores x e y com dimensões compatíveis, temos que $(Qx)^T(Qy) = xQ^T Qy = x^T y$ e $\|Qx\|_2 = \|Q\|_2 \|x\|_2 = \|x\|_2$.

Uma matriz quadrada P é dita de *projeção* se $P^2 = P$. Observe que se $v \in \text{Im}(P)$, então a aplicação de uma projeção sobre ele resulta no próprio v. Matematicamente, temos que se $v \in \text{Im}(P)$, então $v = Px$ para algum x e $Pv = P^2 x = Px = v$. Temos ainda que se P é uma matriz de projeção, então $I - P$ é sua projeção complementar, a qual projeta no $\text{Nulo}\,(P)$ e $\text{Nulo}\,(I - P) = \text{Im}\,(P)$.

Uma matriz de projeção separa $\mathbb{R}^m$ em dois espaços S_1, o espaço no qual se projeta, e S_2, as direções por onde se projeta. Uma matriz de *projeção ortogonal* P é aquela em que S_1 e S_2 são ortogonais entre si e que satisfaz a igualdade $P = P^T$. Se as colunas de uma matriz $V = [v_1, ..., v_k]$ formam uma base ortonormal de um subspaço S, então $P = VV^T$ é uma projeção ortogonal em S. Note que se $v \in \mathbb{R}^n$, então $P = \frac{vv^T}{v^T v}$ é a projeção ortogonal em $S = \text{span}\{v\}$.

Capítulo 3

Fatoração QR

Dada uma matriz $A \in \mathbb{R}^{m\times n}$, $m \geq n$, existe uma matriz ortogonal $Q \in \mathbb{R}^{m\times m}$ e uma matriz triangular superior $R \in \mathbb{R}^{m\times n}$ tais que

$$A = QR. \tag{3.1}$$

Esta decomposição é dita *fatoração QR* de A. Vejamos, nesse capítulo, como obter essa fatoração.

Sejam $a_1, a_2, ..., a_n$ as colunas da matriz A e a seqüência de subspaços gerados por elas:

$$\langle a_1\rangle \subseteq \langle a_1, a_2\rangle \subseteq ... \subseteq \langle a_1, a_2, ..., a_n\rangle$$

Uma idéia da fatoração QR está na construção de uma seqüência de vetores $q_1, q_2, ..., q_n$ ortonormais que geram essa mesma seqüência de subespaços, ou seja, tais que:

$$\langle a_1, a_2, ..., a_i\rangle = \langle q_1, q_2, ..., q_i\rangle, \qquad i = 1, .., n.$$

Dizemos que se A possui posto máximo, então as n colunas de Q formam uma base ortonormal da imagem de A. Para que a_1 seja combinação linear de q_1, devemos ter um escalar r_{11} tal que $a_1 = r_{11}q_1$. Analisaremos agora o caso $\langle a_1, a_2\rangle = \langle q_1, q_2\rangle$. Temos que $t_1a_1 + t_2a_2 = t_3q_1 + t_4q_2$, para t_i, $i = 1, ..., 4$, escalares. Substituindo a_1 por $r_{11}q_1$, podemos escrever: $a_2 = r_{12}q_1 + r_{22}q_2$, com r_{12} e r_{22} escalares. Analogamente, temos:

$$\begin{array}{rcl} a_1 & = & r_{11}q_1, \\ a_2 & = & r_{12}q_1 + r_{22}q_2, \\ & \vdots & \\ a_n & = & r_{1n}q_1 + r_{2n}q_2 + ... + r_{nn}q_n. \end{array}$$

Observe que as equações acima podem ser escritas utilizando matrizes $\hat{Q}$ ortogonal e $\hat{R}$ triangular superior tais que $A = \hat{Q}\hat{R}$, i.e:

$$\left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \dots & a_n \\ & & & \end{array}\right] = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & \dots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix} \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \dots & q_n \\ & & & \end{array}\right]$$

Note que $\hat{Q} \in \mathbb{R}^{m \times n}$ e $\hat{R} \in \mathbb{R}^{n \times n}$. Esta é a fatoração QR *reduzida* de A. Para obtermos a fatoração QR *completa* de A (conforme enunciado no começo do capítulo), devemos adicionar $m - n$ colunas ortonormais em $\hat{Q}$, obtendo-se uma matriz Q quadrada ($m \times m$) e ortogonal. Além disso, obtemos R adicionando-se $m - n$ linhas nulas a $\hat{R}$. Desse modo as colunas adicionadas de Q serão multiplicadas por zero, obtendo-se a matriz A desejada.

A figura 3 ilustra a fatoração QR de uma matriz A. A região hachurada representa quaisquer elementos de uma matriz, enquanto que a região não hachurada representa os elementos nulos. Os retângulos tracejados mostram as colunas de Q e as linhas de R que foram adicionadas para obtermos a fatoração QR completa de A.

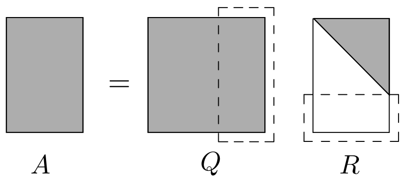


Figura 3: Fatoração QR completa de A ($m \geq n$)

Devemos observar também que as colunas q_i para $i > n$ são ortogonais à imagem de A. Considerando n igual ao posto de A, estas colunas constituem uma base ortonormal de $\mathrm{Im}(A)^{\perp}$, ou equivalentemente do $\mathrm{Nulo}(A^T)$.

Descreveremos a seguir dois métodos para computar a fatoração QR - por reflexão de Householder [GL96, Capítulo 5], [TB97, Capítulo 10] e por rotações de Givens [GL96, Capítulo 5] - e posteriormente discutiremos a fatoração QR com pivotamento de colunas [GL96, Capítulo 5.4].

3.1 Reflexão de Householder

A idéia da *reflexão de Householder* está em triangularizar a matriz A introduzindo zeros apropriadamente em cada coluna. Isto é feito obtendo-se matrizes ortogonais Q_k tais que $Q_n ... Q_2 Q_1 A$ é triangular superior. Uma matriz Q_k é escolhida de modo a zerar os elementos abaixo da diagonal principal da coluna k. Um exemplo é mostrado a seguir:

$$\begin{array}{ccccccc}
\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} & \xrightarrow{Q_1} & \begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix} & \xrightarrow{Q_2} & \begin{bmatrix} \times & \times & \times \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \mathbf{0} & \boldsymbol{\times} \\ 0 & \mathbf{0} & \boldsymbol{\times} \\ 0 & \mathbf{0} & \boldsymbol{\times} \end{bmatrix} & \xrightarrow{Q_3} & \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \boldsymbol{\times} \\ 0 & 0 & \mathbf{0} \\ 0 & 0 & \mathbf{0} \end{bmatrix} \\
A & & Q_1A & & Q_2Q_1A & & Q_3Q_2Q_1A
\end{array}$$

Cada matriz Q_k zera elementos da coluna k correspondente de maneira a não modificar as $k-1$ colunas zeradas anteriormente. No exemplo acima, os elementos em negrito são aqueles que foram modificados de uma iteração a outra pela matriz Q_k. Considerando-se isso, é natural pensarmos que Q_k possui o seguinte formato:

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix},$$

onde I é a matriz identidade $(k-1)\times(k-1)$ e F é uma matriz ortogonal $(m-k+1)\times(m-k+1)$. Observe que a matriz identidade I é, de fato, a responsável por evitar a modificação citada durante o processo de triangularização. De modo geral, uma matriz Q_k opera nas linhas $k, ..., m$ e no começo do passo k a matriz possui as $k-1$ primeiras colunas zeradas.

A matriz $F \in \mathbb{R}^{n\times n}$ deve ser então a responsável por zerar os elementos das colunas desejados. Considere que x seja o vetor formado pelos elementos abaixo da diagonal de uma coluna k. A matriz F, chamada de refletor de Householder, deverá efetuar a seguinte transformação:

$$x = \begin{bmatrix} \times \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \quad \xrightarrow{F} \quad Fx = \begin{bmatrix} \|x\|_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \|x\|_2 e_1.$$

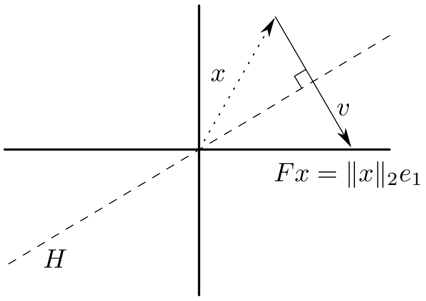


Figura 3.1.1: Reflexão de Householder

Quando um vetor x é multiplicado por F, ele é refletido em um hiperplano ortogonal a $v = \|x\|e_1 - x$. A figura 3.1.1 ilustra a idéia. Observe que cada ponto de x contido em um lado do hiperplano H é refletido na sua imagem Fx. Para deduzirmos a fórmula correspondente a F, utilizaremos o conceito de projeção, dado anteriormente no capítulo 2. Vimos, que para qualquer vetor x,

$$Px = \left(I - \frac{vv^T}{v^T v} \right) x$$

é a projeção ortogonal de x no espaço H. Para que x seja refletida no hiperplano, devemos percorrer nessa mesma direção em dobro. Assim, F difere de P simplesmente pela adição do fator 2 na fórmula:

$$Fx = \left(I - 2\frac{vv^T}{v^T v} \right) x$$

Além disso, note na figura 3.1.1 que esse método faz com que Fx seja igual a $\|x\|e_1$. No entanto, existe mais de uma reflexão possível que permite zerar os elementos desejados. A figura 3.1.2 mostra as duas possíveis reflexões através dos hiperplanos H^+ e H^-. Com isso, $Fx = s\|x\|e_1$, sendo que $s = \pm 1$.

O sinal s deve ser estabelecido de modo a minimizar os erros de cancelamento e sua conseqüente instabilidade numérica [Ove01]. Isso nos obriga a escolher o vetor Fx menos próximo a x. Quando x está no primeiro quadrante, a escolha deve ser $s = -1$. Por outro lado, tomamos $s = 1$ se x estiver no segundo quadrante. Com isso pode-se adequadamente adotar s como sendo igual ao sinal oposto ao primeiro componente de x (denotado por x_1). Portanto, temos:

$$v = \text{sign}(x_1)\|x\|_2 e_1 + x,$$

considerando-se $\text{sign}(x_1) = 1$ quando $x_1 = 0$.

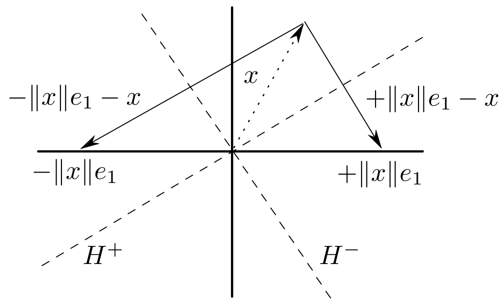


Figura 3.1.2: Escolha do vetor de reflexão

Mostraremos a seguir o algoritmo da reflexão de Householder. Conforme mencionado anteriormente, no capítulo 2, usaremos a notação semelhante a do MATLAB [Mat95] e do Octave [Eat02].

Algoritmo 3.1.1. *Dado um vetor $x \in \mathbb{R}^n$, esta função devolve $v \in \mathbb{R}^n$ tal que $I - \frac{2}{v^T v} vv^T$ é o refletor de Householder correspondente a x.*

function$[v] =$ **house**(x)
  $v = \text{sign}(x_1)\|x\|_2 e_1 + x$
  $v = v/\|v\|_2$
end

Algoritmo 3.1.2. *(Fatoração QR por reflexões de Householder) Dada uma matriz $A \in \mathbb{R}^{m\times n}$, com $m \geq n$, o algoritmo devolve $R \in \mathbb{R}^{m\times n}$ triangular superior (sobrescrito na parte triangular superior de A) e $Q \in \mathbb{R}^{m\times m}$ ortogonal tais que $A = QR$. O j-ésimo vetor de Householder é armazenado em $A(j+1{:}m, j)$, $j < m$. A função* house *usada é dada pelo algoritmo 3.1.1.*

for $k = 1{:}n$
  $v_k =$ **house**$(A(k{:}m, k))$
  $A(k{:}m, k{:}n) = A(k{:}m, k{:}n) - 2v_k(v_k^T A(k{:}m, k{:}n))$
end

Note que este algoritmo não constrói a matriz Q explicitamente. Vale indicar ainda que sua complexidade é de $2n^2(m - n/3)$ flops. A demonstração disto pode ser vista em [TB97, Capítulo 10].

Veremos agora como construir a matriz Q. Para evitar um consumo de tempo desnecessário, sua construção não será baseada na formação de todas as matrizes Q_k seguida de sucessivas multiplicações. Um método de construção implícita de Q está relacionado ao fato de que um sistema do tipo $Ax = b$ pode ser resolvido utilizando-se da fatoração QR de A.

Observe que $Ax = b \Leftrightarrow QRx = b \Leftrightarrow Rx = Q^T b$. O único momento em que foi usada a matriz Q, foi no cômputo de $Q^T b$. Como $Q^T = Q_n Q_{n-1} ... Q_1$, podemos usar as mesmas operações do processo de triangularizar a matriz A em b, isto é:

Algoritmo 3.1.3. *Cálculo implícito do produto $Q^T b$*

for $i = 1{:}n$
  $b(i{:}m) = b(i{:}m) - 2v_i(v_i^T b(i{:}m))$

Podemos, além disso, obter o cálculo implícito de Qx simplesmente executando o algoritmo acima em ordem reversa, como pode ser observado no próximo algoritmo. A notação "**for** $i = n{:}-1{:}1$" usada abaixo indica que i, inicialmente igual a n, é decrementado a cada iteração. O processo pára quando $i = 1$.

Algoritmo 3.1.4. *Cálculo implícito do produto Qx*

for $i = n{:}-1{:}1$
  $x(i{:}m) = x(i{:}m) - 2v_i(v_i^T x(i{:}m))$

Finalmente, para obtermos a matriz Q explicitamente, basta utilizarmos o algoritmo 3.1.4 acima para construir QI (onde I é a matriz identidade), calculando-se Qe_i para $1 \leq i \leq m$. Este algoritmo é dado a seguir. Sua complexidade é de $4(m^2n - mn^2 + n^3/3)$ flops [GL96, Capítulo 5.1].

Algoritmo 3.1.5. *Cálculo explícito da matriz ortogonal Q.*

$Q = I_m$
for $j = 1{:}m$
    $x = q(:, j)$
    for $i = n{:}-1{:}1$
        $x(i{:}m) = x(i{:}m) - 2v_i(v_i^T x(i{:}m))$
    end
    $q(:, j) = x$
end

3.2 Rotações de Givens

Ao contrário das reflexões de Householder, as *rotações de Givens* permitem zerar elementos de uma matriz de uma maneira mais seletiva. A cada passo do algoritmo teremos matrizes G_i responsáveis por zerar um elemento da matriz, conforme o exemplo abaixo:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{G_1} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix} \xrightarrow{G_2} \begin{bmatrix} \times & \times & \times \\ \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \times & \times \end{bmatrix} \xrightarrow{G_3}$$

$$\begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{G_4} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\times} \end{bmatrix} \xrightarrow{G_5} \begin{bmatrix} \times & \times & \times \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\times} \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{G_6} R$$

Com isso, obtém-se R triangular superior tal que $Q^T A = R$, onde Q é obtida através das chamadas rotações de Givens, isto é, $Q = G_1 ... G_t$, com t sendo o número de rotações aplicadas durante o processo. Para entendermos qual o formato de cada matriz G_k, considere inicialmente uma matriz $W \in \mathbb{R}^{2\times 2}$ que possui a seguinte forma:

$$W = \begin{bmatrix} \cos(\theta) & \operatorname{sen}(\theta) \\ -\operatorname{sen}(\theta) & \cos(\theta) \end{bmatrix} \tag{3.2}$$

Seja $x \in \mathbb{R}^2$ um vetor qualquer. Se $y = W^T x$, então, claramente, y corresponde ao vetor x rotacionado de um ângulo θ (no plano do $\mathbb{R}^2$) no sentido anti-horário, conforme figura 3.2.1. Observe que podemos escolher θ apropriadamente de modo que y seja igual a $(\|x\|_2, 0)^T$.

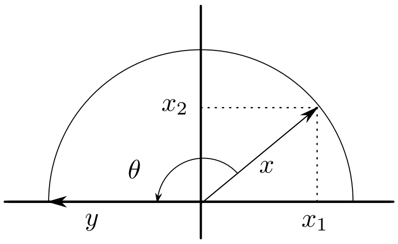


Figura 3.2.1: Rotações de Givens: $\mathbb{R}^2$

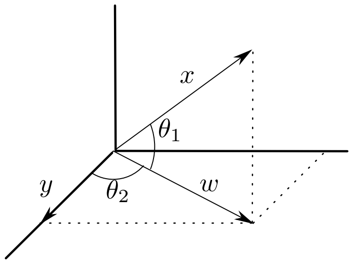


Figura 3.2.2: Rotações de Givens: $\mathbb{R}^3$

Sabemos então como obter, a partir de um vetor de dimensão 2 e de uma transformação ortogonal, um outro com a mesma dimensão e a segunda coordenada nula. Considere agora um vetor $x \in \mathbb{R}^3$. Queremos, a partir do vetor x, obter um vetor da forma $(\|x\|_2, 0, 0)^T$. Para isso, podemos usar duas rotações: a primeira rotaciona x de uma ângulo θ_1 em um plano de coordenadas de modo a obter $w = (w_1, w_2, 0)^T$, para algum w_1 e w_2; e a segunda rotaciona esse w de um ângulo θ_2 em um outro plano de modo a obter $y = (\|x\|_2, 0, 0)^T$ (veja figura 3.2.2).

De modo geral, um vetor $x \in \mathbb{R}^n$ terá seus $n-1$ elementos zerados utilizando-se de $n-1$ rotações. Como cada rotação está associada a um plano de coordenadas (i,k), isto é, a apenas dois eixos, basta colocarmos os elementos da matriz 3.2 nas posições (i,i), (i,k), (k,i) e (k,k). Para desconsiderarmos os outros eixos, utilizamos a matriz identidade. Desse modo, cada rotação de Given tem o seguinte formato:

$$
G(i,k,\theta) = \begin{array}{cc}
\begin{bmatrix}
1 & \dots & 0 & \dots & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & & \vdots & & \vdots \\
0 & \dots & c & \dots & s & \dots & 0 \\
\vdots & & \vdots & \ddots & \vdots & & \vdots \\
0 & \dots & -s & \dots & c & \dots & 0 \\
\vdots & & \vdots & & \vdots & \ddots & \vdots \\
0 & \dots & 0 & \dots & 0 & \dots & 1
\end{bmatrix}
&
\begin{array}{c}
\\ \\ i \\ \\ k \\ \\ \\
\end{array}
\\
\begin{array}{ccccccc}
 & & i & & k & &
\end{array}
&
\end{array}
$$

onde $c = \cos(\theta)$ e $s = \operatorname{sen}(\theta)$ para algum θ. Note que rotações de Givens são ortogonais. Como o produto de matrizes ortogonais é ortogonal, temos também que $Q = G_1 ... G_t$ é ortogonal. Observe ainda que cada rotação modifica apenas duas linhas da matriz, correspondentes aos dois eixos associados ao plano. Isto pode ser observado no exemplo dado no começo desta secção, para cada iteração, em negrito.

Multiplicações à esquerda por $G(i,k,\theta)^T$ geram uma rotação em sentido anti-horário de θ radianos no plano de coordenadas (i,k). Seja $x \in \mathbb{R}^n$ e $y = G(i,k,\theta)^T x$. Então:

$$y_j = \begin{cases} cx_i - sx_k, & \text{se } j = i \\ sx_i + cx_k, & \text{se } j = k \\ x_j, & \text{caso contrário} \end{cases}$$

Além disso, para que y_k seja igual a zero, devemos ter:

$$c = \frac{x_i}{\sqrt{x_i^2 + x_k^2}} \qquad e \qquad s = \frac{-x_k}{\sqrt{x_i^2 + x_k^2}}.$$

Baseado-se nesses fatos, chega-se aos seguintes algoritmos:

Algoritmo 3.2.1. *Dados escalares a e b, a função* givens *calcula $c = cos(\theta)$ e $s = sen(\theta)$ tal que:*

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}.$$

function$[c,s] =$ **givens**(a,b)
  if $b = 0$
    $c = 1; s = 0$
  else
    if $|b| > |a|$
      $r = -a/b; s = 1/\sqrt{1+r^2}; c = sr$
    else
      $r = -b/a; c = 1/\sqrt{1+r^2}; s = cr$
    end
  end
end

Algoritmo 3.2.2. *(Fatoração QR por rotações de Givens) Dada uma matriz $A \in \mathbb{R}^{m\times n}$, com $m \geq n$, o algoritmo calcula R triangular superior e Q ortogonal tais que $A = QR$.*

for $j = 1{:}n$
  for $i = m{:}-1{:}j+1$
    $[c,s] =$ **givens**$(A(i-1,j), A(i,j))$
    $A(i-1{:}i, j{:}n) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T A(i-1{:}i, j{:}n)$
  end

end

Observe que o algoritmo 3.2.1 requer 5 flops e uma única raiz quadrada. O algoritmo 3.2.2, por sua vez, tem o custo de $3n^2(m - n/3)$ flops conforme pode ser visto em [GL96, Capítulo 5.2].

3.3 Fatoração QR com Pivotamento

Se uma matriz $A \in \mathbb{R}^{m \times n}$ e posto$(A) < n$, então a fatoração QR de A não necessariamente produz uma base ortonormal para a Im(A). Considere como exemplo a matriz abaixo e sua fatoração QR usual:

$$A = \left[\, a_1,\, a_2,\, a_3 \,\right] = \left[\, q_1,\, q_2,\, q_3 \,\right] \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Claramente, posto$(A) = 2$, mas a imagem de A não é equivalente a nenhum dos subspaços span$\{q_1, q_2\}$, span$\{q_1, q_3\}$ ou span$\{q_2, q_3\}$. Descrevemos, então, como esse problema pode ser resolvido calculando-se a fatoração QR de A com suas colunas permutadas (i.e., com *pivotamento de colunas*) [GB65]. Tal fatoração é dada por $AP = QR$, onde Q e R são matrizes usuais de uma fatoração QR e P é uma matriz de permutação.

Seja r_A o posto de A, com $r_A < n$. Sabemos que a imagem de A é o espaço gerado por r_A colunas de A definidas como $a_{c_1}, a_{c_2}, ..., a_{c_{r_A}}$. Considere a matriz cujas primeiras colunas de A são exatamente essas colunas. Para obtermos essa matriz, multiplicamos A por uma matriz P (chamada matriz de permutação), que é a matriz identidade com colunas adequadamente trocadas. Temos, portanto:

$$AP = \left[\, a_{c_1},\, \ldots,\, a_{c_{r_A}},\, \ldots,\, a_{c_n} \,\right].$$

Sabemos que Im$(A) = \langle a_{c_1}, a_{c_2}, ..., a_{c_{r_A}} \rangle$ e que uma idéia da fatoração QR está em construir vetores ortonormais $q_1, ..., q_{r_A}$ tais que $\langle a_{c_1}, a_{c_2}, ..., a_{c_k} \rangle = \langle q_1, q_2, ..., q_k \rangle$ para $k = 1, .., r_A$. Uma idéia natural seria tentarmos obter uma fatoração tal que

$$Q = \left[\, q_1,\, \ldots,\, q_{r_A},\, q_{r_A+1},\, \ldots,\, q_m \,\right] \qquad \text{e} \qquad R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}.$$

onde $R_{11} \in \mathbb{R}^{r_A \times r_A}$ é triagonal superior e não-singular, e $R_{12} \in \mathbb{R}^{r_A \times (n - r_A)}$. Note que com isso teremos uma matriz resultante QR com posto igual a r_A e imagem igual a $\langle q_1, q_2, ..., q_{r_A} \rangle$. Isto significa que Im$(A) = $ span$\{q_1, ..., q_{r_A}\}$. Observando também que

$$a_{c_k} = \sum_{i=1}^{\min\{r_A, k\}} r_{ik} q_i \in \text{span}\{q_1, ..., q_{r_A}\},$$

para $k = 1{:}n$, podemos concluir que $AP = QR$.

Mostremos agora que uma modificação simples no algoritmo do Householder é o suficiente para obtermos a fatoração $AP = QR$. Considere que no início de uma iteração k, temos as matrizes ortogonais $Q_1, ..., Q_{k-1}$ e as de permutação $P_1, ..., P_{k-1}$ tais que

$$(Q_{k-1}...Q_1)A(P_1...P_{k-1}) = R^{(k-1)} = \begin{bmatrix} R_{11}^{(k-1)} & R_{12}^{(k-1)} \\ 0 & R_{22}^{(k-1)} \end{bmatrix}$$

onde $R_{11}^{(k-1)}$ é uma matriz de dimensão $(k-1) \times (k-1)$, não-singular e triangular superior e $R_{22}^{(k-1)}$ é uma matriz de dimensão $(m-k+1) \times (n-k+1)$.

Considere que $R_{22}^{(k-1)} = \left[z_k^{(k-1)}, ..., z_n^{(k-1)}\right]$. A idéia é, basicamente, mover a coluna de $R_{22}^{(k-1)}$ de maior norma para a posição corrente e zerar os elementos desejados, como no Householder usual. Em outras palavras, devemos procurar um índice p tal que $k \leq p \leq n$ e

$$\|z_p^{(k-1)}\|_2 = \max\left\{\|z_k^{(k-1)}\|_2, ..., \|z_n^{(k-1)}\|_2\right\}.$$

Em seguida, considerando P_k a matriz identidade $n \times n$ com as colunas p e k trocadas, determinamos a matriz Q_k tal que $R^{(k)} = Q_k R^{(k-1)} P_k$ possui todos os elementos da coluna k e linhas de $k+1$ a m iguais a zero. Por fim, temos que $Q = Q_1...Q_{r_A}$ e $P = P_1...P_{r_A}$.

Observe que a escolha do índice p exige o cômputo das normas das colunas de $R_{22}^{(k-1)}$ para cada iteração. Sabemos, no entanto, que a multiplicação por uma matriz ortogonal preserva a norma. Desse modo, temos que

$$Q^T z^{(j-1)} = \begin{array}{cc} \begin{bmatrix} r_{kj} \\ z^{(j)} \end{bmatrix} & \begin{array}{c} 1 \\ j-1 \end{array} \end{array} \qquad \Longrightarrow \qquad \|z^{(j)}\|_2^2 = \|z^{(j-1)}\|_2^2 - r_{kj}^2$$

ou seja, obtemos a nova norma da coluna a partir da norma anterior. Baseando-se na discussão acima, obtemos o algoritmo a seguir:

Algoritmo 3.3.1. *(Fatoração QR com Pivotamento de Colunas) Dada uma matriz* $A \in \mathbb{R}^{m \times n}$*, com* $m \geq n$*, o algoritmo computa uma matriz ortogonal* $Q = Q_1...Q_{r_A}$*, uma matriz de permutação* $P = P_1..P_{r_A}$ *e uma matriz triangular superior* R *sobrescrita na parte triangular superior de* A *tais que* $AP = QR$*. A matriz* Q *não é dada explicitamente, mas o j-ésimo vetor de Householder é armazenada em* $A(j+1{:}m, j)$*. A permutação* P*, por sua vez, é obtida através do vetor de inteiros* piv*, sendo que* P_i *é a matriz identidade com as linhas* i *e* piv(i) *trocadas. A função* house *é dada pelo algoritmo 3.1.1.*

for $j = 1{:}n$
  $c(j) = A(1{:}m, j)^T A(1{:}m, j)$
end
$r = 0; \tau = \max\{c(1), ..., c(n)\}$
Encontrar menor k, com $1 \leq k \leq n$ tal que $c(k) = \tau$

while $\tau > 0$
    $r = r + 1$
    $piv(r) = k; A(1{:}m, r) \leftrightarrow A(1{:}m, k); c(r) \leftrightarrow c(k)$
    $v =$ **house**$(A(r{:}m, r))$
    $A(r{:}m, r{:}n) = A(r{:}m, r{:}n) - 2v(v^T A(r{:}m, r{:}n))$
    $A(r + 1{:}m, r) = v(2{:}m - r + 1)$
    for $i = r + 1{:}n$
        $c(i) = c(i) - A(r, i)^2$
    end
    if $r < n$
        $\tau = \max\{c(r + 1), ..., c(n))\}$
        Encontrar menor k com $r + 1 \leq k \leq n$ tal que $c(k) = \tau$.
    else
        $\tau = 0$
    end
end

A complexidade do algoritmo é de $4mnr_A - 2r_A^2(m+n) + 4r_A^3/3$ flops, onde r_A é o posto da matriz A [GL96, Capítulo 5.4].

Capítulo 4

Decomposição SVD

Dada uma matriz $A \in \mathbb{R}^{m \times n}$, existem matrizes ortogonais $U \in \mathbb{R}^{m \times m}$ e $V \in \mathbb{R}^{n \times n}$ e uma matriz diagonal $\Sigma \in \mathbb{R}^{m \times n}$ tais que

$$A = U\Sigma V^T. \tag{4.1}$$

Esta fatoração é única e é dita *decomposição por valores singulares* (ou SVD) de A. A demonstração da existência e da unicidade de tal decomposição de matriz pode ser vista em [TB97, Capítulo 4] e [GL96, Capítulo 2.5]. Mencionaremos a seguir a sua idéia e suas propriedades, bem como a maneira de computá-la.

Considerando o universo dos números reais, a existência do SVD está associada ao fato de que a imagem de uma esfera unitária sob uma matriz $m \times n$ é uma *hiperelipse* no $\mathbb{R}^m$. Esta, por sua vez, se refere a uma generalização de uma elipse. Ela pode ser definida por vetores ortonormais $u_1, u_2, ..., u_n \in \mathbb{R}^m$ e escalares $\sigma_1, \sigma_2, ..., \sigma_n$ tais que $\sigma_i u_i$ são os semi-eixos de comprimento σ_i, $i = 1, ..., n$. Do mesmo modo, uma esfera unitária pode ser definida por vetores ortonormais $v_1, v_2, ..., v_n \in \mathbb{R}^n$.

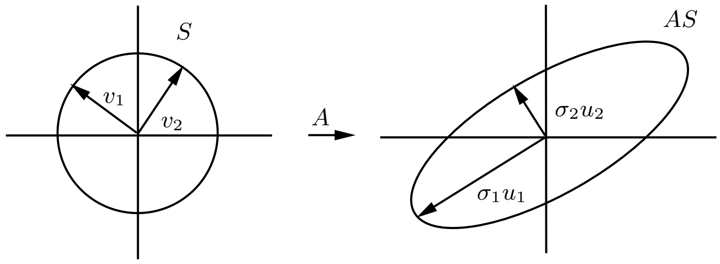


Figura 4.1: SVD de $A \in \mathbb{R}^2$.

A figura 4.1 mostra a transformação da esfera unitária S em um espaço n-dimensional por A. Considere AS como a imagem dada por tal transformação. Definiremos agora algumas propriedades da matriz A em termos de AS. Considere ainda que A tem posto máximo.

Os comprimentos dos semi-eixos de AS, dados por $\sigma_1, \sigma_2, ..., \sigma_n$, são ditos *valores singulares* de A. É conveniente defini-los de forma a estarem em ordem decrescente isto é, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$. Além disso, define-se os *vetores singulares à esquerda* como os vetores unitários $u_1, u_2, ..., u_n$ e os *vetores singulares à direita* como $v_1, v_2, ..., v_n \in S$, também unitários. Observe que desse modo, $Av_i = \sigma_i u_i$, $i = 1, ..., n$.

Estas n equações podem ser escritas como $AV = U\Sigma$, ou, equivalentemente, $A = U\Sigma V^{-1}$, onde U é a matriz com colunas u_i, V é a matriz com colunas v_i e Σ é diagonal, contendo os valores singulares σ_i em ordem crescente. Da maneira que definimos v_i, temos, claramente, que V é ortogonal. Portanto, podemos escrever: $A = U\Sigma V^T$.

Esta é a decomposição SVD *reduzida*. Note que $U \in \mathbb{R}^{m\times n}$, $V \in \mathbb{R}^{n\times n}$ e $\Sigma \in \mathbb{R}^{n\times n}$, ou seja, as dimensões de U e de Σ são diferentes do que definimos no início deste capítulo. Mostraremos agora como transformá-la em uma decomposição dita *completa* (i.e., conforme equação (4.1)).

Considerando ainda que A tem posto completo, sabemos que as colunas de U são vetores ortonormais em um espaço m-dimensional. Sabemos que, a não ser que $m = n$, elas não formam uma base de $\mathbb{R}^m$ e U não é uma matriz ortogonal. Basta, então, introduzirmos $m - n$ colunas em U para que ela seja ortogonal. No entanto, essa modificação exige que mudemos também a matriz Σ. Para que o produto dessas matrizes não se altere, as $m - n$ colunas de U adicionadas devem ser multiplicadas por zero. Logo, incluímos $m - n$ linhas zeradas em Σ e temos o SVD completo de A.

A idéia acima é mostrada na figura abaixo. A representação de matrizes é a mesma da figura do QR completo do capítulo anterior. A região não hachurada da matriz Σ representa os elementos nulos e as linhas tracejadas indicam as colunas de U e as linhas de Σ que foram adicionadas.

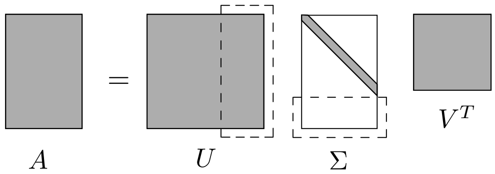


Figura 4.2: SVD Completo de $A \in \mathbb{R}^{m\times n}$, $m \geq n$.

Finalmente, observa-se que a decomposição SVD mencionada pode ser utilizada para uma matriz A sem que seja necessariamente de posto máximo. Seja r_A o posto de A. Para construir U, basta adicionarmos $m - r_A$ colunas e para construir V incluímos $n - r_A$ colunas ortonormais. A matriz Σ terá r_A entradas positivas e $n - r_A$ elementos iguais a zero.

4.1 Aproximação Para Um Menor Posto

Um dos aspectos que valorizam ainda mais o SVD é sua capacidade de lidar com o conceito de posto de matriz. Muitos teoremas de álgebra linear mostram suas afirmações com a suposição de que a matriz tenha posto completo. No entanto, problemas de computação numérica, como os erros de arredondamento, fazem com que seja difícil determinar precisamente tal posto. Nesta secção mostraremos que o SVD é uma chave para esse problema por caracterizar eficientemente uma aproximação de matrizes de um posto definido. Considere, inicialmente o seguinte teorema:

Teorema 4.1.1. *Seja $U\Sigma V^T$ o SVD de uma matriz $A \in \mathbb{R}^{m\times n}$, com $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, ..., \sigma_n)$. Temos, então, que $\|A\|_2 = \sigma_1$.*

Demonstração. Sabemos que $\|U\|_2 = \|V^T\|_2 = 1$ porque U e V são ortogonais. Assim,

$$\|A\|_2 = \|U\Sigma V^T\|_2 = \|U\|_2\|\Sigma\|_2\|V^T\|_2 = \|\Sigma\|_2 = \max_{\|x\|_2=1}\left(\sum_{i=1}^{n}(\sigma_i x_i)^2\right)^{1/2} = \sigma_1.$$

$\square$

A propriedade do SVD que nos interessa é dada pelo seguinte teorema:

Teorema 4.1.2. *Considere o SVD de $A \in \mathbb{R}^{m\times n}$ como sendo $A = U\Sigma V^T$. Para todo $0 \le k \le \mathrm{posto}(A) = r_A$, definimos:*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

Então, temos:

$$\|A - A_k\|_2 = \inf_{\substack{B\in\mathbb{R}^{m\times n}\\ \mathrm{posto}(B)\le k}} \|A - B\|_2 = \sigma_{k+1}.$$

Demonstração. Como $U^T A_k V = \mathrm{diag}(\sigma_1, ..., \sigma_k, 0, ..., 0)$, temos que $\mathrm{posto}(A_k) = k$. Além disso, como $U^T(A - A_k)V = \mathrm{diag}(0, ..., 0, \sigma_{k+1}, ..., \sigma_{r_A})$ então $\|A - A_k\|_2 = \sigma_{k+1}$. Note a diferença entre essa afirmação e o teorema 4.1.1 citado anteriormente. Suponha agora que exista uma matriz B com $\mathrm{posto}(B) \le k$ tal que $\|A - B\|_2 < \|A - A_k\|_2 = \sigma_{k+1}$. Então existe um subspaço W de $\mathbb{R}^m$ que é $(m - k)$-dimensional e tal que $w \in W \Rightarrow Bw = 0$. Assim, para todo $w \in W$, temos $Aw = (A - B)w$ e

$$\|Aw\|_2 = \|(A - B)w\|_2 \le \|A - B\|_2\|w\|_2 < \sigma_{k+1}\|w\|_2.$$

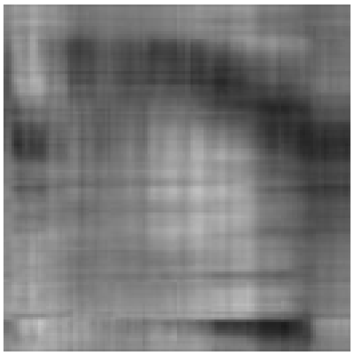
Dessa forma, W é um subspaço $(m - k)$-dimensional onde $\|Aw\|_2 < \sigma_{k+1}\|w\|_2$. Mas existe um subspaço $(k + 1)$-dimensional onde $\|Aw\|_2 \ge \sigma_{k+1}\|w\|_2$, que é o espaço gerado pelas primeiras $k + 1$ colunas de V. Como a soma das dimensões desses espaços é maior que m, chega-se a uma contradição e completamos a prova do teorema. $\square$

Pode-se provar, de maneira análoga, o seguinte resultado [Mir60]:

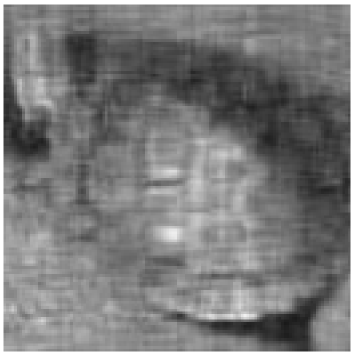
$$\|A - A_k\|_F = \inf_{\substack{B \in \mathbb{R}^{m \times n} \\ \text{posto}(B) \leq k}} \|A - B\|_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_{r_A}^2}.$$

O teorema 4.1.2 possui uma interpretação geométrica que responde a seguinte pergunta: Qual a melhor aproximação de dimensão k (com $k < m$) de uma hiperelipse no $\mathbb{R}^m$? Observe que a resposta dada pelo teorema a essa pergunta é a mesma que temos intuitivamente, ou seja, tomamos a hiperelipse gerada pelos maiores eixos em módulo. Mais precisamente, pegamos os eixos correspondentes aos k maiores valores singulares da matriz. Note que quando $k = \text{posto}(A)$, capturamos toda a matriz A.

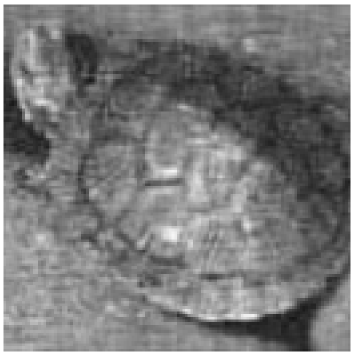
Uma aplicação interessante da aproximação do posto de matriz usando o SVD está na *compressão de imagens* [AP75]. Sabemos que uma imagem pode ser representada por uma matriz de posto p. Se, ao invés disso, armazenarmos tal imagem em uma matriz de posto $k < p$ (i.e., usarmos a aproximação de menor posto da matriz, é intuitivo pensarmos que a imagem originada desta será menos nítida que a original, já que foram retirados $p - k$ eixos (que representam "informações" na imagem) da hiperelipse. A figura 4.1 ilustra esse fato.



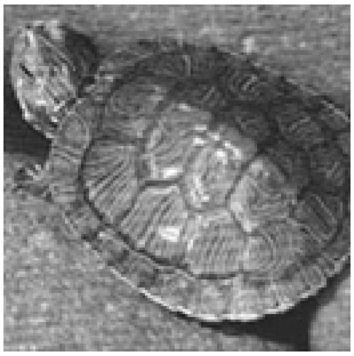
(a) Aproximação de posto k = 3



(b) Aproximação de posto k = 10



(c) Aproximação de posto k = 20



(d) Figura original de posto p completo

Figura 4.1: Compressão de imagens usando SVD.

4.2 Auto-Valores: Conceitos e Algoritmos

Alguns tópicos serão abordados nesta secção para compreender claramente o algoritmo SVD. Indicaremos apenas os conceitos que estejam relacionados diretamente ao SVD, omitindo-se, portanto, várias propriedades e resultados existentes. Maiores detalhes para os itens podem ser obtidos em [GL96, Capítulos 2 e 7] e [TB97, Secções I e V].

Auto-valores e Auto-vetores

Seja $A \in \mathbb{R}^{m \times m}$ uma matriz quadrada. Um vetor não nulo $x \in \mathbb{R}^m$ é um *auto-vetor* de A, e $\lambda \in \mathbb{R}$ é seu *auto-valor* correspondente, se

$$Ax = \lambda x.$$

A idéia por trás dessa igualdade é que, em certos casos, a ação de uma matriz A em um subspaço de $\mathbb{R}^m$ pode ser dada por uma simples multiplicação por um escalar. Algumas utilidades dos auto-valores e auto-vetores serão mostrados ao longo do texto.

Transformação de Similaridade

Se $X \in \mathbb{R}^{m \times m}$ é não singular, então o mapeamento $A \mapsto X^{-1}AX$ é chamado de *transformação de similaridade* de A. Dizemos ainda que duas matrizes A e B são *similares* se existe uma matriz X não singular tal que $B = X^{-1}AX$. Uma propriedade compartilhada entre matrizes similares pode ser vista no teorema abaixo:

Teorema 4.2.1. *Se X é não-singular, então A e $B = X^{-1}AX$ possuem os mesmos auto-valores.*

Demonstração. Considere λ um auto-valor de A associado a um auto-vetor x. Então:

$$Ax = \lambda x \Leftrightarrow AX^{-1}Xx = \lambda X^{-1}Xx \Leftrightarrow XAX^{-1}Xx = \lambda Xx \Leftrightarrow BXx = \lambda Xx,$$

ou seja, λ é auto-valor de B correspondente a um auto-vetor Xx. $\square$

Decomposição por Auto-valores

Considere uma matriz quadrada $A \in \mathbb{R}^{m \times m}$. Sua *decomposição por auto-valores* é dada por:

$$A = X\Lambda X^{-1},$$

onde X é uma matriz não-singular que possui como colunas os auto-vetores de A, e Λ é uma matriz diagonal com os elementos sendo os auto-valores de A.

Observe que $A = X\Lambda X^{-1}$ é equivalente a $AX = \Lambda X$ e que esta pode ser escrita como m equações do tipo $Ax_i = \lambda_i x_i$. Desse modo, a i-ésima coluna de X é o auto-vetor correspondente a i-ésima entrada de Λ (auto-valor). Vale indicar também que mesmo que os elementos de A pertençam a $\mathbb{R}$, muitas vezes os auto-valores correspondentes são complexos e não reais.

Vimos que qualquer matriz possui uma decomposição SVD. O mesmo não ocorre, no entanto, para a decomposição por auto-valores. Mais precisamente, esta decomposição existe apenas para a classe de matrizes chamadas *não-defectivas* (ou *diagonalizáveis*), ou seja, aquelas que possuem *multiplicidade algébrica* e *geométrica* iguais para cada um dos seus auto-valores. As definições desses termos e as propriedades relacionadas podem ser vistas em [TB97, Capítulo 24].

Fatoração de Schur

Dada uma matriz quadrada $A \in \mathbb{R}^{m \times m}$, sua *fatoração de Schur* é dada por

$$A = QTQ^T,$$

onde Q é uma matriz ortogonal e T é triangular superior. Como T possui essa forma triangular superior, seus auto-valores correspondem necessariamente aos elementos da sua diagonal. No entanto, como A e T são similares, pelo teorema 4.2.1, elas possuem os mesmos auto-valores.

Ao contrário da decomposição por auto-valores, a fatoração de Schur existe para qualquer matriz quadrada. A demonstração da existência pode ser vista em [TB97, Capítulo 24]. Veremos posteriormente que essa fatoração será uma das bases para o algoritmo QR, que computa auto-valores.

Redução à Forma Hessenberg

Para compreender o algoritmo SVD, deve-se estudar o algoritmo QR, que veremos no próximo item desta secção. Antes disso, no entanto, indicaremos uma estratégia - a ser usada no algoritmo QR - para transformar uma matriz quadrada qualquer na forma *Hessenberg*. Uma matriz A está na forma Hessenberg (ou Hessenberg superior) se $a_{ij} = 0$ para $i > j + 1$, conforme o exemplo abaixo:

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

A redução à forma Hessenberg [MW68] está baseada na chamada *decomposição de Hessenberg* dada por: $H = Q^T AQ$, com $A \in \mathbb{R}^{m \times m}$, e onde Q é uma matriz ortogonal e H está na forma Hessenberg.

Para computar essa decomposição, pode-se utilizar os refletores de Householder (vistos na secção 3.1). Inicialmente, seleciona-se um refletor Q_1^T responsável por zerar as linhas $3, ..., m$ da primeira coluna. Isso é feito sem alterar a primeira linha da matriz. Naturalmente, quando se multiplica Q_1 à direita de $Q_1^T A$, a primeira coluna permanece inalterada e os zeros introduzidos anteriormente permanecem. Esta idéia se repete para zerar as demais colunas, até obtermos a matriz com o formato desejado. Um exemplo do processo é ilustrado a seguir. Em negrito podem ser observados os elementos da matriz alterados em cada iteração:

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1^T} \underset{Q_1^T A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}} \xrightarrow{Q_1} \underset{Q_1^T A Q_1}{\begin{bmatrix} \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}$$

$$\xrightarrow{Q_2^T} \underset{Q_2^T Q_1^T A Q_1}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}} \xrightarrow{Q_2} \underset{Q_2^T Q_1^T A Q_1 Q_2}{\begin{bmatrix} \times & \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \times & \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}$$

$$\xrightarrow{Q_3^T} \underset{Q_3^T Q_2^T Q_1^T A Q_1 Q_2}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}} \xrightarrow{Q_3} \underset{Q_3^T Q_2^T Q_1^T A Q_1 Q_2 Q_3}{\begin{bmatrix} \times & \times & \times & \boldsymbol{\times} & \boldsymbol{\times} \\ \times & \times & \times & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \times & \times & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & 0 & \times & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & 0 & 0 & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}$$

Assim, definimos $Q = Q_1 Q_2 ... Q_{m-2}$ e temos $H = Q^T AQ$. O algoritmo é formulado a seguir. Vale indicar que seu custo é de $10m^3/3$ flops.

Algoritmo 4.2.1. *(Redução à forma Hessenberg usando refletores de Householder) Dada uma matriz $A \in \mathbb{R}^{m \times m}$, o seguinte algoritmo sobrescreve A com $H = Q^T AQ$, onde H é uma matriz no formato Hessenberg superior e Q é o produto de refletores de Householder. A função* house *é dada pelo algoritmo 3.1.1.*

function hessenberg(A)
    for $k = 1{:}m - 2$
        $v =$ **house**$(A(k + 1{:}m, k))$

$A(k+1{:}m, k{:}m) = (I - 2vv^T)A(k+1{:}m, k{:}m)$
$A(1{:}m, k+1{:}m) = A(1{:}m, k+1{:}m)(I - 2vv^T)$
end
end

Terminaremos esse tópico indicando um teorema [Dem97, Capítulo 4.4] que será importante para o algoritmo SVD. Tal teorema é válido para matrizes de Hessenberg *irredutíveis*, ou seja, que não possuem zeros na superdiagonal.

Teorema 4.2.2. *(Q Implícito) Seja $Q^T AQ = H$ uma matriz de Hessenberg superior irredutível. Então as i-ésimas colunas de Q, para $i = 2, ..., n$, são determinadas unicamente (a menos de sinal) pela primeira coluna de Q.*

Demonstração. Considere $Q^T AQ = H$ e $V^T AV = G$ matrizes de Hessenberg irredutíveis, com Q e V ortogonais e tal que $q_1 = v_1$. Queremos mostrar que $q_i = \pm v_i$ para todo $i > 1$, ou, equivalentemente, que $W = V^T Q = \text{diag}(\pm 1, ..., \pm 1)$. Temos que:

$$W = V^T Q \Rightarrow GW = GV^T Q = V^T AQ = V^T QH = WH.$$

Além disso, se $GW = WH$, então:

$$Gw_i = (GW)_i = (WH)_i = \sum_{j=1}^{i+1} h_{ji} w_j \Rightarrow h_{i+1,i} w_{i+1} = Gw_i - \sum_{j=1}^{i} h_{ji} w_j,$$

sendo que $(GW)_i$ e $(WH)_i$ correspondem as i-ésimas colunas das matrizes GW e WH respectivamente. Como $w_1 = [1, 0, ..., 0]^T$ e G é da forma Hessenberg, podemos usar indução em i para mostrar que w_i é não nulo apenas nos seus primeiros i elementos. Desse modo, W é triangular superior. Como W é também ortogonal, então, claramente, $W = \text{diag}(\pm 1, ..., \pm 1)$. $\square$

Quociente de Rayleigh

Considere uma matriz $A \in \mathbb{R}^{m \times n}$. O *quociente de Rayleigh* de um vetor $x \in \mathbb{R}^m$ é um escalar definido por

$$r(x) \doteq \frac{x^T Ax}{x^T x}.$$

A fórmula acima visa responder a seguinte questão: dado x, que escalar α "seria seu auto-valor" de modo a minimizar $\|Ax - \alpha x\|_2$? Pode-se provar também que o gradiente de $r(x)$, denotado por $\nabla r(x)$, é tal que $\nabla r(x) = \frac{2}{x^T x}(Ax - r(x)x)$ e se $\nabla r(x) = 0$, com $x \neq 0$, então x é um auto-vetor e $r(x)$ é seu auto-valor correspondente. Outras propriedades envolvendo o quociente de Rayleigh podem ser vistos em [TB97, Capítulo 27].

Métodos da Potência e da Iteração Inversa

Considere uma matriz $A \in \mathbb{R}^{m\times n}$ e um vetor $v^{(0)}$ com $\|v^{(0)}\| = 1$. O *método da potência* [PP73] produz uma seqüência $v^{(i)}$ que converge para o auto-vetor correspondente ao maior auto-valor de A. Tal método será uma das bases para a demonstração da convergência do algoritmo QR, o qual veremos posteriormente. O algoritmo do método da potência é dado abaixo:

Algoritmo 4.2.2. *(Método da Potência) Dada um matriz $A \in \mathbb{R}^{m\times n}$, este algoritmo produz uma seqüência $v^{(i)}$ que converge para o auto-vetor correspondente ao maior auto-valor dessa matriz A.*

$v^{(0)}$ = algum vetor com norma igual a 1
for $k = 1, 2, ...$
  $w = Av^{(k+1)}$
  $v^{(k)} = w/\|w\|$
  $\lambda^{(k)} = (v^{(k)})^T Av^{(k)}$
end

A análise do procedimento acima é simples. Podemos escrever $v^{(0)}$ como combinação linear de auto-vetores ortonormais q_i, ou seja, $v^{(0)} = a_1q_1 + a_2q_2 + ... + a_mq_m$. Como $v^{(k)}$ é um múltiplo de $A^kv^{(0)}$, temos, para alguma constante c^k:

$$\begin{aligned} v^{(k)} &= c^kA^kv^{(0)} \\ &= c^k(a_1\lambda_1^kq_1 + a_2\lambda_2^kq_2 + ... + a_m\lambda_m^kq_m) \\ &= c^k\lambda_1^k(a_1q_1 + a_2(\lambda_2/\lambda_1)^kq_2 + ... + a_m(\lambda_m/\lambda_1)^kq_m). \end{aligned}$$

Note que para $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_m| \geq 0$, temos que $v^{(k)}$ tende ao auto-vetor correspondente ao auto-valor λ_1 quando $k \to \infty$.

Um outro método similar ao método da potência é chamado de *método da iteração inversa*. Para todo escalar μ que não é auto-valor de A, os auto-vetores de $(A - \mu I)^{-1}$ são idênticos aos auto-vetores de A e seus auto-valores correspondentes são dados por $\{(\lambda_i - \mu)^{-1}\}$, com $\{\lambda_i\}$ sendo os auto-valores de A. Desse modo, fixando um escalar μ, temos um auto-valor λ_I mais próximo a ela, e $(\lambda_I - \mu)^{-1}$ será maior que $(\lambda_i - \mu)^{-1}$ para todo $i \neq I$. Se aplicarmos, então o método da potência para $(A-\mu I)^{-1}$, o processo converge rapidamente para o auto-vetor correspondente a λ_I e a convergência é mais rápida quanto melhor for o μ escolhido. Esta é a idéia da iteração inversa.

Algoritmo QR

O algoritmo QR será uma base importante para computarmos o SVD. A idéia básica deste algoritmo é calcular, para cada passo k, a fatoração QR da k-ésima potência da matriz. A versão mais básica do algoritmo QR e uma breve explicação desta são dadas a seguir.

Algoritmo 4.2.3. *(Algoritmo QR Simples) Dada uma matriz $A \in \mathbb{R}^{m \times m}$, este algoritmo retorna (no próprio A) uma matriz triangular superior com os elementos da diagonal convergindo para os auto-valores de A. Utiliza-se da função* qr*, dada pelo algoritmo 3.1.2 ou 3.2.2.*

$A^{(0)} = A$
for $k = 1, 2, ...$
    $[Q^{(k)}, R^{(k)}] = \mathbf{qr}(A^{(k-1)})$
    $A^{(k)} = R^{(k)} Q^{(k)}$
end

O algorimo acima converge a matriz para sua forma de Schur, ou seja, à forma triangular superior se A for arbitrária e para uma forma diagonal se A for simétrica. Esta convergência seria útil para encontrarmos os auto-valores da matriz. Para isso, seria necessário usarmos transformações de similaridade. Note, no entanto, que o algoritmo tem como operação justamente esse tipo de transformação, pois $A^{(k)} = R^{(k)} Q^{(k)} = (Q^{(k)})^T A^{(k-1)} Q^{(k)}$ e $Q^{(k)}$ é não singular.

Para que o algoritmo QR com matrizes simétricas seja utilizada na prática, algumas estratégias adicionais são conferidas. Os seguintes itens são úteis para que a convergência da matriz seja cúbica:

1. Antes de começar a iteração do algoritmo em si, a matriz é reduzida para a forma tridigonal. Isto é feito utilizando-se o algoritmo 4.2.1.

2. Para cada iteração, a matriz a ser fatorada será dada por $A^{(k)} - \mu^{(k)} I$, onde $\mu^{(k)}$ é um auto-valor estimado (que chamaremos de *shift*).

3. Quando um auto-valor é encontrado, o problema é “reduzido”, dividindo-se a matriz $A^{(k)}$ em submatrizes.

Sabemos que o item 1 requer $O(m^3)$ flops. A diagonalização de uma matriz a partir de uma matriz na forma Hessenberg é dada, na prática, com $O(m)$ iterações. Como cada iteração requer $O(m^2)$ flops, temos uma complexidade total cúbica. Caso o item 1 não seja realizado, cada iteração necessitará de $O(m^3)$ flops por usar a matriz inteira e a complexidade total seria de $O(m^4)$. Isto mostra a importância do item 1.

O item 3, por sua vez, mostra a redução do problema em dois subproblemas e isso naturalmente diminui o custo do algoritmo. O item 2 será discutido posteriormente. O algoritmo que incorpora as modificações citadas é dado abaixo:

Algoritmo 4.2.4. *(Algoritmo QR “Prático”) Dada uma matriz $A \in \mathbb{R}^{m \times m}$ simétrica, este algoritmo retorna (no próprio A) uma matriz diagonal com os elementos convergindo para os auto-valores de A. Utiliza-se aqui a função* qr*, dada pelo algoritmo 3.1.2 ou 3.2.2 e a função* hessenberg*, dada por 4.2.1.*

$[A^{(0)}, Q^{(0)}] = \mathbf{hessenberg}(A)$
for $k = 1, 2, ...$
  Escolher um shift $\mu^{(k)}$ (que será discutido posteriormente).
  $[Q^{(k)}, R^{(k)}] = \mathbf{qr}(A^{(k-1)} - \mu^{(k)} I)$
  $A^{(k)} = R^{(k)} Q^{(k)} + \mu^{(k)} I$
  if um elemento $A^{(k)}_{j,j+1}$ for suficientemente perto de zero
    Coloque $A_{j,j+1} = A_{j+1,j} = 0$ para obter $\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} = A^{(k)}$ e
    aplique o algoritmo QR para A_1 e A_2.
  end
end

O algoritmo é, essencialmente, uma implementação de um procedimento conhecido como *iteração simultânea* [TB97, Capítulo 28], que é uma extensão do método da potência [TB97, Capítulo 27], [GL96, Capítulo 7.3], mencionado em um tópico anterior desta mesma secção.

A idéia da iteração simultânea é aplicar o método da potência em vários vetores de uma única vez. Considere um conjunto de vetores $v_1^{(0)}, ..., v_n^{(0)}$ linearmente independentes. Se $A^k v_1^{(0)}$ converge para o maior auto-valor de A quando $k \to \infty$, é natural pensarmos que o espaço $\langle A^k v_1^{(0)}, ..., A^k v_n^{(0)} \rangle$ deve convergir para o espaço $\langle q_1, ..., q_n \rangle$, gerado pelos auto-vetores $q_1, ..., q_n$ de A correspondentes aos n maiores auto-valores da matriz.

O algoritmo da iteração simultânea define uma matriz com as colunas correspondentes aos vetores $v_i^{(0)}$, $i = 1{:}n$. Tal matriz é escrito como $\hat{Q}^{(0)}$ no pseudo-código a seguir:

Algoritmo 4.2.5. *Iteração Simultânea*

Escolher $\hat{Q}^{(0)} \in \mathbb{R}^{m \times n}$ com as colunas ortonormais.
for $k = 1, 2...$
  $Z = A\hat{Q}^{(k-1)}$
  $\hat{Q}^{(k)} \hat{R}^{(k)} = Z$
end

Comparando o algoritmo da iteração simultânea com o algoritmo QR simples, temos, claramente, que este é equivalente ao outro quando se escolhe como matriz inicial a identidade, ou seja, $\hat{Q}^{(0)} = I$. Para uma demonstração mais detalhada, veja [TB97, Capítulo 28]. Isto mostra que o algoritmo QR de fato computa os auto-valores de uma matriz.

Discutiremos agora a escolha de um shift no algoritmo QR. Assim como o cômputo de auto-valores é justificado pelo método da potência, a escolha do shift pode ser explicada pelo método da iteração inversa. Note que neste último método, a convergência será dada por um tempo arbitrário, devido a dependência do valor de um shift μ. No caso do algoritmo QR, a idéia é escolhermos um shift que garanta e acelere tal convergência.

Essa escolha do shift pode ser dada por diversas maneiras. Um dos métodos é usar o chamado *shift de Wilkinson.* Considere B como a submatriz $A^{(k)}(m-1{:}m, m-1{:}m)$:

$$B = \begin{bmatrix} a_{m-1} & b_{m-1} \\ b_{m-1} & a_m \end{bmatrix}.$$

O shift de Wilkinson é definido como o auto-valor de B mais próximo de a_m. Se os dois auto-valores de B forem igualmente próximos de a_m, a escolha será dada arbitrariamente. Uma fórmula para o shift de Wilkinson é:

$$\mu \doteq a_m + \delta - \text{sign}(\delta)\sqrt{\delta^2 + b_{m-1}^2},$$

onde $\delta \doteq (a_{m-1} - a_m)/2$. Se $\delta = 0$, então sign(δ) pode ser definido de modo arbitrário como 1 ou -1. Wilkinson (1968) mostrou em [Wil68] que o shift definido acima atinge, em média, uma convergência cúbica e usou heurísticas para dizer que tal shift deve ser usado preferencialmente.

4.3 Cômputo do SVD

Uma maneira de computar o SVD de uma matriz é utilizar a decomposição por auto-valores de uma matriz simétrica correspondente. O modo mais simples de fazer isso é descrito a seguir. Considere, inicialmente, o seguinte teorema:

Teorema 4.3.1. *Os valores singulares não-nulos de uma matriz A são as raízes quadradas dos auto-valores não-nulos de A^TA (ou AA^T).*

Demonstração. Temos que

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^T U^T U\Sigma V^T = V\Sigma^2 V^T.$$

Então A^TA é similar a Σ^2, e pelo teorema 4.2.1, eles possuem os mesmos n auto-valores. Claramente, os auto-valores da matriz diagonal Σ^2 são $\sigma_1^2, \sigma_2^2, ..., \sigma_p^2$ com $n-p$ auto-valores nulos adicionais quando $n > p$. A prova para a matriz AA^T é similar. □

Considere agora uma matriz $A \in \mathbb{R}^{m\times n}$, com $m \geq n$ e sua decomposição SVD como sendo $A = U\Sigma V^T$. Sabemos que $A^TA = V\Sigma^2V^T$. Utilizando-se a idéia do teorema mencionado acima, podemos calcular a decomposição SVD de A da seguinte forma:

1. Seja $C \doteq A^TA$.
2. Use o algoritmo QR (4.2.4) para computar a decomposição por auto-valores de C, ou seja, $C = V\Lambda V^T$, com $\Lambda = \text{diag}(\sigma_i^2)$.
3. Considere Σ como sendo a matriz que contêm as raízes quadradas dos elemetos de Λ.

4. Utilize a fatoração QR com pivotamento para obter U tal que $U\Sigma = AVP$, sendo P uma matriz de permutação.

Apesar desse algoritmo ser usado com uma certa freqüência, ele possui o problema de ser instável. A transformação de um problema de SVD a um problema de decomposição por auto-valores faz com que o número de condição do problema se eleve ao quadrado, ou seja, a sensibilidade à perturbações aumenta bastante [Ove01]. Uma outra maneira de computar o SVD, com estabilidade garantida, é dada a seguir.

Considere uma matriz $A \in \mathbb{R}^{m\times n}$ quadrada. Esta suposição não afeta o resultado que queremos, pois os valores singulares de uma matriz retangular pode ser reduzida aos valores singulares de uma matriz quadrada. Seja H uma matriz simétrica de dimensão $2m \times 2m$:

$$H \doteq \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}.$$

Sabemos que $A = U\Sigma V^T \Rightarrow AV = U\Sigma$ e que $A^T U = V\Sigma^T = V\Sigma$. Essas igualdades podem ser rearranjadas do seguinte modo:

$$\begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} V & V \\ U & -U \end{bmatrix} = \begin{bmatrix} V & V \\ U & -U \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix}.$$

Note que isso correponde justamente à decomposição de auto-valores de H. Temos ainda que os valores singulares de A são os valores absolutos dos auto-valores de H e que os vetores singulares de A podem ser facilmente obtidos dos auto-vetores de H. O algoritmo padrão do SVD, descrito por Golub e Kahan (1965) em [GK65], utiliza essa idéia. Veremos, no entanto, que não será necessária formarmos a matriz H de dimensão $m + n$ explicitamente.

A técnica utiliza duas fases. A primeira é responsável pela transformação da matriz A na forma bidiagonal e serve para diminuir a complexidade do algoritmo final. A segunda, por sua vez, consiste na aplicação do algoritmo QR implícito em H, conforme mencionado acima. Esse algoritmo QR implícito é uma adaptação do algoritmo QR que vimos, com a vantagem de que a matriz $A^{(k-1)} - \mu^{(k)} I$ não é formada explicitamente, para cada k. Discutiremos a idéia de tal algoritmo no próprio cômputo do SVD, na secção 4.3.2.

4.3.1 Bidiagonalização

A primeira etapa da decomposição SVD de A envolve o cômputo de matrizes U e V tais que $U^T AV$ é uma matriz bidiagonal. Um método conhecido como *bidiagonalização de Golub-Kahan* [TB97, Capítulo 31] utiliza reflexões de Householder à direita e à esquerda alternadamente. Cada refletor à esquerda zera uma coluna abaixo da diagonal e cada refletor à direita zera os elementos da linha à direita da superdiagonal correspondente. Um exemplo é mostrado a seguir:

Observe que neste processo n refletores são aplicados à esquerda e $n - 2$ são aplicados à direita. A matriz U é então obtida multiplicando-se todos os refletores à esquerda, ou seja,

$$
\underset{A}{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}
\xrightarrow{U_1^T}
\underset{U_1^T A}{\begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}
\xrightarrow{V_1}
\underset{U_1^T A V_1}{\begin{bmatrix} \times & \boldsymbol{\times} & \mathbf{0} & \mathbf{0} \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}
$$

$$
\xrightarrow{U_2^T}
\underset{U_2^T U_1^T A V_1}{\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}
\xrightarrow{V_2}
\underset{U_2^T U_1^T A V_1 V_2}{\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \boldsymbol{\times} & \mathbf{0} \\ 0 & 0 & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & 0 & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & 0 & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & 0 & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}
$$

$$
\xrightarrow{U_3^T}
\underset{U_3^T U_2^T U_1^T A V_1 V_2}{\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \boldsymbol{\times} & \boldsymbol{\times} \\ 0 & 0 & \mathbf{0} & \boldsymbol{\times} \\ 0 & 0 & \mathbf{0} & \boldsymbol{\times} \\ 0 & 0 & \mathbf{0} & \boldsymbol{\times} \end{bmatrix}}
\xrightarrow{U_4^T}
\underset{U_4^T U_3^T U_2^T U_1^T A V_1 V_2}{\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \boldsymbol{\times} \\ 0 & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & \mathbf{0} \end{bmatrix}}
$$

$U = U_1 ... U_n$. De modo análogo, temos que $V = V_1 ... V_{n-2}$. O algoritmo é dado a seguir. Vale indicar que seu custo é de $4mn^2 - \frac{4}{3}n^3$ flops.

Algoritmo 4.3.1. *(Bidiagonalização de Golub-Kahan) Dada uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$, o algoritmo fornece uma matriz bidiagonal $B = U^T AV$ (sobrescrito em A), onde U e V são ortogonais.*

for $j = 1{:}n$
  $v = \mathbf{house}(A(j{:}m, j{:}n))$
  $A(j{:}m, j{:}n) = (I_{m-j+1} - 2vv^T)A(j{:}m, j{:}n)$
  $A(j+1{:}m, j) = v(2{:}m-j+1)$
  if $j \leq n-2$
    $v = \mathbf{house}(A(j{:}m, j{:}n)^T)$
    $A(j{:}m, j+1{:}n) = A(j{:}m, j+1{:}n)(I_{n-j} - 2vv^T)$
    $A(j, j+2{:}n) = v(2{:}n-j)^T$
  end

end

Uma outra alternativa para o processo é utilizar a *bidiagonalização Lawson-Hanson-Chan* (ou LHC), que envolve a fatoração QR, a qual zera os elementos abaixo da diagonal principal. Inicialmente, obtém-se a fatoração QR de A, isto é, $A = QR$. Em seguida, usa-se a bidiagonalização de Golub-Kahan em R, ou seja, $B = U^T RV$. Este processo é ilustrado na figura 4.3.1.1. A fatoração QR tem o custo de $2mn^2 - \frac{2}{3}n^3$ flops e o Golub-Kahan requer $4nn^2 - \frac{4}{3}n^3 = \frac{8}{3}n^3$ flops. Deste modo, o custo total é de $2mn^2 + 2n^3$ flops.

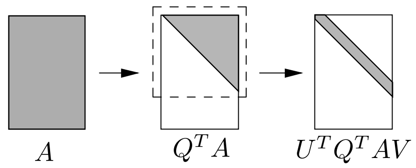


Figura 4.3.1.1: Bidiagonalização de Lawson-Hanson-Chan

Observe que a bidiagonalização LHC tem menor custo que o Golub-Kahan quando $4mn^2 - \frac{4}{3}n^3 > 2mn^2 + 2n^3 \Leftrightarrow m > \frac{5}{3}n$.

Discutiremos agora um outro processo de bidiagonalização que generaliza a idéia do LHC, mas que possui o custo menor para qualquer $m > n$. Tal procedimento é chamado de *bidiagonalização em 3 passos* e consiste em aplicar o processo do Golub-Kahan no início e, em um dado momento, aplicar o LHC. Considere $r = (m - k)/(n - k)$, como sendo a razão entre o número de linhas restantes (i.e., não diagonalizadas) e o número de colunas restantes no passo k. Para cada passo, computa-se r e quando $r = 2$, utiliza-se o LHC [TB97]. A idéia do algoritmo é mostrado na figura 4.3.1.2.

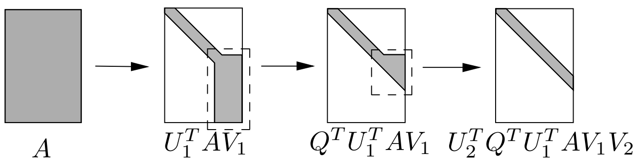


Figura 4.3.1.2: Bidiagonalização em 3 Fases

Desse modo, a bidiagonalização em 3 fases possui custo de $4mn^2 - \frac{4}{3}n^3 - \frac{2}{3}(m - n)^3$, o que é uma diminuição pequena, porém considerável para $n < m < 2n$.

4.3.2 Golub-Kahan

O passo que será dado iterativamente no algoritmo SVD é conhecido como o passo de *Golub-Kahan*. Conforme mencionado anteriormente, esta etapa consiste na aplicação do algoritmo QR implícito. Considere B como a matriz bidiagonal obtida pela fase anterior tirando-se os elementos nulos, ou seja, tal que $U_B^T A V_B = [B, 0]^T$.

O problema da decomposição da matriz A reduziu-se, portanto, em um problema de SVD de B. Definimos $d_1, d_2, ..., d_n$ e $f_1, f_2, ..., f_{n-1}$ como sendo os elementos da diagonal e da superdiagonal de B respectivamente. Mostraremos agora o algoritmo QR implícito aplicado na matriz tridiagonal $T = B^T B$:

1. Computar o auto-valor λ de
$$T(n-1:n, n-1:n) = \begin{bmatrix} d_{n-1}^2 + f_{n-2}^2 & d_{n-1}f_{n-1} \\ d_{n-1}f_{n-1} & d_n^2 + f_{n-1}^2 \end{bmatrix}$$
mais próximo de $d_n^2 + f_{n-1}^2$ (i.e., o shift de Wilkinson de T).

2. Computar $c_1 = \cos(\theta_1)$ e $s_1 = \mathrm{sen}(\theta_1)$ tal que
$$\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}^T \begin{bmatrix} d_1^2 - \lambda \\ d_1 f_1 \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}.$$
Definiremos $G_1 = G(1, 2, \theta_1)$.

3. Computar rotações de Givens $G_2, ..., G_{n-1}$ tais que $Q^T T Q$ é tridiagonal quando $Q = G_1 ... G_{n-1}$, sendo que a primeira coluna de Q e G_1 são iguais.

Note, no entanto, que esse último passo requer a formação explícita da matriz $T = B^T B$, o que não garante estabilidade. Uma alternativa para isso é discutida a seguir. Suponha que G_1 seja aplicada em B diretamente. Esse resultado é ilustrado com o exemplo abaixo:
$$BG_1 = \begin{bmatrix} \times & \times & 0 & 0 \\ + & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}.$$

Note que, a não ser pelo elemento indicado pelo sinal de soma, a matriz é bidiagonal. Para que ela tenha de fato esse formato, basta determinarmos rotações de Givens $U_1, V_2, U_2, ..., V_{n-1}, U_{n-1}$ conforme podemos observar abaixo:

Assim, temos uma nova matriz bidiagonal $\bar{B} = \bar{U}^T B \bar{V}$ tal que $\bar{U} = U_1 U_2 ... U_{n-1}$ e $\bar{V} = G_1 V_2 ... V_{n-1}$. Como cada V_i é tal que $V_i = G(i, i+1, \theta_i)$, $i = 2, ..., n-1$, então $\bar{V} e_1 = Q e_1$. Pelo teorema do Q implícito (4.2.2), temos que $\bar{V}$ e Q são essencialmente iguais. Toda essa idéia do passo de Golub-Kahan é dada pelo algoritmo abaixo:

Algoritmo 4.3.2. *(Golub-Kahan SVD) Dada uma matriz bidiagonal $B \in \mathbb{R}^{m \times n}$ sem elementos nulos na diagonal e na superdiagonal, o algoritmo fornece uma matriz bidiagonal $U^T B V$ (sobrescrito em B) onde U e V são ortogonais.*

$$B \xrightarrow{G_1} \underset{BG_1}{\begin{bmatrix} \times & \times & 0 & 0 \\ + & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}} \xrightarrow{U_1^T} \underset{U_1^T BG_1}{\begin{bmatrix} \times & \times & + & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}} \xrightarrow{V_2} \underset{U_1^T BG_1V_2}{\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & + & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}}$$

$$\xrightarrow{U_2^T} \underset{U_2^T U_1^T BG_1V_2}{\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & + \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}} \xrightarrow{V_3} \underset{U_2^T U_1^T BG_1V_2V_3}{\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & + & \times \end{bmatrix}} \xrightarrow{U_3^T} \underset{U_3^T U_2^T U_1^T BG_1V_2V_3}{\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}}$$

Ache μ o auto-valor da submatriz de $T = B'B$ (i.e, $T(n-1{:}n, n-1{:}n)$)
mais próximo de $T(n,n)$.
$y = t(1,1) - \mu$
$z = t(1,2)$
for $k = 1{:}n-1$
    Determine $c = \cos(\theta)$ e $s = \mathrm{sen}(\theta)$ tal que
$$\begin{bmatrix} y & z \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} * & 0 \end{bmatrix}$$
    $B = BG(k, k+1, \theta)$
    $y = b(k,k)$; $z = b(k+1,k)$
    Determine $c = \cos(\theta)$ e $s = \mathrm{sen}(\theta)$ tais que
$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$
    $B = G(k, k+1, \theta)^T B$
    if $k < n-1$
        $y = b(k, k+1)$; $z = b(k, k+2)$
    end
end

4.3.3 Algoritmo do SVD

Vimos que o algoritmo da decomposição de valores singulares consiste em computar a bidiagonalização da matriz e em aplicar o passo de Golub-Kahan em cada iteração. Considere ainda as definições de matrizes da secção anterior. Uma condição necessária para que se possa aplicar o passo de Golub-Kahan é que a matriz tridiagonal seja irredutível. Observemos, portanto, a existência de elementos nulos na diagonal e na superdiagonal de B^TB.

Se $f_k = 0$ para algum k, então montamos a seguinte matriz bloco-estrutural:

$$B = \begin{array}{cc} \left[\begin{array}{cc} B_1 & 0 \\ 0 & B_2 \end{array}\right] & \begin{array}{c} k \\ n-k \end{array} \\ \begin{array}{cc} k & n-k \end{array} & \end{array}$$

E quebra-se o problema original em dois outros sub-problemas envolvendo as matrizes B_1 e B_2, de dimensão menor. Se $d_k = 0$ para algum $k < n$, pode-se zerar o elemento f_k usando rotações de Givens. Esta idéia é semelhante a do passo de Golub-Kahan.

$$B = \left[\begin{array}{ccccc} \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{array}\right] \xrightarrow{G(2,3,\theta_1)} \left[\begin{array}{ccccc} \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & + & 0 \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{array}\right]$$

$$\xrightarrow{G(2,4,\theta_2)} \left[\begin{array}{ccccc} \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & + \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{array}\right] \xrightarrow{G(2,5,\theta_3)} \left[\begin{array}{ccccc} \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{array}\right]$$

Se $d_n = 0$, então a última coluna é zerada com uma série de rotações nos planos $(n-1, n)$,$(n-2, n)$,...,$(1, n)$. Desse modo, os casos em que $f_k = 0$ ou $d_k = 0$, para algum k, são facilmente tratados. Combinando o algoritmo da bidiagonalização, o algoritmo do passo de Golub-Kahan e a idéia mencionada acima, temos, finalmente, o algoritmo do SVD:

Algoritmo 4.3.3. *(Algoritmo SVD) Dada uma matriz $A \in \mathbb{R}^{m\times n}$ com $m \geq n$ e um ϵ, múltiplo pequeno do ϵ da máquina [Ove01], o seguinte algoritmo calcula $S = U^T AV$ diagonal (sobrescrito em A), onde $U \in \mathbb{R}^{m\times n}$ e $V \in \mathbb{R}^{n\times n}$ são ortogonais.*

Computar bidiagonalização abaixo (com algoritmo 4.3.2):
$\left[\begin{array}{c} B \\ 0 \end{array}\right] \longleftarrow (U_1...U_n)^T A(V_1...V_{n-2})$
while $q \neq n$
  for $i = 1{:}n-1$
    if $|b(i, i+1)| \leq \epsilon(|b(i, i)| + |b(i+1, i+1)|)$
      $b(1, 1+1) = 0$
    end
  end
  Encontrar maior q e o menor p tal que se

$$B = \begin{array}{cc} \left[\begin{array}{ccc} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{array}\right] & \begin{array}{c} p \\ n-p-q \\ q \end{array} \\ \begin{array}{ccc} p & n-p-q & q \end{array} & \end{array}$$

então B_{33} é diagonal e B_{22} não tem elemento nulo na superdiagonal.

if $q \leq n$

if existe elemento nulo na diagonal de B_{22}

Zerar o elemento da superdiagonal da mesma linha.

else

Aplicar algoritmo 4.3.2 em B_{22}

$B = \mathrm{diag}(I_p, U, I_{q+m-n})^T B \mathrm{diag}(I_p, V, I_q)$

end

end

O algoritmo acima tem como custo $4mn^2 + 8mn^2 + 9n^3$ quando computamos explicitamente a matriz de valores e vetores singulares. Quando se requer apenas a matriz de valores singulares, o custo diminui para $4mn^2 - \frac{4}{3}n^3$ [GL96, Capítulo 5].

Capítulo 5

Recuperação de Informações

A recuperação de informações (IR, *information retrieval*) lida com a representação, o armazenamento, a organização e o acesso às informações. Nesse contexto, tais informações são compostas por textos, imagens, áudios, vídeos e outros objetos de multimídia. Apesar destes diversos formatos, um sistema de IR manipula essas informações como se fossem apenas textos.

Neste trabalho, usaremos a palavra *usuário* para se referir aos usuários de tais sistemas, que não necessitam de conhecimentos profundos de computação ou tópicos relacionados. O termo *item*, por sua vez, será usado para representar uma pequena e completa unidade manipulada pelo sistema. A definição desta unidade, porém, varia com o tipo de manipulação de informação. Documentos completos, tais como livros, revistas e jornais, podem ser tratados como itens. Em outras ocasiões, um item seria um capítulo de livro ou um artigo.

As palavras *termo* e *palavra-chave* serão usadas para representar uma unidade associada a um determinado conceito. Podemos imaginá-las como sendo palavras de um dicionário. O termo *documento* será usado de modo similar ao termo *item*. Além disso, ambas são representações do conceito de um item, estabelecido pelos vários *termos* associados.

Um sistema de IR consiste em um programa que facilita usuários a encontrarem informações desejadas. Ele pode usar um hardware convencional ou especializado que suporte as funções de pesquisa e de conversão de objetos de multimídia para dados textuais. Seu principal objetivo é de minimizar o trabalho de um usuário durante a pesquisa da informação desejada. Esse trabalho pode ser expresso pelo tempo gasto durante a busca e pela qualidade da informação obtida.

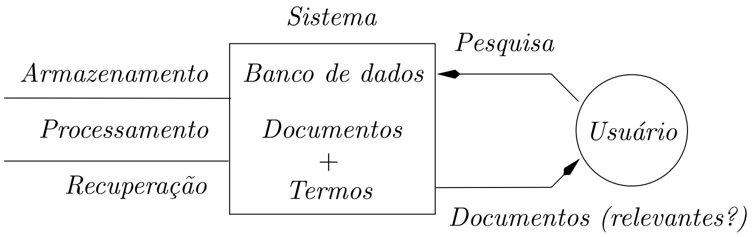


Figura 5.1: Sistema de recuperação de informações.

Claramente, o sucesso e a eficiência de um sistema de IR são medidos de maneira subjetiva. Em algumas circunstâncias, as informações desejadas são todos os dados que o sistema possui relacionados à pesquisa do usuário. Em outros casos, o usuário deseja apenas algumas informações que sejam suficientes a ele, e o retorno de todos os dados relevantes do sistema poderia atrapalhá-lo.

No contexto em que trabalhamos, o termo *relevante* se aplica a todos os itens contidos no sistema que representam as informações desejadas pelo usuário com sua pesquisa. Do ponto de vista do usuário, *relevante* e *necessário* são sinônimos. Por outro lado, na visão do sistema, algumas informações não relevantes para o usuário são consideradas relevantes por esse. Como exemplo, temos casos em que o usuário já conhece uma dessas informações consideradas relevantes ao sistema.

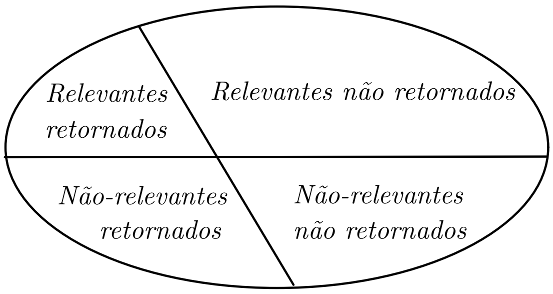


Figura 5.2: Possíveis efeitos de uma pesquisa sobre o espaço total de documentos.

Para uma certa pesquisa, o sistema retorna alguns documentos, relevantes ou não ao usuário. Além disso, no seu banco de dados temos duas partições de documentos: os relevantes e os não relevantes. A figura 5.2 ilustra o conjunto total de documentos e suas divisões para uma pesquisa. Levando-se isso em consideração, temos duas medidas comuns associadas a sistemas de IR - a precisão e o retorno - as quais são definidas abaixo:

$$\text{Precisão} \doteq \frac{\text{Número de documentos relevantes retornados}}{\text{Número total de documentos retornados}}.$$

$$\text{Retorno} \doteq \frac{\text{Número de documentos relevantes retornados}}{\text{Número total de documentos relevantes possíveis}}.$$

Inúmeros outros conceitos e tópicos de recuperação de informações não serão abordados neste trabalho, que foca essencialmente em um único modelo de sistema de IR - o modelo vetorial. Outros detalhes que envolvem IR podem ser obtidos em [BYRN99, KM00, vR79].

5.1 Modelagem

Nesta secção, apresentaremos brevemente os modelos clássicos de IR existentes. Para isso, devemos dizer claramente o que é tal modelo. Sua definição formal é dada por uma quádrupla $(D, P, \mathcal{F}, \mathrm{sim})$ onde:

a) D é o conjunto de documentos da coleção;

b) P é o conjunto de pesquisas (*queries*) possíveis;

c) $\mathcal{F}$ é o sistema usado para modelar as representações de termos, documentos, pesquisas e suas relações;

d) $\mathrm{sim} : (P, D) \to \mathbb{R}$ é uma função que para cada pesquisa $p_i \in P$ e cada documento $d_j \in D$, tem-se um número real $\mathrm{sim}(p_i, d_j)$ correspondente. Este número indica o quanto um documento d_j está relacionado a uma pesquisa p_i. Também pode indicar a posição (*ranking*) que um documento tem em relação aos outros com a pesquisa correspondente. Denotamos tal número como sendo a *similaridade* de p_i e d_j.

Veremos que esses quatro itens são essenciais para todos os modelos descritos a seguir. Os modelos clássicos de IR consideram que cada documento é representado por um conjunto de termos semanticamente relacionados. Considerando-se todas as palavras-chaves do sistema associadas a um documento, observa-se que nem todas são úteis para descrevê-lo. Por exemplo, considere uma coleção de cem mil documentos. Uma palavra que descreve todas elas não seria útil pois não fornece nenhuma informação para decidir quais documentos o usuário deseja.

Formalmente, seja m o número de termos (ou palavras-chaves) do sistema e t_i um termo qualquer. O conjunto de todos os termos é dado por $\{t_1 ..., t_m\}$. Um peso $a_{ij} \geq 0$ mede o quanto um termo t_i está relacionado semanticamente a um documento d_j. Se tal termo não tiver qualquer relação com tal documento, $a_{ij} = 0$. Desse modo, um documento d_j pode ser representado por um vetor $(a_{1j}, a_{2j}, ..., a_{mj})^T$. Ademais, definimos f_i como sendo uma função que dado um documento, retorna o peso associado ao termo t_i (i.e., $f_i(d_j) = a_{ij}$).

Vejamos agora um modelo simples e intuitivo: o *booleano*, cuja base está na teoria de álgebra booleana. Para esse modelo, todos os pesos são variáveis binárias, ou seja, $a_{ij} \in \{0, 1\}$. A pesquisa, por sua vez, será dada por uma expressão booleana, essencialmente na forma normal disjuntiva (DNF) [1]. Considere, por exemplo, uma pesquisa $p = [t_1 \wedge (t_2 \vee \neg t_3)]$.

[1]Uma expressão está em DNF se for uma disjunção de conjunções de literais, ou seja, da forma $\bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} t_{ij}$. Um literal, por sua vez, é uma expressão que é uma variável ou a negação de uma variável.

Ela pode ser escrita como:

$$p = (t_1 \wedge t_2) \vee (t_1 \wedge \neg t_3) = (t_1 \wedge t_2 \wedge t_3) \vee (t_1 \wedge t_2 \wedge \neg t_3) \vee (t_1 \wedge \neg t_2 \wedge \neg t_3).$$

Desse modo, temos que $p_{DNF} = [(1,1,1) \vee (1,1,0) \vee (1,0,0)]$, onde cada elemento p^i_{DNF} da disjunção é um vetor de pesos associados a tripla (t_1, t_2, t_3). A similaridade de um documento d_j com uma pesquisa p definida acima é dada por:

$$\text{sim}(d_j, p) = \begin{cases} 1 & \text{se } \exists p^i_{DNF} \mid \forall i = 1, ..., m,\ f_i(d_j) = f_i(p^i_{DNF}) \\ 0 & \text{caso contrário} \end{cases}$$

Se $\text{sim}(d_j, p) = 1$, então o modelo booleano considera que o documento d_j é relevante à pesquisa p. Caso contrário, é considerado não relevante. Observe que neste modelo, não existe a noção do quanto um documento é relevante a uma pesquisa. Essa pouca informação fornecida é uma das principais desvantagens de se usar o modelo booleano. Uma vantagem, no entanto, é sua simplicidade e formalismo que está por trás do modelo.

Ao contrário do modelo booleano, o modelo *probabilístico* nos permite obter informação sobre a relevância dos documentos retornados com a pesquisa. Essencialmente, esse modelo responde à questão: "Qual a probabilidade de um certo documento ser relevante à uma dada pesquisa?" Considere uma pesquisa representada por $p = (p_1, p_2, ..., p_m)$. Todos os pesos são também binários, ou seja, $a_{ij} \in \{0, 1\}$ e $p_i \in \{0, 1\}$. Temos ainda R_p como sendo o conjunto de documentos relevantes para uma pesquisa p. Seu complementar é denotado por $\bar{R}_p$.

Definimos $P(R_p|d_j)$ como sendo a probabilidade de que o documento d_j seja relevante à pesquisa p e $P(\bar{R}_p|d_j)$ como a probabilidade de d_j não ser relevante a p. A similaridade entre um documento d_j e uma pesquisa p é dado pela razão:

$$\text{sim}(d_j, p) = \frac{P(R_p|d_j)}{P(\bar{R}_p|d_j)} = \frac{P(d_j|R_p)P(R_p)}{P(d_j|\bar{R}_p)P(\bar{R}_p)}$$

$P(d_j|R_p)$ denota a probabilidade de selecionar o documento d_j entre o conjunto R_p de relevantes. Além disso, $P(R_p)$ indica a probabilidade de selecionar um documento relevante na coleção total. Os significados de $P(d_j|\bar{R}_p)$ e $P(\bar{R}_p)$ são análogos para os complementares. Como $P(R_p)$ e $P(\bar{R}_p)$ são idênticos para todos os documentos, então:

$$\text{sim}(d_j, p) \propto \frac{P(d_j|R_p)}{P(d_j|\bar{R}_p)}.$$

Para facilitar, redefinimos o valor de $\text{sim}(d_j, p)$ como sendo a razão $P(d_j|R_p)/P(d_j|\bar{R}_p)$ acima. Assumindo ainda que os termos são independentes, podemos escrever:

$$\text{sim}(d_j, p) = \frac{(\prod_{f_i(d_j)=1} P(t_i|R_p))(\prod_{f_i(d_j)=0} P(\bar{t}_i|R_p))}{(\prod_{f_i(d_j)=1} P(t_i|\bar{R}_p))(\prod_{f_i(d_j)=0} P(\bar{t}_i|\bar{R}_p))}.$$

A probabilidade de um termo t_i estar relacionado a um documento selecionado aleatoriamente do conjunto R_p é denotado por $P(t_i|R_p)$. Seu complementar é representado por $P(\bar{t_i}|R_p)$. Analogamente, $P(t_i|\bar{R_p})$ e $P(\bar{t_i}|\bar{R_p})$ são as probabilidades do termo t_i estar e não estar relacionado a um documento de $\bar{R_p}$, respectivamente. Sabendo que $P(t_i|R_p) + P(\bar{t_i}|R_p) = 1$ e que $P(t_i|\bar{R_p}) + P(\bar{t_i}|\bar{R_p}) = 1$, podemos ainda escrever, tomando-se logaritmos das expressões:

$$\mathrm{sim}(d_j, p) = \sum_{i=1}^{m} p_i a_{ij} \left(\log \frac{P(t_i|R_p)}{1 - P(t_i|R_p)} + \log \frac{1 - P(t_i|\bar{R_p})}{P(t_i|\bar{R_p})} \right)$$

Temos, assim, a função de similaridade mais usada para o modelo probabilístico. Note, no entanto, que o conjunto R_p não é conhecido inicialmente, o que exige a busca por alternativas no cômputo das probabilidades $P(t_i|R_p)$ e $P(t_i|\bar{R_p})$. Maiores detalhes em relação a isso poderão ser vistos em [vR79].

O terceiro modelo clássico de IR é o *modelo vetorial* [Ber01], que veremos com detalhes na próxima secção. Comparando-se os modelos citados acima, podemos dizer que o booleano é considerado o mais simples, porém o mais fraco, justamente por não permitir compararmos os documentos relevantes retornados.

Existe uma controvérsia em relação aos modelos probabilístico e vetorial. Croft [Cro83] realizou experiências, concluindo que o probabilístico possuía melhor performance que as demais, incluindo o vetorial. Em contrapartida, experimentos feitos por Salton e Buckley [SB88] mostraram que o modelo vetorial supera o probabilístico em coleções gerais.

5.2 Modelo Vetorial

No modelo de espaço vetorial, os itens ou documentos da coleção são representados por um vetor, e cada componente de tal vetor corresponde a um termo (ou palavra-chave) associada. O valor associado a essa componente, o peso, reflete a importância do termo na representação semântica do documento. Considere a matriz A de termos por documentos abaixo:

$$A = \begin{array}{cc} \begin{array}{ccccc} \phantom{a_{11}} & & d_j & & \phantom{a_{1n}} \end{array} & \\ \begin{bmatrix} a_{11} & \dots & a_{1j} & \dots & a_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \dots & a_{ij} & \dots & a_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mj} & \dots & a_{mn} \end{bmatrix} & t_i \end{array}$$

Conforme descrito acima, temos que a_{ij} é o peso do termo t_i associado ao documento d_j para $1 \leq i \leq m$, $1 \leq j \leq n$, com m e n sendo o número de termos e de documentos, respectivamente, contidos no banco de dados do sistema. Considere ainda p como sendo o vetor de pesquisa, representado por $p = (p_1, p_2, ..., p_m)$.

Os pesos a_{ij} podem ser representados de diversas formas. Usualmente, cada elemento a_{ij} pode ser decomposto em duas partes l_{ij} e g_i tais que $a_{ij} = l_{ij}g_i$. Nesse caso, g_i denota a influência *global* que um termo t_i tem sobre todos os documentos. Por sua vez, l_{ij} é o peso *local* que um termo t_i possui sobre um documento d_j.

Suponhamos, por exemplo, um conjunto de documentos com conteúdo matemático. O termo "matemática", nesse caso, teria uma influência global, sobre todos os documentos. Como a inclusão de tal termo na descrição desses documentos não é muito importante, é suficiente termos um valor de g_i pequeno. A palavra "geometria", por sua vez, é mais específica. Nem todos os documentos iriam tratar disso, por isso é apropriado termos pesos l_{ij} diferentes para cada documento d_j. Os fatores l_{ij} podem ser dadas de formas variadas, algumas das quais são listadas abaixo:

a) Pesos binários: 1 se o termo t_i tiver relação com o documento d_j e 0 caso contrário.

b) Número de vezes que o termo t_i aparece no documento d_j, denotado por $\mathrm{tf_{ij}}$.

c) Freqüência em que o termo t_i aparece no documento d_j, isto é, $\mathrm{tf_{ij}}/(\sum_i tf_{ij})$.

d) Freqüência de termo aumentado e normalizado, ou seja, $0.5 + 0.5(\mathrm{tf_{ij}}/\max \mathrm{tf_{ij}})$.

Outros formatos de l_{ij} mais sofisticados envolvem freqüências inversas, logaritmos de termos de freqüências e certas normalizações. Detalhes relacionados podem ser observados em [SB88, SJ72].

Considerando a existência de muitas palavras-chaves manipuladas por um sistema de IR, é natural que o número de termos relacionados a um documento é pequeno em relação ao número total de termos na coleção. A matriz A é, portanto, esparsa [2], bem como o vetor p. Veremos depois que essa característica ajuda na elaboração de sistemas mais eficientes.

Tendo-se a matriz A, a pesquisa p e os pesos conforme discutido acima, falta indicarmos a medição da similaridade de um documento d_j com uma pesquisa p. Sabendo que d_j e p são representados por vetores, podemos pensar que a similaridade está relacionada ao cosseno do ângulo formado por estes vetores. Se não tiverem relação alguma, os vetores são ortogonais entre si e o cosseno será zero. De maneira geral, temos:

$$\mathrm{sim}(d_j, p) = \cos(\theta_j) = \frac{a_j^T p}{\|a_j\|_2 \|p\|_2} = \frac{\sum_{i=1}^m a_{ij} p_i}{\sqrt{\sum_{i=1}^m a_{ij}^2}\sqrt{\sum_{i=1}^m p_i^2}}$$

para $j = 1, ..., n$ e sendo θ_j o ângulo formado pelo vetor de documento d_j e pela pesquisa p. Essa é a maneira mais simples de calcularmos similaridades. Note que como p e A são

[2] Uma matriz é esparsa se muitos dos seus elementos forem iguais a zero.

esparsos, os produtos e cálculos de normas são consideravelmente baratos de computar. Observe ainda que $\|a_j\|_2$ não precisa ser recomputada a cada pesquisa realizada.

Quanto maior o valor de $\mathrm{sim}(d_j, p)$, mais d_j e p estão relacionados. Para indicar o conjunto R_p de documentos relevantes à pesquisa p, basta escolhermos um limiar L adequadamente. Assim, $R_p = \{d_j | \, \mathrm{sim}(d_j, p) \geq L\}$. Esse valor de L é usualmente definido através de experimentos e é dependente do tipo de coleção que se está trabalhando. De forma geral, um bom limiar para a similaridade mencionada acima é 0.9 [BDO95].

Um Exemplo

Mostraremos agora os conceitos vistos anteriormente com um pequeno exemplo. O espaço vetorial, nesse caso, terá dimensão 7. Considere, inicialmente, $m = 7$ termos da coleção:

T1: surrealis(mo,ta)
T2: exposiç(ão,ões)
T3: Salvador
T4: Miró
T5: obra(s)
T6: subconsciência
T7: arte

Observe que os termos são indicados pela radical da palavra. Por essa razão, "surrealismo" e "surrealista" são semanticamente equivalentes e identificadas de maneira única. Do mesmo modo, o plural de uma palavra é identificada juntamente com seu singular correspondente. Sejam os $n = 5$ documentos existentes no sistema dados por:

D1: As principais *exposições* no centro de *Salvador*.
D2: Agenda de *exposições* do ano.
D3: *Exposição* no Museu X: o *surrealista* Joan *Miró*.
D4: História da *arte* de *Salvador*.
D5: *Surrealismo* e *subconsciência* de *Salvador* Dalí e Joan *Miró*: melhores *obras* da *exposição* de *arte* de 2004.

A matriz de termos × documentos é dada por:

$$\hat{A} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Usamos, nesse caso, pesos binários, com 1 se o termo estiver relacionado ao documento e zero caso contrário. Normalizando as colunas de $\hat{A}$, obtemos:

$$A = \begin{bmatrix} 0 & 0 & 0.5774 & 0 & 0.3780 \\ 0.7071 & 1.0000 & 0.5774 & 0 & 0.3780 \\ 0.7071 & 0 & 0 & 0.7071 & 0.3780 \\ 0 & 0 & 0.5774 & 0 & 0.3780 \\ 0 & 0 & 0 & 0 & 0.3780 \\ 0 & 0 & 0 & 0 & 0.3780 \\ 0 & 0 & 0 & 0.7071 & 0.3780 \end{bmatrix} \tag{5.1}$$

Considere o seguinte vetor de pesquisa, que representa a busca pelas palavras "surrealismo" e "Miró":

$$p_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T. \tag{5.2}$$

O vetor de cossenos, que mede a similaridade entre a pesquisa p_1 e os 5 documentos, é a seguinte:

$$\cos(\theta) = \begin{bmatrix} 0 & 0 & 0.8165 & 0 & 0.5345 \end{bmatrix}^T.$$

Considerando que os números de termos e de documentos desse exemplo são pequenos, escolhemos um limiar não muito grande de $L = 0.4$. Conclui-se, nesse caso, que os documentos $D3$ e $D5$ são relevantes. Os demais documentos foram corretamente considerados como não relevantes. Considere agora um outro vetor de pesquisa, que representa a busca pela palavra "Miró", apenas:

$$p_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T. \tag{5.3}$$

O vetor de cossenos, que mede a similaridade entre a pesquisa p_2 e os documentos da coleção, é dado por:

$$\cos(\theta) = \begin{bmatrix} 0 & 0 & 0.5774 & 0 & 0.3780 \end{bmatrix}^T.$$

Observe que um limiar de $L = 0.4$, faz com que apenas o documento $D3$ seja relevante. O documento $D5$, que possivelmente é do interesse do usuário, não foi retornado como relevante.

Diversos tipos de técnicas são utilizados para evitar esse tipo de erro, comum no modelo vetorial. Um método recente, chamado *Latent Semantic Indexing* (LSI), diminui a probabilidade de obter erros pela troca da matriz original por uma aproximada. O processo será demonstrado nas próximas secções, inicialmente com a fatoração QR e depois com a decomposição SVD, esta última usada de fato pela comunidade IR e desenvolvedores do LSI.

5.3 Fatoração QR

A fatoração QR pode ser uma ferramenta útil para identificar e remover informações redundates da matriz A de termos por documentos utilizada no sistema de IR. Em outras palavras, usaremos essa fatoração para identificar a melhor aproximação de A de posto reduzido. Considere, $AP = QR$ como sendo a fatoração QR de A (a mesma matriz dada na secção anterior) com pivotamento de colunas tais que:

$$Q = \left[\begin{array}{rrrrr|rr} -0.3780 & -0.4364 & 0.2801 & -0.1297 & 0.2673 & -0.4516 & 0.5441 \\ -0.3780 & -0.4364 & -0.5601 & 0.2593 & -0.5345 & 0 & 0 \\ -0.3780 & 0.3273 & -0.6301 & -0.2593 & 0.5345 & 0 & 0 \\ -0.3780 & -0.4364 & 0.2801 & -0.1297 & 0.2673 & 0.4516 & -0.5441 \\ -0.3780 & 0.3273 & 0.2100 & 0.4537 & 0 & -0.5441 & -0.4516 \\ -0.3780 & 0.3273 & 0.2100 & 0.4537 & 0 & 0.5441 & 0.4516 \\ -0.3780 & 0.3273 & 0.2100 & -0.6482 & -0.5345 & -0.1185 & 0 \end{array}\right],$$

$$R = \left[\begin{array}{rrrrr} -2.6457 & -1.1339 & -0.7559 & -0.7559 & -0.3780 \\ 0 & -1.3093 & -0.1091 & 0.6547 & -0.4364 \\ 0 & 0 & -1.1902 & -0.4201 & -0.5601 \\ 0 & 0 & 0 & -0.9075 & 0.2593 \\ 0 & 0 & 0 & 0 & -0.5345 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right] \quad \text{e} \quad P = \left[\begin{array}{ccccc} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{array}\right]$$

Podemos particionar a matriz R em duas partes: uma correspondente às linhas nulas de R e outra relacionada aos elementos restantes, que denotamos por R_A. As colunas de Q correspondentes a R_A formam uma submatriz que chamaremos de Q_A. A submatriz que completa Q e que está associada às linhas nulas de R é dado por $Q_A^{\perp}$. Com isso, podemos escrever:

$$A = \left[\begin{array}{cc} Q_A & Q_A^{\perp} \end{array}\right] \left[\begin{array}{c} R_A \\ 0 \end{array}\right] = Q_A R_A.$$

Claramente temos que $Q_A^{\perp}$ não contribui para os valores de A e que os postos de A e R_A são iguais. Denotamos tal posto por r_A. Sabe-se que as colunas de Q_A por si só formam uma base para o espaço gerado pelas colunas de A de posto r_A. Além disso, as colunas de $Q_A^{\perp}$ formam uma base para o complementar ortogonal do espaço gerado pelas colunas da matriz AP.

Para facilitar, usaremos a matriz A no lugar de AP. Tendo a fatoração QR da matriz, podemos escrever a similaridade entre um documento d_j e uma pesquisa p com sendo:

$$\text{sim}(d_j, p) = \cos(\theta_j) = \frac{a_j^T p}{\|a_j\|_2 \|p\|_2} = \frac{(Q_A r_j)^T p}{\|Q_A r_j\|_2 \|p\|_2} = \frac{r_j^T (Q_A^T p)}{\|r_j\|_2 \|p\|_2}.$$

Ao utilizar o cálculo de similaridades acima, observa-se que o vetor de cossenos continua o mesmo. Ou seja, não houve perda de informação conforme se previa. Note agora que se uma matriz é ortogonal, temos:

$$I = QQ^T = \begin{bmatrix} Q_A & Q_A^{\perp} \end{bmatrix} \begin{bmatrix} Q_A & Q_A^{\perp} \end{bmatrix}^T = Q_A^T Q_A + Q_A^{\perp}(Q_A^{\perp})^T.$$

Podemos, então, escrever o vetor de pesquisa p como a soma da projeção ortogonal de p no espaço gerado pelas colunas de A com a projeção ortogonal complementar de p no espaço gerado pelas mesmas colunas, ou seja:

$$p = Ip = QQ^T p = (Q_A Q_A^T + Q_A^{\perp}(Q_A^{\perp})^T)p = Q_A Q_A^T p + Q_A^{\perp}(Q_A^{\perp})^T p.$$

Definindo $p_A \doteq Q_A Q_A^T p$ e $p_A^{\perp} \doteq Q_A^{\perp}(Q_A^{\perp})^T p$, podemos dizer que p_A é a melhor aproximação de p no espaço gerado pelas colunas de A. Substituindo p pela expressão $p_A + p_A^{\perp}$, temos que a similaridade de p com um documento d_j é dada por:

$$\text{sim}(d_j, p) = \cos(\theta_j) = \frac{a_j^T p_A + a_j^T p_A^{\perp}}{\|a_j\|_2 \|p\|_2} = \frac{a_j^T p_A + a_j^T Q_A^{\perp}(Q_A^{\perp})^T p}{\|a_j\|_2 \|p\|_2}.$$

Como a_j é ortogonal às colunas de $Q_A^{\perp}$, temos que $a_j Q_A^{\perp} = 0$ e assim:

$$\cos(\theta_j) = \frac{a_j^T p_A}{\|a_j\|_2 \|p\|_2}. \tag{5.4}$$

Esta última formulação do vetor de cossenos sugere que o uso da fatoração QR nos oferece automaticamente vetores de pesquisa mais compatíveis que a original dada pelo usuário. A componente $p_A^{\perp}$ da pesquisa, que não está associada ao espaço gerado pelas colunas de A, são ignoradas. Levando-se tal observação mais adiante, podemos ter uma nova medida de similaridade, com a comparação na projeção do vetor de pesquisa no espaço gerado pelos vetores de documentos:

$$\cos(\theta_j^*) = \frac{a_j^T p_A}{\|a_j\|_2 \|p_A\|_2}. \tag{5.5}$$

A relação entre 5.4 e 5.5 é verificada a seguir:

$$\cos(\theta_j) = \cos(\theta_j^*) \frac{\|p_A\|_2}{\|p\|_2} = \cos(\theta_j^*) \frac{\|p_A\|_2}{\sqrt{\|p_A\|_2^2 + \|p_A^{\perp}\|_2^2}} \Rightarrow \cos(\theta_j) \leq \cos(\theta_j^*).$$

Isso mostra que usar a similaridade 5.5 pode retornar mais documentos relevantes ao sistema que 5.4. Na prática, isso implica que pode-se também aumentar o número de documentos não relevantes no conjunto dos retornados. Usando-se a terminologia de IR vistos no começo desse capítulo, temos um aumento de *retorno* sob o risco de reduzir a *precisão*.

5.4 Aproximação de Posto de Matriz

Um banco de dados de um sistema IR representado pela matriz de termos $\times$ documentos pode ser construído por um longo tempo, por inúmeras pessoas de opiniões e conhecimentos distintos. Todas essas incertezas são refletidas na matriz. Um bom meio de contornar esse problema é usar a fatoração QR ou a decomposição SVD da matriz. Veremos nessa secção como a fatoração QR pode lidar com essas incertezas, auxiliando no retorno de documentos mais relevantes para uma pesquisa. Posteriormente, em outra secção, descreveremos o uso da decomposição SVD nesse mesmo contexto.

Vimos na secção 3.3 que a fatoração QR com pivotamento de uma matriz A fornece matrizes P, Q e R tais que $AP = QR$ e que as primeiras k colunas de Q formam uma base do espaço gerado pelas primeiras k colunas de AP. Além disso, as primeiras k linhas de R fornecem os coeficientes das combinações lineares dos vetores dessa base. Em outras palavras, se Q_k é a matriz $m \times k$ com as primeiras k colunas de Q e r_i representa a i-ésima coluna de R, então $APe_i = Q_A r_i$. Definimos agora uma matriz A_k tal que:

$$A_k \doteq Q_k R_k P^T$$

onde Q_k e R_k correspondem às primeiras k colunas de Q e as primeiras k linhas de R, respectivamente. Denotamos esse A_k como sendo a *aproximação de A de posto k.*

As incertezas da matriz original A fazem com que possamos substitui-la por uma outra, $A - E$, onde E seria uma "matriz de incertezas", ou seja, que corresponde as mudanças feitas para que as informações incompletas e duvidosas diminuam. Veremos que essa troca será dada justamente por uma matriz aproximada de A. Considere, então, o exemplo das secções anteriores (matriz 5.1) e a fatoração QR dada. Particionamos a matriz R da seguinte forma:

$$R = \left[\begin{array}{rrr|rr} -2.6457 & -1.1339 & -0.7559 & -0.7559 & -0.3780 \\ 0 & -1.3093 & -0.1091 & 0.6547 & -0.4364 \\ 0 & 0 & -1.1902 & -0.4201 & -0.5601 \\ \hline 0 & 0 & 0 & -0.9075 & 0.2593 \\ 0 & 0 & 0 & 0 & -0.5345 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right] = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$$

Uma matriz A_3, ou seja, uma aproximação de A de posto 3, pode ser contruída simplesmente zerando a matriz R_{22}. Observe agora que a matriz E é dada por:

$$E = A - A_k = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} - Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} = Q \begin{bmatrix} 0 & 0 \\ 0 & R_{22} \end{bmatrix}$$

Além disso, temos que

$$\|E\|_F = \left\| Q \begin{bmatrix} 0 & 0 \\ 0 & R_{22} \end{bmatrix} \right\|_F = \left\| \begin{bmatrix} 0 & 0 \\ 0 & R_{22} \end{bmatrix} \right\|_F = \|R_{22}\|_F$$

Sabemos também que $\|A\|_F = \|R\|_F$. Portanto, a mudança relativa entre uma matriz aproximada A_k e a original A é dada por:

$$\frac{\|E\|_F}{\|A\|_F} = \frac{\|A - A_k\|_F}{\|A\|_F} = \frac{\|R_{22}\|_F}{\|R\|_F}.$$

No exemplo, $\|E\|_F/\|A\|_F = 32.7\%$, ou seja, uma mudança de 32.7% na matriz original faz com que o posto diminua de 2. Usando-se a nova matriz A_3, podemos calcular o vetor de similaridades para a pesquisa p_2. O resultado é o seguinte:

$$\cos(\theta) = \begin{bmatrix} 0 & 0 & 0.5774 & 0 & 0.4472 \end{bmatrix}^T.$$

Usando-se o mesmo limiar anterior, isto é, $L = 0.4$, observa-se que o documento D_5, que não tinha sido retornado anteriormente, aparece corretamente como um dos documentos relevantes. Isto mostra que aparentemente A_3 representa melhor o banco de dados que a matriz original A, de posto 5.

Considere agora uma aproximação de A de posto 2 (i.e., A_2). Nesse caso, $\|E\|_F/\|A\|_F = 55.4\%$ e o vetor de cossenos para a pesquisa $p2$ é dada por:

$$\cos(\theta) = \begin{bmatrix} 0.3652 & 0.5774 & 0.5774 & 0 & 0.4472 \end{bmatrix}^T.$$

Note que agora temos um documento a mais - $D2$ - retornado como relevante. No entanto, este não tem relação alguma com a pesquisa, o que nos leva à conclusão de que uma mudança de 55.4% na matriz é inaceitável.

Em geral, não é possível explicar a razão de uma variante da matriz ser melhor que outra para um conjunto de pesquisas dado. Observamos, porém, que uma diminuição razovelmente pequena do posto da matriz pode melhorar a performance do sistema. Experimentos mostraram que em sistemas reais de IR com uma coleção consideravelmente rica e no contexto científico, deve se ter uma exatidão de aproximadamente 0.1% [BDO95].

5.5 Decomposição por Valores Singulares

Um outro meio de contornar o problema dos erros e incertezas do banco de dados é usar a decomposição SVD da matriz de termos $\times$ documentos. Essa é a base do *Latent Semantic Indexing* (LSI) [BDJ99, BDO95, HSD01]. O SVD possui uma propriedade interessante (teorema 4.1.2): a fácil obtenção da melhor aproximação de uma matriz de um posto definido. Outra diferença entre o SVD e a fatoração QR é que a primeira, ao contrário da segunda, oferece não apenas uma redução de posto da base das colunas, mas também informações sobre o espaço gerado pelas linhas da matriz.

Em outras palavras, se QR é a fatoração QR de A e se $U\Sigma V^T$ é sua decomposição SVD, temos que Q e U possuem colunas que geram o mesmo espaço gerado pelas colunas de A. O SVD, no entanto, fornece também a matriz V, cujas colunas geram o mesmo espaço das linhas de A. Isso é uma vantagem, já que dessa forma poderão ser feitas comparações entre termos (veja secção 5.6 desse texto).

Vejamos agora, utilizando-se a matriz 5.1, as vantagens de se usar essa decomposição. Do mesmo modo que a fatoração QR, cria-se uma matriz A_k, denotada por aproximação de A de posto k. Note que essa idéia já foi abordada na secção 4.1 e que uma das aplicações está na compressão de imagens. A idéia aqui é análoga. Considere, então, a decomposição SVD de A dada por $U\Sigma V^T$ tais que:

$$U = \begin{bmatrix} -0.2672 & 0.1448 & -0.5725 & -0.2807 & -0.3298 & -0.5319 & -0.7051 \\ -0.7548 & 0.4762 & 0.3851 & 0.1582 & 0.1740 & 0 & 0 \\ -0.4537 & -0.6326 & 0.2438 & -0.3361 & -0.4708 & 0 & 0 \\ -0.2672 & 0.1448 & -0.5725 & -0.2807 & -0.3298 & 0.0532 & 0.7051 \\ -0.1131 & -0.0678 & -0.2257 & 0.5712 & -0.3246 & -0.7051 & 0.0532 \\ -0.1131 & -0.0678 & -0.2257 & 0.5712 & -0.3246 & 0.7051 & -0.0532 \\ -0.2369 & -0.5675 & -0.1867 & 0.2277 & 0.7315 & 0 & 0 \end{bmatrix},$$

$$\Sigma = \begin{bmatrix} 1.6696 & 0 & 0 & 0 & 0 \\ 0 & 1.0958 & 0 & 0 & 0 \\ 0 & 0 & 0.8547 & 0 & 0 \\ 0 & 0 & 0 & 0.3972 & 0 \\ 0 & 0 & 0 & 0 & 0.3513 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ e}$$

$$V = \begin{bmatrix} -0.5118 & -0.1009 & 0.5203 & -0.3167 & -0.5974 \\ -0.4527 & 0.4346 & 0.4506 & 0.3983 & 0.4951 \\ -0.4458 & 0.4034 & -0.5133 & -0.5861 & 0.1775 \\ -0.2925 & -0.7743 & 0.0472 & -0.1929 & 0.5248 \\ -0.4994 & -0.1966 & -0.5105 & 0.6003 & -0.3018 \end{bmatrix}.$$

Vimos no teorema 4.1.2 que uma aproximação de A de posto $k < \text{posto}(A) = r_A$ é dada por $A_k = U_k \Sigma_k V_k^T = \sum_{i=1}^{k} u_i \sigma_i v_i^T$, onde U_k e V_k correspondem as primeiras k colunas de U e V, respectivamente e Σ_k é a matriz $k \times k$ com os k maiores valores singulares de A. Note que para formar A_k, basta zerar os últimos $m - k$ valores singulares em Σ. Assim, a mudança relativa de A para A_k é dada por:

$$\frac{\|E\|_F}{\|A\|_F} = \frac{\|A - A_k\|_F}{\|A\|_F} = \sqrt{\frac{\sum_{i=k+1}^{r_A} \sigma_i^2}{\sum_{i=1}^{r_A} \sigma_i^2}}$$

No nosso exemplo, temos que $\|A - A_3\|_F / \|A\|_F = 23.7\%$ e que $\|A - A_2\|_F / \|A\|_F = 45\%$. Conforme se esperava, as mudanças relativas de 23.7% e 45% usando a decomposição

SVD são inferiores às mudanças de 32.7% e 55.4% necessárias usando-se a fatoração QR. Considere as matrizes A_3 geradas por cada fatoração:

$$A_3^{(QR)} = \begin{bmatrix} 0 & 0 & 0.5774 & 0 & 0.3780 \\ 0.7071 & 1.0000 & 0.5774 & 0 & 0.3780 \\ 0.3535 & 0 & 0 & 0.7071 & 0.3780 \\ 0 & 0 & 0.5774 & 0 & 0.3780 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.3535 & 0 & 0 & 0.7071 & 0.3780 \end{bmatrix}$$

$$A_3^{(SVD)} = \begin{bmatrix} -0.0422 & 0.0501 & 0.5141 & -0.0154 & 0.4414 \\ 0.7635 & 0.9447 & 0.6033 & -0.0200 & 0.3587 \\ 0.5660 & 0.1351 & -0.0489 & 0.7681 & 0.4082 \\ -0.0422 & 0.0501 & 0.5141 & -0.0154 & 0.4414 \\ 0.0037 & -0.0339 & 0.1532 & 0.1036 & 0.2074 \\ 0.0037 & -0.0339 & 0.1532 & 0.1036 & 0.2074 \\ 0.1822 & -0.1633 & 0.0074 & 0.5897 & 0.4012 \end{bmatrix}$$

Observe que a matriz A original é visualmente mais semelhante à aproximação dada pela fatoração QR. Isso mostra que não se pode assumir nada simplesmente pela aparência da matriz. Uma outra observação interessante está no fato de que alguns elementos de $A_3^{(SVD)}$ são negativos. Isto não é um problema, já que tais elementos são combinações lineares dos elementos de A. Devemos lembrar também que o modelo vetorial leva em consideração as relações geométricas dos vetores de documentos e não seus componentes individuais.

Vejamos agora como usar a decomposição por valores singulares para achar a similaridade entre documentos e uma pesquisa. Sejam A_k a matriz de termos $\times$ documentos aproximada com posto k e p o vetor de pesquisa. Então:

$$\operatorname{sim}(d_j, p) = \cos(\theta_j) = \frac{(A_k e_j)^T p}{\|A_k e_j\|_2 \|p\|_2} = \frac{(U_k \Sigma_k V_k^T e_j)^T p}{\|U_k \Sigma_k V_k^T e_j\|_2 \|p\|_2} = \frac{e_j^T V_k \Sigma_k (U_k^T p)}{\|\Sigma_k V_k^T e_j\|_2 \|p\|_2}$$

para $j = 1, ..., n$. Considere $s_j \doteq \Sigma_k V_k^T e_j$. Logo, para $j = 1, ..., n$, temos:

$$\cos(\theta_j) = \frac{s_j^T (U_k^T p)}{\|s_j\|_2 \|p\|_2}. \tag{5.6}$$

Note que 5.6 pode ser computado sem formar a matriz A_k explicitamente e que $\|s_j\|$ é calculada apenas uma vez para cada matriz de termos $\times$ documentos e independemente da pesquisa.

Sabemos que $U_k^T A_k e_j = \Sigma_k V_k^T e_j$, ou seja, os k elementos de s_j são as coordenadas da j-ésima coluna de A_k na base definida pelas colunas de U_k. Além disso, os k elementos de $U_k^T p$ são as coordenadas na base da projeção $U_k U_k^T p$ do vetor de pesquisa p no espaço gerado pelas colunas de A_k. A partir dessas observações, podemos criar uma nova medida de similaridade:

$$\cos(\theta_j^*) = \frac{s_j^T (U_k^T p)}{\|s_j\|_2 \|U_k^T p\|_2}. \tag{5.7}$$

Observe ainda que:

$$\cos(\theta_j) = \cos(\theta_j^*) \frac{\|U_k^T p\|_2}{\|p\|_2} \Rightarrow \cos(\theta_j) \leq \cos(\theta_j^*),$$

o que mostra que o uso da similaridade 5.7 pode retornar mais documentos que usando 5.6, ou seja, aumenta-se o retorno sob o risco de diminuir a precisão.

Na prática, no LSI [BDS95, BF96], usar a matriz aproximada diminui o custo de verificação dos documentos relevantes durante a pesquisa. O custo diminui mais ainda se levarmos em conta que a decomposição completa de A não precisa ser computada. É suficiente selecionar apenas os valores e vetores singulares que formam U_k, Σ_k e V_k.

Voltando à pesquisa $p2$ do exemplo deste capítulo, usando a aproximação de posto 3 (mudança de 23.7%), temos o seguinte vetor de similaridades como resultado:

$$\cos(\theta) = \begin{bmatrix} -0.0436 & 0.0516 & 0.5297 & -0.0158 & 0.4572 \end{bmatrix}^T.$$

Note que, utilizando-se o mesmo limiar de $L = 0.4$ anterior, apenas os documentos $D3$ e $D5$ são identificados como relevantes, como queríamos. Para uma aproximação de posto 2, temos:

$$\cos(\theta) = \begin{bmatrix} 0.2464 & 0.3032 & 0.3037 & 0.0078 & 0.2225 \end{bmatrix}^T.$$

De modo análogo ao que vimos na fatoração QR, uma mudança muito grande na matriz (de 45% usando o SVD) faz com que tenhamos um resultado não desejável. Nesse caso, todos os documentos foram considerados não relevantes.

5.6 Agrupamento de Termos

O conceito de agrupamento (*clustering*) de termos se origina pela criação de *thesauri*. Um *thesaurus*, por sua vez, consiste em uma lista de palavras importantes de um domínio de conhecimento onde cada palavra possui uma lista de *termos relacionados*. Dizemos que uma palavra está relacionada a outra se possuem um grau alto de similaridade, ou seja, se são sinônimas ou se possuem significados razoavelmente próximos. Denotamos ainda uma *classe* como sendo um conjunto de termos relacionados.

Segundo Foskett [For97], as principais utilidades de um *thesaurus* são, basicamente: (i) possuir um vocabulário padrão (ou sistema de referências) para indexação e pesquisa; (ii) auxiliar usuários a localizar termos com formulações de pesquisas próprias; (iii) e obter classificações hierárquicas que permitem ampliar ou reduzir dados de pesquisas de acordo com as necessidades dos usuários.

O processo de agrupamento segue os passos abaixo:

a) Definir o domínio em que se dá os agrupamentos, ou seja, identificar o escopo dos objetos que serão usados no processo de agrupamento.

b) Determinar os atributos dos objetos a serem agrupados. Se um *thesaurus* está sendo criado, deve-se especificar as palavras a serem usadas no processo de agrupamento.

c) Determinar os pesos dos atributos, ou seja, verificar quais objetos são sinônimos ou o quanto eles estão relacionados.

d) Aplicar um algoritmo para determinar as classes que cada item pertence.

Cada classe deve ainda possuir certas características, as quais são listadas a seguir:

a) Cada classe deve possui uma definição semântica "bem definida". A existência de classes "computador" e "hardware", por exemplo, pode confundir um usuário interessado por "memória de computador".

b) As classes devem ter a mesma ordem de grandeza em relação ao tamanho. Uma classe que contém 90% das palavras seria pouco útil.

c) Um objeto de uma classe não deve dominá-la. Considere uma classe que contém as palavras "microprocessador"', "processador 286" e "processador 386". Se o termo "microprocessador" é encontrado em 85% das vezes e os outros são encontrados apenas em 5% das vezes, existe uma grande possibilidade de que "processador 286" e "processador 386" sejam sinônimos de "microprocessador". Seria melhor, dessa maneira, criarmos uma classe separada para o termo "microprocessador".

d) Um objeto pode ser colocado em classes distintas até certo ponto. Colocar uma palavra em várias classes facilita a especificação de sua semântica, mas dificulta a criação e a manutenção das classes em que ela foi inserida.

Todas essas características desejáveis de classes dificultam o modo de criá-las. A criação de um *thesaurus* com a intervenção humana possibilita diferentes relações entre palavras. Aitchison e Gilchrist [AGB00] especificam três tipos de relacionamentos: *equivalentes*, *hierárquicos* e *não-hierárquicos*. A equivalência a é representação mais comum para os sinônimos e palavras de significados próximos. A relação de hierarquia, por outro lado, associa uma classe geral com uma específica. O exemplo citado anteriormente, com a classe geral "computador" e específica "hardware" ilustra esse tipo de relação. Demais tipos de relacionamentos são classificados como sendo não-hierárquicos.

Uma outra dificuldade está na existência de homógrafos, ou seja, palavras que possuem mais de um significado. Tais tipos de palavras dificultam o processo de recuperação de informações. Um possível modo de determinar seu verdadeiro significado em um documento é comparando-o com outros termos da pesquisa. Limitar o domínio do *thesaurus* também pode auxiliar a driblar esse tipo de problema.

Naturalmente, a obrigação de lidar com a ambigüidade de palavras dificulta o processo de agrupamento de termos. Um bom agrupamento envolve, entre outras coisas, a seleção de uma boa medida de similaridade e a existência de um bom algoritmo de inserção de objetos em classes. Mesmo depois disso, deve-se ter ainda a preocupação de que um aumento do *retorno* ocasionado pelo agrupamento pode gerar uma diminuição na *precisão*.

5.6.1 Agrupamento Automático de Termos

Veremos, nesta secção, uma série de algoritmos de agrupamento de termos, cujas utilidades e dificuldades foram discutidas anteriormente. Todos os algoritmos criarão as classes a partir de uma matriz de termos $\times$ documentos. Usaremos a seguinte matriz $m \times n$ como exemplo:

$$A = \begin{bmatrix} 0 & 3 & 3 & 0 & 2 \\ 4 & 1 & 0 & 1 & 2 \\ 0 & 4 & 0 & 0 & 2 \\ 0 & 3 & 0 & 3 & 3 \\ 0 & 1 & 3 & 0 & 1 \\ 2 & 2 & 0 & 0 & 4 \\ 1 & 0 & 3 & 2 & 0 \\ 3 & 1 & 0 & 0 & 2 \end{bmatrix} \tag{5.8}$$

A função de similaridade de dois termos a ser usada aqui é dada por:

$$\text{sim}(\text{termo}\, i, \text{termo}\, j) = \sum_{k=1}^{n} a_{ik} a_{jk} \tag{5.9}$$

Utilizando-se essa função, cria-se uma matriz $\bar{A}$ de termos $\times$ termos, onde cada elemento (i, j) corresponde à similaridade entre um termo i e um termo j. Note que a fórmula acima é reflexiva, ou seja, $\text{sim}(\text{termo}\, i, \text{termo}\, j) = \text{sim}(\text{termo}\, j, \text{termo}\, i)$, então a matriz é simétrica. Além disso, os elementos da diagonal, por relacionarem um termo com ele próprio, serão deixados em branco. No exemplo, essa matriz é dada por:

$$\bar{A} = \begin{bmatrix} & 7 & 16 & 15 & 14 & 14 & 9 & 7 \\ 7 & & 8 & 12 & 3 & 18 & 6 & 17 \\ 16 & 8 & & 18 & 6 & 16 & 0 & 8 \\ 15 & 12 & 18 & & 6 & 18 & 6 & 9 \\ 14 & 3 & 6 & 6 & & 6 & 9 & 3 \\ 14 & 18 & 16 & 18 & 6 & & 2 & 16 \\ 9 & 6 & 0 & 6 & 9 & 2 & & 3 \\ 7 & 17 & 8 & 9 & 3 & 16 & 3 & \end{bmatrix}$$

O próximo passo é selecionar um limiar L que determina se dois termos são suficientemente similares. Cria-se uma nova matriz de termos $\times$ termos $\tilde{A}$ de modo que se $\bar{a}_{ij} \geq L$, então $\tilde{a}_{ij} = 1$ e caso contrário, $\tilde{a}_{ij} = 0$. O resultado do exemplo, considerando $L = 10$, é dado a seguir:

$$\tilde{A} = \begin{bmatrix} & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & \end{bmatrix}$$

O último passo é determinar as classes em que os termos devem pertencer. Para isso, veremos que existem vários algoritmos diferentes, com qualidades e características distintas. A primeira metodologia a ser vista é a do *clique*. Nesse caso, todos os termos agrupados devem ser similares aos demais do grupo. Considere o seguinte algoritmo:

Algoritmo 5.6.1. *(Método do Clique)*

1. Seja $i = 1$;
2. Selecione um termo i e coloque-o em uma nova classe;
3. Comece pelo termo k, onde $r = k = i + 1$;
4. Coloque o termo k na classe corrente se ele tiver relação com todos os termos dessa classe;
5. Senão, $k = k + 1$;
6. Se $k > m$ (número de termos), então:
   $r = r + 1$;
   Se $r = m$, então vá para o passo 7;
   Senão: $k = r$;
      Crie uma nova classe contendo o termo i;
      Vá para o passo 4;
   Senão vá para o passo 4;
7. Se a classe atual tem apenas o termo i e não existem outras classes contendo esse termo, então:
      Apague a classe corrente;
   Senão $i = i + 1$;
8. Se $i = m + 1$, então vá para o passo 9;
   Senão, vá para o passo 2;
9. Elimine classes duplicadas ou que estejam contidas em outra.

Aplicando o algoritmo acima para o nosso exemplo, obtemos as seguintes classes:

Classe 1: termo 1, termo 3, termo 4, termo 6
Classe 2: termo 1, termo 5
Classe 3: termo 2, termo 4, termo 6
Classe 4: termo 2, termo 6, termo 8
Classe 5: termo 7

Note que os termos 1 e 6 aparecem em mais de uma classe, isto é, essa técnica permite inserir termos em mais de uma classe. O algoritmo acima pode ser explicado observando-se um diagrama de relacionamentos de termos. A figura 5.6.1.1 corresponde ao diagrama do nosso exemplo. Em termos de teoria de grafos, cada vértice está associada a um termo e existe uma aresta que liga dois vértices se, e somente se, tais termos correspondentes são similares.

Observe que a técnica do clique [3] tem esse nome justamente porque ela devolve, para cada vértice, o clique maximal [4] que a contém. Note, por exemplo, que o clique maximal que contém o vértice $T1$ (bem como os vértices $T3$, $T4$ e $T6$) é formado pelos termos 1, 3, 4 e 6, o que corresponde a classe 1 formada anteriormente. Os cliques maximais que contém $T2$, por sua vez, são justamente as classes 3 e 4. Por fim, os cliques máximais que contém $T5$ e $T7$ correspondem às classes 2 e 5, respectivamente.

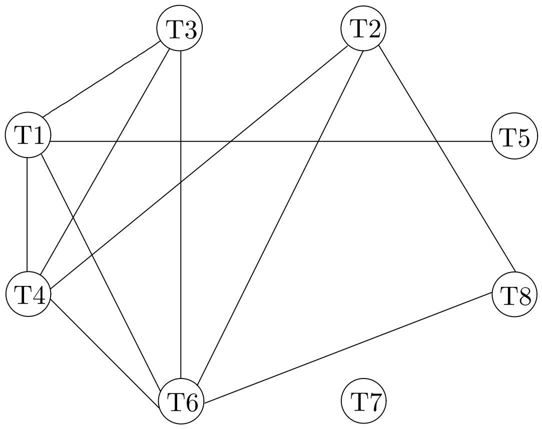


Figura 5.6.1.1: Grafo de Similaridades de Termos.

Uma segunda metodologia é conhecida como *single link*. Nesse caso, um termo similar a qualquer termo da classe pode ser inserida nela. Ao contrário da técnica do clique, ela não permite que termos estejam contidas em mais de uma classe. Seu algoritmo é dado por:

[3] Um clique em um grafo não-dirigido é um conjunto de vértices dois a dois adjacentes.
[4] Um clique C é maximal se não existe um outro clique que seja superconjunto próprio de C.

Algoritmo 5.6.2. *(Método do Single Link)*

1. Selecione um termo que não está numa classe e coloque-o em uma nova classe;
2. Coloque nessa classe todos os outros termos que são relacionados a ele;
3. Para cada termo colocado nessa classe, vá para o passo 2;
4. Quando nenhum termo novo é identificado no passo 2, vá para o passo 1.

Eis o resultado dessa técnica no exemplo:

Classe 1: termo 1, termo 3, termo 4, termo 5, termo 6, termo 2, termo 8
Classe 2: termo 7

Observe que, comparado ao método do clique, o *single link* resulta em poucas classes. Além disso, no nosso exemplo, grande parte dos termos foram agrupados em uma única classe. Vejamos agora a chamada técnica *Star*. Seu algoritmo é dado abaixo:

Algoritmo 5.6.3. *(Método Star)*

1. Se existe um termo i que não está contido em nenhuma classe,
   Coloque-o em uma nova classe;
   Senão termine o algoritmo;
2. Adicione nessa classe todos os termos relacionados ao termo i;
3. Vá para o passo 1.

No algoritmo acima, um termo pode estar contido em múltiplas classes. Essa característica pode ser mudada se tivermos uma condição de adicionar um termo em uma classe apenas se ela não foi inserida anteriormente em outra. Observe também que existem vários conjuntos de classes diferentes que podem ser construídos utilizando-se essa técnica. Se considerarmos no passo 1, a escolha de um termo com menor índice, obtemos o seguinte:

Classe 1: termo 1, termo 3, termo 4, termo 5, termo 6
Classe 2: termo 2, termo 4, termo 8, termo 6
Classe 3: termo 7

Um outro agrupamento possível seria considerar a escolha de uma termo de maior índice no passo 1:

Classe 1: termo 8, termo 2, termo 6
Classe 2: termo 7
Classe 3: termo 5, termo 1

Classe 4: termo 4, termo 1, termo 2, termo 3, termo 6

O último método que veremos relaciona-se ao algoritmo abaixo:

Algoritmo 5.6.4. *(Método do String)*

1. Se existe um termo i que não está contido em nenhuma classe,
   Coloque-o em uma nova classe;
   Senão, termine o algoritmo;
2. Se existe um termo k similar a i e que não está contido na classe corrente,
   Coloque-o nessa classe;
   Senão, vá para o passo 1;
3. $i = k$;
4. Vá para o passo 2.

Observe que assim como a técnica *Star*, o algoritmo *String* pode retornar diferentes conjuntos de classes, dependendo da escolha dos termos no passo 1 e 2. Se o critério de escolha for o termo de menor índice, temos no nosso exemplo, o seguinte resultado:

Classe 1: termo 1, termo 3, termo 4, termo 2, termo 6, termo 8
Classe 2: termo 5
Classe 3: termo 7

Um outro resultado possível é dado a seguir. Nesse caso, o termo inicial é definido como sendo o termo 5 e o critério de escolha no passo 2 continua sendo a de menor índice:

Classe 1: termo 5, termo 1, termo 3, termo 4, termo 2, termo 6, termo 8
Classe 2: termo 7

Note que este agrupamento é idêntico ao dado pelo método do *single link*.

Comparando-se as quatro técnicas vistas, pode-se dizer que a do clique é a que produz maior grau de similaridade entre os termos de uma mesma classe. Ela também é a que fornece, em média, maior número de classes. Em contrapartida, a técnica do *single link* produz o menor número de classes em média e com o menor grau de relação entre os termos de uma classe [Sal72, Sal75, SJ71]. No entanto, esta requer apenas $O(n^2)$ comparações [Cro77], enquanto que a do clique possui um gasto de $O(n^3)$ [KM00].

Observa-se também que o método do *String* é, em média, um pouco melhor que a do *single link*. A técnica *Star*, por sua vez, fornece resultados consideráveis, com desempenho situado entre o método do *String* e do clique. A escolha do algoritmo a ser usado deverá ser feita de acordo com os resultados desejados e com o limite do tempo de processamento que se possui [KM00].

5.6.2 Agrupamento de Termos Usando Classes Existentes

Um método alternativo para criação de classes de termos semelhantes está na utilização de classes já existentes. Tal técnica reduz o número de cálculos de similaridades e está baseada no conceito de *centróide*. Considerando o espaço determinado pelos n vetores de documentos, o centróide seria um ponto n-dimensional que representa o "centro de massa" (ou a média) de alguns desses vetores.

Mais precisamente, um centróide está associado a uma única classe e ele representa a média dos termos contidos nessa classe. A cada passo, a similaridade entre os termos e os centróides são calculados. A seguir, associa-se cada termo com o centróide (ou classe) que ofereceu a maior similaridade. O processo continua até que se estabilize. O gasto desse processo é de apenas $O(n)$.

Apliquemos agora o processo descrito com a matriz 5.8, dada na secção anterior. Suponha que temos as seguintes classes existentes iniciais:

Classe 1: termo 1, termo 2
Classe 2: termo 3, termo 4
Classe 3: termo 5, termo 6

Conforme vimos, um centróide c é um vetor n-dimensional correspondente a uma classe C_k, onde cada elemento do centróide c_i é dado por:

$$c_i = \sum_{\substack{j=1 \\ T_j \in C_k}}^{m} a_{ji}, \qquad i = 1, .., n.$$

Portanto, no exemplo, os centróides de cada classe são:

Classe 1: 4/2, 4/2, 3/2, 1/2, 4/2
Classe 2: 0/2, 7/2, 0/2, 3/2, 5/2
Classe 3: 2/2, 3/2, 3/2, 0/2, 5/2

A similaridade de cada centróide c e cada termo i (T_i) é calculada de modo análogo ao de termos (veja igualdade 5.9 da secção anterior):

$$\text{sim}(c, T_i) = \sum_{j=1}^{n} c_j a_{ij}.$$

O cômputo de todas as similaridades nos fornece uma matriz de classes $\times$ termos abaixo:

$$\begin{bmatrix} 29/2 & 29/2 & 24/2 & 27/2 & 17/2 & 32/2 & 15/2 & 24/2 \\ 31/2 & 20/2 & 38/2 & 45/2 & 12/2 & 34/2 & 6/2 & 17/2 \\ 28/2 & 21/2 & 22/2 & 24/2 & 17/2 & 30/2 & 11/2 & 19/2 \end{bmatrix}$$

Com isso, podemos associar cada termo com a classe que possui o melhor grau de similaridade:

Termo 1	Termo 2	Termo 3	Termo 4	Termo 5	Termo 6	Termo 7	Termo 8
Classe 2	Classe 1	Classe 2	Classe 2	Classe 3	Classe 2	Classe 1	Classe 1

Note que o termo 5 poderia ter sido associado a classe 1 ou 3. Tendo-se as novas classes, podemos calcular os novos centróides:

Classe 1: 8/3, 2/3, 3/3, 3/3, 4/3
Classe 2: 2/4, 12/4, 3/4, 3/4, 11/4
Classe 3: 0/1, 1/1, 3/1, 0/1, 1/1

A nova matriz de similaridade entre termos e centróides é dada por:

$$\begin{bmatrix} 23/3 & 45/3 & 16/3 & 27/3 & 15/3 & 36/3 & 23/3 & 34/3 \\ 67/4 & 45/4 & 70/4 & 78/7 & 33/4 & 72/4 & 17/4 & 40/4 \\ 12/1 & 3/1 & 6/1 & 6/1 & 11/1 & 6/1 & 9/1 & 3/1 \end{bmatrix}$$

A associação de cada termo a uma classe é dada na tabela a seguir:

Termo 1	Termo 2	Termo 3	Termo 4	Termo 5	Termo 6	Termo 7	Termo 8
Classe 2	Classe 1	Classe 2	Classe 2	Classe 3	Classe 2	Classe 3	Classe 1

Nessa última iteração do processo, apenas o termo 7 mudou de classe. Isto é razoável, já que ele realmente não possui relação com os termos da classe 1. Um problema do processo é que o número de classes não pode mudar. Isso pode forçar um termo a estar em uma classe não muito desejável. Apesar disso, a complexidade do algoritmo é bastante motivador.

Vimos alguns métodos de agrupamentos de termos, com e sem classes iniciais. Essas técnicas poderão ser facilmente adaptadas para fazer agrupamento de documentos. Os detalhes poderão ser vistos em [KM00, Capítulo 6], com apoio de [Dam95, Coh95].

5.6.3 Agrupamento de Termos Usando SVD

Conforme vimos anteriormente, para obtermos um bom agrupamento, é necessário termos uma boa medida de similaridade. O exemplo da secção 5.6.1 mostra uma medida simples de similaridade. No contexto do modelo vetorial e, particularmente do LSI, essa medida será dada pelo cosseno do ângulo entre os vetores de termos. Considere um par de termos i e j. A similaridade será, portanto:

$$\text{sim}(\text{termo } i, \text{termo } j) = \cos(\bar{a}_{ij}) = \frac{(e_i^T A)(A^T e_j)}{\|A^T e_i\|_2 \|A^T e_j\|_2},$$

para $i, j = 1, ..., m$. Utilizando agora uma aproximação de A de posto k e sua decomposição SVD correspondente, $U_k \Sigma_k V_k^T$, obtemos:

$$\cos(\bar{a}_{ij}) = \frac{(e_i^T U_k \Sigma_k V_k^T)(V_k \Sigma_k U_k^T e_j)}{\|V_k \Sigma_k U_k^T e_i\|_2 \|V_k \Sigma_k U_k^T e_j\|_2} = \frac{(e_i^T U_k \Sigma_k)(\Sigma_k U_k^T e_j)}{\|\Sigma_k U_k^T e_i\|_2 \|\Sigma_k U_k^T e_j\|_2}.$$

para $i, j = 1, ..., m$. Seja $s_j \doteq \Sigma_k U_k^T e_j$. Então,

$$\cos(\bar{a}_{ij}) = \frac{s_i^T s_j}{\|s_i\|_2 \|s_j\|_2}, \qquad i, j = 1, ..., m.$$

Com isso, pode-se criar a matriz $\bar{A}$ de similaridades entre termos, aplicar os algoritmos vistos na secção 5.6.1 e assim obter um agrupamento de termos. Geometricamente, o cálculo de similaridades acima mostra que dois vetores de termos são relacionados se eles estão suficientemente próximos no espaço gerado pelas linhas da matriz de termos $\times$ documentos. Além disso, eles não têm relação alguma se forem ortogonais entre si.

5.7 Expansão de Pesquisa

Para que um sistema de IR seja eficiente, ele deve obter alta precisão e alto retorno durante as pesquisas, ou seja, deve identificar todos os documentos relevantes sem retornar os não relevantes. Sabemos, no entanto, que isso não é possível devido às "incertezas" do banco de dados e da própria pesquisa. Ainda assim, é possível aumentar essa precisão com técnicas baseadas na *expansão (ou transformação) de pesquisa*, que consiste em substituir o vetor de pesquisa original por um outro através da análise prévia de documentos e termos da coleção.

A expansão de pesquisa automática tem sido objeto de estudo de pesquisadores de IR nas últimas décadas. A idéia mais simples está em obter documentos relevantes através da expansão baseada na co-ocorrência de termos. Os vários métodos propostos podem ser classificados em quatro grupos listados abaixo:

a) Utilização simples da co-ocorrência: Inicialmente, calcula-se as similaridades entre termos e os agrupa em classes. Para cada termo do vetor de pesquisa, adiciona-se todos os termos que fazem parte da sua classe [MWZ72, SJB71]. Ao final do processo, o número de termos na pesquisa pode ser muito alto, sem que todas elas sejam de fato relacionadas. Por essa razão, esse método é considerado "ingênuo" [MWZ72, SJ91].

b) Utilização da classificação de documentos: Inicialmente, os documentos são classificados utilizando-se de um algoritmo dado. Os termos freqüentes encontrados em uma classe de documentos são considerados similares e colocados em uma classe de termos. Substitui-se, então, o termo pela sua classe na indexação de documentos e pesquisas [Cro90]. Nesse caso, a eficiência depende de parâmetros difíceis de determinar. Além disso, o custo pode ser alto, por ser proporcional ao número de documentos da coleção [CY92].

c) Uso do contexto sintático: A relação entre termos é gerada com base na lingüística e estatísticas relacionadas a co-ocorrência [Gre92]. Nesse caso, usa-se a gramática e o dicionário para extrair, para cada termo, uma lista de termos que estão relacionados a ele. As similaridades são, portanto, calculadas baseando-se em tais listas e a expansão da pesquisa é feita adicionando-se os termos mais similares à pesquisa. Experimentos mostram que a utilização desse novo vetor de pesquisa oferece uma melhora muito pequena em relação ao original [Gre92].

d) Uso de informações relevantes: Informações relevantes são usadas para construir estruturas globais, tais como *thesaurus* [Sal72, Sal80, Qiu92] e árvore geradora máxima [SvCR83]. A expansão de pesquisa é baseada nas informações dadas por essas estruturas. A eficiência deste método depende, por exemplo, das informações relevantes dadas pelo usuário na sua pesquisa.

Uma abordagem conhecida é a expansão semi-automática de pesquisas [ERW92, HB92]. Nela, o usuário escolhe os termos a serem adicionados através da lista de termos fornecidos por um dos métodos acima. Discutiremos essa técnica para o modelo vetorial na secção 5.7.1.

Abordagens totalmente automáticas podem envolver dois tipos de estratégias distintas: a global [BYRN99] e a local [AF77]. Na primeira, todos os documentos do sistema são utilizados para determinar um *thesaurus* de termos. Na outra, os documentos retornados com uma dada pesquisa são examinados em tempo de execução para que se possa determinar os termos utilizados na modificação de pesquisa. A expansão local será discutida na secção 5.7.2.

Na secção 5.7.3 veremos métodos modernos de expansão global de pesquisa, baseados na adição de termos que são mais similares ao *conceito* de uma pesquisa ao invés de termos mais similares aos termos dessa pesquisa [QF95, QF93]. Este capítulo se encerra com a utilização do SVD nesse contexto, na secção 5.7.4.

5.7.1 Expansão Semi-Automática

Na expansão semi-automática de pesquisas, o usuário recebe uma lista de documentos retornados para uma pesquisa e, depois de examiná-las, seleciona aqueles que considera realmente relevantes. Isso faz com que novos termos importantes sejam selecionados e associados à nova pesquisa. No modelo vetorial, reformulamos o vetor de pesquisa de modo com que este fique mais próximo do espaço dado pelos documentos relevantes.

Antes de indicar o processo de expansão, considere as seguintes definições: D_r como o conjunto de documentos retornados pela pesquisa original e identificados pelo usuário como sendo relevantes; D_n como sendo o conjunto de documentos não relevantes entre os documentos retornados; e C_r como o conjunto dos documentos relevantes entre todos os documentos da coleção. Além disso, sejam $|D_r|$, $|D_n|$ e $|C_r|$ os números de documentos contidos em cada um desses conjuntos.

Suponhamos inicialmente que se conhece todos os documentos relevantes da coleção para uma pesquisa p. Seja n o número total de documentos da coleção. Um vetor de pesquisa novo que, nesse caso, diferencia os documentos relevantes dos não relevantes, é dado a seguir:

$$p_{exp} = \frac{1}{|C_r|} \sum_{d_j \in C_r} d_j - \frac{1}{n - |C_r|} \sum_{d_j \notin C_r} d_j. \tag{5.10}$$

O problema dessa formulação é que não conhecemos C_r a priori. A idéia é, então, modificar o vetor de pesquisa original e utilizar os conjuntos D_r e D_n que conhecemos. Uma adaptação da equação 5.10, proposta por Rochio [Roc71], é dada por:

$$p_{exp} = \alpha\, p + \frac{\beta}{|D_r|} \sum_{d_j \in D_r} d_j - \frac{\gamma}{|D_n|} \sum_{d_j \in D_n} d_j,$$

sendo α, β e γ constantes positivos definidos pelo sistema.

O termo αp aparece na formulação porque, na prática, o vetor p original contém algumas informações importantes. Além disso, usualmente as informações contidas nos documentos relevantes são mais importantes que dos não relevantes. Isto sugere uma atribuição de γ a um escalar menor que β. Outras duas formas clássicas de expansão de pesquisa foram dadas por Ide [Ide71]:

$$p_{exp} = \alpha\, p + \beta \sum_{d_j \in D_r} d_j - \gamma \sum_{d_j \in D_n} d_j,$$

$$p_{exp} = \alpha\, p + \beta \sum_{d_j \in D_r} d_j - \gamma \max_{d_j \in D_n} d_j,$$

onde $\max_{d_j \in D_n} d_j$ se refere ao documento mais não relevante possível. Considera-se atualmente que as três formas de expansão indicadas oferecem resultados semelhantes. Para obter mais informações em relação a essas formulações, veja [Ide71].

As principais vantagens dessa expansão semi-automática está na simplicidade e nos resultados razoavelmente bons, comprovados experimentalmente. A desvantagem, por sua vez, está na necessidade de intervenção de um usuário no processo.

5.7.2 Expansão Automática Local

Uma outra abordagem de modificação de pesquisa consiste em obter uma classe de documentos relevantes em tempo de execução e sem a intervenção do usuário, ou seja, de modo totalmente automático. O sistema identifica os termos que estão relacionados aos termos contidos na pesquisa original e elabora uma estrutura a partir dos documentos retornados. Indicaremos três tipos de agrupamentos de termos possíveis.

A primeira, denominada de *agrupamento por associação*, é baseada na co-ocorrência de termos em documentos. Seja D_t o conjunto dos documentos retornados com a pesquisa original. A idéia é que termos que co-ocorrem freqüentemente em documentos pertencentes a D_t podem ter significados próximos. Considere a matriz de termos × documentos usual, ou seja, com a_{ij} indicando o peso do termo t_i no item d_j. A matriz $S \doteq AA^T$ é denotada matriz de similaridade local de termos. Cada elemento s_{uv} indica a correlação $\text{sim}(t_u, t_v)$ entre termos dada por:

$$\text{sim}(t_u, t_v) = \sum_{d_j \in D_t} a_{uj} a_{vj}.$$

Podemos definir a matriz S como sendo não normalizada, ou seja, $s_{uv} \doteq \text{sim}(t_u, t_v)$. Para S normalizado, temos:

$$s_{uv} \doteq \frac{\text{sim}(t_u, t_v)}{\text{sim}(t_u, t_u) + \text{sim}(t_v, t_v) - \text{sim}(t_u, t_v)}.$$

O agrupamento por associação está baseado na co-ocorrência de termos, mas não leva em consideração onde os termos ocorrem em um documento. Por outro lado, o *agrupamento métrico* está baseado na idéia de que dois termos que co-ocorrem em uma mesma sentença são mais relacionados que dois termos de um documento distantes entre si.

Seja $V(s)$ o conjunto de palavras semanticamente idênticas a um radical s mas gramaticalmente distintas. Exemplificando, se $s = filme$, então:

$$V(s) = \{filme, filmar, filmando, filmado, ...\}.$$

Considere ainda $d(k_i, k_j)$ como sendo a distância entre as palavras k_i e k_j, dada pelo número de palavras do documento contidas entre esses termos. Se k_i e k_j não estiverem em um mesmo documento, $d(k_i, k_j) = \infty$. A similaridade entre termos t_u e t_v é a seguinte:

$$\text{sim}(t_u, t_v) = \sum_{k_i \in V(t_u)} \sum_{k_j \in V(t_v)} \frac{1}{d(k_i, k_j)}.$$

Da mesma forma que vimos anteriormente, podemos definir a matriz de similaridade local S como sendo não normalizada, ou seja, $s_{uv} \doteq \text{sim}(t_u, t_v)$. Para S normalizado, temos:

$$s_{uv} \doteq \frac{\text{sim}(t_u, t_v)}{|V(t_u)|\,|V(t_v)|}$$

O terceiro tipo de método é chamado *agrupamento escalar* e sua idéia é que duas palavras com vizinhanças parecidas são similares entre si. Para isso, é necessário termos uma matriz S pré-estabelecida. Considere s_i como o vetor contendo os n maiores valores da i-ésima linha de S, para um dado n. Quantificar vizinhanças de dois termos s_u e s_v é medir a similaridade entre eles. Uma maneira de fazer isso é verificar o cosseno do ângulo entre esses vetores. A nova matriz S é tal que:

$$s_{uv} \doteq \cos(s_u, s_v) = \frac{s_u^T s_v}{\|s_u\|_2 \|s_v\|_2}.$$

Construindo-se a matriz de similaridade local S, pode-se usar um dos agoritmos vistos na secção 5.6.1 para estruturar as classes de termos semelhantes. A modificação de pesquisa será dada a partir dessas classes, adicionando-se no novo vetor de pesquisa alguns elementos de classes correspondentes aos termos da pequisa original.

Na expansão automática local, operações são realizadas em tempo de execução. O acesso ao texto dos documentos retornados, no entanto, consome um tempo considerável, o que faz com que essa estratégia não seja recomendada para todos os tipos de sistemas. Na internet, por exemplo, o tempo consumido deve ser obrigatoriamente pequeno, já que várias pesquisas devem ser processadas em um mesmo segundo. No entanto, a abordagem pode ser útil em intranets ou em sistemas contendo documentos especializados.

5.7.3 Expansão Automática Global

Uma outra abordagem de modificação de pesquisa consiste em construir as similaridades entre termos utilizando-se da coleção total de documentos. Até o começo da década de 90, a análise global era considerada um fracasso para coleções gerais. Entretanto, com o aparecimento de variações modernas, a metodologia tornou-se promissora. Tais variações se baseiam na utilização de termos mais similares ao *conceito* de uma pesquisa.

Vimos, até agora, que cada documento está associado a vários termos, que oferecem, em conjunto, o *conceito* de tal documento. Uma idéia alternativa está na importância que um documento tem na representação de um termo. Note, no entanto, que o fato de um termo descrever adequadamente um documento não implica necessariamente que esse documento representa bem o significado de tal termo.

Com essas observações, pode-se criar uma nova estrutura de comparação de termos, denotada de *thesaurus de similaridades*. Ao contrário das matrizes de termos $\times$ termos que vimos anteriormente, essa estrutura está baseada na maneira com que os termos de uma coleção são indexados pelos documentos. Em outras palavras, os termos fazem o papel de itens a serem retornados e os documentos são os indexadores desses termos.

A construção do *thesaurus* envolve, então, a estimativa do peso de um documento na representação do significado de um termo. Os seguintes itens são importantes para isso:

a) Um documento pequeno tem papel mais importante que um longo, já que este possui, em média, mais tópicos. Ou seja, se dois termos co-ocorrem em um documento grande, a probabilidade desses dois termos serem similares é menor que no caso de co-ocorrência em um documento pequeno.

b) Quanto maior o número de ocorrências de um termo em um documento, maior a probabilidade desse documento representar o significado de tal termo.

c) Se um termo aparece em vários documentos, ele pode tanto representar um único conceito, quanto ter vários significados diferentes.

Sejam m o número de termos na coleção, n o número de documentos e f_{ij} a freqüência de ocorrência do termo t_i no documento d_j. Considere ainda $|d_j|$ como o número de termos indexadores distintos do documento d_j. A freqüência inversa de termos de um documento d_j é definido por:

$$\mathrm{itf}(d_j) \doteq \log \frac{m}{|d_j|}, \qquad j = 1, ..., n.$$

Uma maneira [BYRN99, QF95] de definir o peso de um documento d_k na representação de um termo t_i é dada a seguir:

$$w(t_i, d_k) \doteq \begin{cases} \Big(\delta(t_i, d_k)\ \mathrm{itf}(d_k)\Big) \Big/ \sqrt{\sum_{d_j \in t_i} \big(\delta(t_i, d_j)\ \mathrm{itf}(d_j)\big)^2}, & \text{se } f(t_i, d_k) > 0. \\ 0, & \text{caso contrário.} \end{cases}$$

onde $\delta(t_i, d_j) \doteq 0.5 + 0.5\frac{f(t_i,d_j)}{\max f(t_i)}$ e $\max f(t_i) \doteq \max\{f(t_i, d_k) | d_k \in t_i\}$. A partir disso, representa-se um termo no espaço vetorial de documentos, representado por todos os itens da coleção:

$$t_i = \begin{bmatrix} w(t_i, d_1) & w(t_i, d_2) & \dots & w(t_i, d_n) \end{bmatrix}^T.$$

Finalmente, a similaridade entre termos t_i e t_j pode ser dada pelo produto interno entre os vetores correspondentes:

$$\mathrm{sim}(t_i, t_j) \doteq t_i^T t_j = \sum_{k=1}^{n} w(t_i, d_k)\, w(t_j, d_k) = \sum_{d_k \in\, t_i \cap t_j} w(t_i, d_k)\, w(t_j, d_k).$$

Observe que os vetores de termos são normalizados. Isso significa que a similaridade acima é igual ao cosseno do ângulo entre esses vetores. Assim, construímos a matriz S de *thesaurus* de similaridades global com

$$s_{ij} \doteq \mathrm{sim}(t_i, t_j), \qquad i, j = 1, ..., m.$$

Considere agora $p = [p_1, p_2, ..., p_m]^T$ como sendo o vetor de pesquisa no espaço vetorial dos termos da coleção. Assim como a matriz de *thesaurus* de similaridades, mapeamos o vetor p no espaço vetorial de documentos. Dessa forma, pode-se estimar as similaridades entre os termos e a pesquisa. A representação do vetor de pesquisa no espaço de termos é a seguinte:

$$p_c = \sum_{t_i \in p} p_i\, t_i.$$

Sabendo que p e t são normalizados, a similaridade entre eles é dada pelo cosseno do ângulo ou o produto interno entre p_c e t, ou seja:

$$\mathrm{sim}(p, t) \doteq p_c^T t = \left(\sum_{t_i \in p} p_i\, t_i \right) t = \sum_{t_i \in p} p_i\, (t_i^T t) = \sum_{t_i \in p} p_i\ \mathrm{sim}(t_i, t).$$

Todos os termos da coleção podem ser ordenados de acordo com os valores de $\mathrm{sim}(p, t_i)$, $i = 1, ..., m$ calculados. Seja R um conjunto de cardinalidade r contendo os termos de valores mais altos. Tais termos podem, então, serem adicionados no vetor de pesquisa. A razão de estabelecer r ao invés de escolher os termos maiores que um limiar é que nesse último caso o número de termos adicionados não é fixo, podendo ser incofortavelmente grande.

Os pesos dos termos adicionados podem ser dados da seguinte forma:

$$\theta(p, t) = \frac{\mathrm{sim}(p, t)}{\sum_{t_i \in p} p_i},$$

o isto implica que $0 \leq \theta(p, t) \leq 1$. O vetor de pesquisa a ser adicionado será dado por:

$$p_e = \left[\begin{array}{cccc} p_e^1 & p_e^2 & \ldots & p_e^m \end{array}\right]^T.$$

onde

$$p_e^i \doteq \begin{cases} \theta(p, t_i), & \text{se } t_i \in R \\ 0, & \text{caso contrário.} \end{cases}$$

Finalmente, o novo vetor de pesquisa é dado por

$$p_{exp} = \alpha\, p + \beta\, p_e, \qquad \alpha, \beta \in \mathbb{R}.$$

Os valores de α e β são definidas experimentalmente. No entanto, muitos pesquisadores da área adotam $\alpha = \beta = 1$ com o argumento de que valores ótimos desses parâmetros encontrados para uma classe de pesquisa não necessariamente continuam ótimos para uma outra classe. Mostramos, dessa forma, como fazer a expansão de pesquisa global usando um *thesaurus* de similaridades.

Discutiremos agora uma extensão dessa técnica proposta por Qiu e Frei [QF95]. No modelo original, o conceito de uma pesquisa é representada pelo seu centróide. Sabemos, no entanto, que a pesquisa pode conter "incertezas", e isto faz com que o centróide achado não corresponda necessariamente ao conceito que queremos. Em outras palavras, é preferível que alguns dos termos da pesquisa sejam inutilizados. Determinar tais termos não é fácil. Considere as duas suposições a seguir de um sistema IR:

1. A distribuição dos termos de pesquisa em documentos relevantes é diferente da distriubuição dada pelos não relevantes.

2. Os termos similares a um termo de um documento relevante são mais prováveis de ocorrerem em documentos relevantes. Da mesma forma, termos similares a um termo de um documento não relevante têm maior probabilidade de ocorrerem em documentos não relevantes.

Seja Z o conjunto dos documentos mais relevantes possíveis, com cardinalidade definida. A idéia do método é indicado pelos passos a seguir: (i) Para um pesquisa p, ordene os documentos retornados em ordem de relevância e a partir disto crie o conjunto Z. (ii) Agrupe em G_t os termos da pesquisa original que ocorrem em documentos de Z (estes são considerados termos "bons"). (iii) Encontre a similaridade $\text{sim}(p,t)$ entre uma pesquisa p e cada termo $t \in G_t$. (iv) Crie o conjunto de termos com maiores valores de $\text{sim}(p,t)$ e a partir disso estabeleça $\theta(p,t)$, $t \in G_t$ da mesma forma que vimos anteriormente. (v) Crie q_e e faça a expansão de pesquisa utilizando-se sempre termos de G_t.

As abordagens descritas nesta secção foram testadas por Qiu e Frei e o resultado é bem satisfatório. Conforme se previa, a extensão do modelo de expansão global automática melhorou a performance durante a busca. Informações detalhadas das experiências podem ser obtidas em [QF95].

5.7.4 Expansão Automática Usando SVD

Nesta secção, descreveremos como fazer a expansão de pesquisa utilizando-se da decomposição SVD já calculada. Considere A_k a aproximação de posto k da matriz de termos $\times$ documentos da coleção e p um vetor de pesquisa. Inicialmente, projeta-se o vetor p no espaço gerado pelas colunas de A_k, ou seja, toma-se $p \leftarrow U_k U_k^T p$. Isso é feito para que a pesquisa e os documentos possam ser descritos na mesma base desse espaço.

Observe que as coordenadas do vetor de pesquisa são os elementos do vetor $U_k^T p$ e as coordenadas da j-ésima coluna de $U_k \Sigma_k V_k^T e_j$ da matriz A_k são os elementos do vetor $\Sigma_k V_k^T e_j$. Considere primeiro o caso em que há apenas um documento a_j retornado com a pesquisa original. Uma maneira simples de definir o novo vetor de pesquisa é:

$$p_{exp} \doteq \alpha\, U_k U_k^T p + \beta\, a_j = \alpha\, U_k U_k^T p + \beta\, U_k \Sigma_k V_k^T e_j = U_k(\alpha\, U_k^T p + \beta\, \Sigma_k V_k^T e_j),$$

onde α e β são escalares definidos pelo sistema, da mesma forma ao que vimos na secção anterior. Para um número qualquer de documentos retornados, basta definirmos a expansão de pesquisa da seguinte forma:

$$p_{exp} \doteq \alpha\, p + \beta \sum_{j=1}^{n} w_j a_j = \alpha\, U_k(U_k^T p + \beta\, \Sigma_k V_k^T w), \tag{5.11}$$

onde w_j, $j = 1, .., n$ é igual a 1 se a_j é relevante e 0 caso contrário.

Uma variação de 5.11 considera escalares β diferentes para cada documento retornado. A idéia é que documentos de similaridades com a pesquisa p mais altas tenham um valor de β maior que os documentos menos relevantes. Assim, temos:

$$p_{exp} \doteq \alpha\, p + \sum_{j=1}^{n} \beta_j w_j a_j = U_k \left(\alpha\, U_k^T p + \sum_{j=1}^{n} \beta_j w_j \Sigma_k V_k^T e_j \right),$$

com β_j, $j = 1, ..., n$ correspondendo a um documento a_j.

Sabe-se que o novo vetor de pesquisa p_{exp} definido pode aumentar a performance do sistema baseado no LSI. Análises de alguns experimentos podem ser vistos em [Dum91].

5.8 Gerenciamento de Coleções Dinâmicas

Grande parte dos sistemas de IR precisam lidar com as constantes mudanças do seu banco de dados. Na Internet, por exemplo, informações são retiradas ou adicionadas a cada segundo. Para o modelo LSI, essas atualizações podem ser facilmente dadas recomputando-se o SVD da matriz de termos $\times$ documentos. Essa estratégia, no entanto, só seria recomendada para coleções bem pequenas, já que o custo para isso é alto em tempo e em espaço.

O primeiro método para gerenciar coleções dinâmicas é denominado *folding-in*. Sua aplicação é computacionalmente barata, mas ele devolve uma representação inexata do banco de dados. Esse método será discutido na secção 5.8.1. Uma outra técnica, que será dada na secção 5.8.2, é denominada *SVD-updating* e, apesar de ter um custo maior que a anterior, preserva a representação do banco de dados. O capítulo termina com a secção 5.8.3, onde teremos justificativas teóricas das metodologias apresentadas em 5.8.2.

5.8.1 Folding-In

O fato do processo *folding-in* resultar em uma representação inexata do banco de dados faz com que ele seja apropriado apenas ocasionalmente. Porém, se combinarmos essa técnica com o *SVD-updating*, seu uso poderá ser mais freqüente. Considere, por exemplo, um sistema de uma empresa utilizado por funcionários durante o dia. Supondo que a quantidade de dados retirados ou armazenados seja grande, o *folding-in* poderia ser aplicado a cada hora e o *SVD-updating* somente à noite, quando poucos funcionários trabalham.

O primeiro passo do *folding-in* de um documento $\hat{d}$ é projetar esse vetor no espaço gerado pelas colunas da matriz de termos $\times$ documentos. Seja $A_k = U_k \Sigma_k V_k^T$ a aproximação de posto k dessa matriz. A projeção d de $\hat{d}$ em tal espaço é dado por $d = U_k U_k^T \hat{d}$. Note que as coordenadas de d na base U_k são dadas pelos elementos do vetor $U_k^T \hat{d}$ e que as coordenadas da i-ésima coluna de A_k são os elementos de $\Sigma_k V_k^T e_i$.

O novo documento é, então, adicionado na coleção guardando-se o vetor $U_k^T \hat{d}$ como uma nova coluna de $\Sigma_k V_k^T$. Esta matriz, no entanto, não é computada explicitamente. A solução para isso está em adicionar $\hat{d}^T U_k \Sigma_k^{-1}$ como uma nova linha de V_k, formando a nova matriz $\bar{V}_k$. O produto $\Sigma_k \bar{V}_k$ dado implicitamente é o resultado desejado.

Observe que a matriz $\bar{V}_k$ não é ortogonal e que o espaço gerado pelas colunas dessa matriz não representa o espaço gerado pelas linhas da nova matriz de termos $\times$ documentos. Além disso, para casos em que o vetor d é quase ortogonal às colunas de U_k, a maioria das informações contidas de $\hat{d}$ são perdidas durante a projeção.

Considere agora o problema de adicionar um novo termo $\hat{t}$ na coleção. A idéia para isso é análoga a de documentos. Inicialmente, projeta-se o vetor $\hat{t}$ no espaço gerado pelas linhas de A_k, ou seja, $t = V_k V_k^T \hat{t}$. A adição do termo será dada, portanto, com a colocação de $V_k^T \hat{t}$ em U_k como uma nova linha. Isso forma a matriz não ortogonal $\bar{U}_T$, cujos espaços gerados pelas suas colunas diferem do espaço gerado pelas colunas da nova matriz de termos $\times$ documentos. Análises detalhadas do *folding-in* podem ser obtidas em [BDO95, O'B94].

5.8.2 SVD-Updating

Veremos nesta secção como realizar a atualização de coleções dinâmicas com modelo LSI utilizando-se de uma técnica conhecida como *SVD-updating* [BDJ99, BDO95, ZS99]. Ao contrário do *folding-in*, essa abordagem mantém a ortogonalidade das matrizes. Além disso, o espaço gerado pelas colunas da matriz de vetores singulares a esquerda é a mesma do espaço gerado pelas colunas da matriz de termos $\times$ documentos. O mesmo ocorre com as linhas dessa matriz e a matriz de vetores singulares à direita. O *SVD-updating* abrange três possíveis passos: atualização de termos, de documentos e de pesos dos termos.

Atualização de Termos

Considere $A_k = U_k \Sigma_k V_k^T \in \mathbb{R}^{m \times n}$ a aproximação de posto k da matriz de termos $\times$ documentos da coleção. Para uma adição de q termos, monta-se uma matriz $T \in \mathbb{R}^{q \times n}$, onde cada linha corresponde a um desses termos. A nova matriz de termos $\times$ documentos de dimensão $(m+q) \times n$ é dada por:

$$B = \begin{bmatrix} A_k \\ T \end{bmatrix}.$$

Mostraremos agora como obter B_k, i.e., a aproximação de B de posto k eficientemente, sem precisar recalcular a decomposição SVD. Utilizando o SVD de A_k que conhecemos, temos as seguintes igualdades:

$$\begin{aligned}
B &= \begin{bmatrix} A_k \\ T \end{bmatrix} = \begin{bmatrix} U_k \Sigma_k V_k^T \\ T \end{bmatrix} = \begin{bmatrix} U_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Sigma_k V_k^T \\ T \end{bmatrix} \\
&= \begin{bmatrix} U_k & 0 \\ 0 & I \end{bmatrix} \left(\begin{bmatrix} \Sigma_k \\ T V_k \end{bmatrix} V_k^T + \begin{bmatrix} 0 \\ T(I - V_k V_k^T) \end{bmatrix} \right) \\
&= \begin{bmatrix} U_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ T V_k & I \end{bmatrix} \begin{bmatrix} V_k^T \\ T(I - V_k V_k^T) \end{bmatrix} \\
&= \begin{bmatrix} U_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ T V_k & I \end{bmatrix} \begin{bmatrix} V_k & (I - V_k V_k^T) T^T \end{bmatrix}^T.
\end{aligned}$$

Dessa forma, B é dada como produto de três matrizes, sendo que a primeira é ortogonal e a segunda é triangular inferior. Seja $\tilde{V}_k \doteq (I - V_k V_k^T) T^T$. Observa-se que, apesar de $\tilde{V}_k$

não ser ortogonal, suas colunas pertencem à projeção ortogonal complementar do espaço gerado pelas colunas de T^T. Seja r_v o posto de $\tilde{V}_k$. Para transformar $\tilde{V}_k$ em uma matriz ortogonal, usa-se sua fatoração QR com pivotamento de colunas, dada por $\tilde{V}_k P_v = \hat{V}_k R_q$, onde $P_v \in \mathbb{R}^{q\times q}$ é uma matriz de permutação, $\hat{V}_k \in \mathbb{R}^{d\times r_v}$ é ortogonal e $R_q \in \mathbb{R}^{r_v\times q}$ é triangular superior. Assim,

$$\left[\begin{array}{cc} V_k & (I - V_k V_k^T)T^T \end{array}\right] = \left[\begin{array}{cc} V_k & \hat{V}_k R_q P_v^T \end{array}\right],$$

e B pode ser escrito na seguinte forma:

$$\begin{aligned} B &= \left[\begin{array}{cc} U_k & 0 \\ 0 & I \end{array}\right] \left[\begin{array}{cc} \Sigma_k & 0 \\ TV_k & I \end{array}\right] \left[\begin{array}{cc} V_k & \hat{V}_k R_q P_v^T \end{array}\right]^T \\ &= \left[\begin{array}{cc} U_k & 0 \\ 0 & I \end{array}\right] \left[\begin{array}{cc} \Sigma_k & 0 \\ TV_k & P_v R_q^T \end{array}\right] \left[\begin{array}{c} V_k^T \\ \hat{V}_k^T \end{array}\right]. \end{aligned}$$

Note que agora tanto a matriz da direita quanto da esquerda são ortogonais. Definimos, agora, a decomposição SVD da segunda matriz do produto por

$$\left[\begin{array}{cc} \Sigma_k & 0 \\ TV_k & P_v R_q^T \end{array}\right] \doteq \left[\begin{array}{cc} W_k & W_k^{\perp} \end{array}\right] \left[\begin{array}{cc} \hat{\Sigma}_k & 0 \\ 0 & \hat{\Sigma}_q \end{array}\right] \left[\begin{array}{cc} Q_k & Q_k^{\perp} \end{array}\right]^T,$$

com $W_k \in \mathbb{R}^{(k+q)\times k}$ e $Q_k \in \mathbb{R}^{(k+r_v)\times k}$ ortogonais e $\hat{\Sigma}_k \in \mathbb{R}^{k\times k}$. Portanto, a melhor aproximação de posto k de B é dada tomando-se $\hat{\Sigma}_q = 0$:

$$B_k = \left[\begin{array}{cc} U_k^T & 0 \\ 0 & I_q \end{array}\right] W_k\, \hat{\Sigma}_k \big(\left[\begin{array}{cc} V_k & \hat{V}_k \end{array}\right] Q_k \big)^T.$$

E assim obtemos $B_k = U_B \Sigma_B V_B^T$ tais que:

$$U_B = \left[\begin{array}{cc} U_k^T & 0 \\ 0 & I_q \end{array}\right] W_k, \qquad V_B = \left[\begin{array}{cc} V_k & \hat{V}_k \end{array}\right] Q_k \qquad \text{e} \qquad \Sigma_B = \hat{\Sigma}_k.$$

Atualização de Documentos

O método de adição de documentos novos na coleção é análogo a de termos. Seja $D \in \mathbb{R}^{m\times p}$ a matriz cujas colunas representam os p documentos adicionados. A matriz de termos $\times$ documentos resultante é a seguinte:

$$B = \left[\begin{array}{cc} A_k & D \end{array}\right].$$

Note que $B \in \mathbb{R}^{m\times(n+p)}$ é uma matriz de posto maior ou igual a k. Usando a decomposição SVD de A_k, temos:

$$\begin{aligned} B &= \left[\begin{array}{cc} A_k & D \end{array}\right] = \left[\begin{array}{cc} U_k\Sigma_k V_k^T & D \end{array}\right] = \left[\begin{array}{cc} U_k\Sigma_k & D \end{array}\right] \left[\begin{array}{cc} V_k^T & 0 \\ 0 & I \end{array}\right] \\ &= \Big(U_k \left[\begin{array}{cc} \Sigma_k & U_k^T D \end{array}\right] + \left[\begin{array}{cc} 0 & (I - U_k U_k^T)D \end{array}\right]\Big) \left[\begin{array}{cc} V_k^T & 0 \\ 0 & I \end{array}\right] \\ &= \left[\begin{array}{cc} U_k & (I - U_k U_k^T)D \end{array}\right] \left[\begin{array}{cc} \Sigma_k & U_k^T D \\ 0 & I \end{array}\right] \left[\begin{array}{cc} V_k^T & 0 \\ 0 & I \end{array}\right]. \end{aligned}$$

Considere a fatoração QR com pivotamento de $\tilde{U}_k \doteq (I - U_k U_k^T)D$, de posto r_u dada por:

$$\tilde{U}_k P_u = (I - U_k U_k^T) D P_u = \hat{U}_k R_p,$$

onde $P_u \in \mathbb{R}^{p\times p}$ é uma matriz de permutação, $R_p \in \mathbb{R}^{r_u\times p}$ é triangular superior e $\hat{U}_k^T \in \mathbb{R}^{m\times r_u}$ é ortogonal. Desse modo, pode-se escrever:

$$B = \left[\begin{array}{cc} A_k & D \end{array}\right] = \left[\begin{array}{cc} U_k & \hat{U}_k \end{array}\right] \left[\begin{array}{cc} \Sigma_k & U_k^T D \\ 0 & R_p P_u^T \end{array}\right] \left[\begin{array}{cc} V_k^T & 0 \\ 0 & I_p \end{array}\right].$$

Definindo o SVD da segunda matriz do produto acima, temos:

$$\left[\begin{array}{cc} \Sigma_k & U_k^T D \\ 0 & R_p P_u^T \end{array}\right] \doteq \left[\begin{array}{cc} W_k & W_k^\perp \end{array}\right] \left[\begin{array}{cc} \hat{\Sigma}_k & 0 \\ 0 & \hat{\Sigma}_p \end{array}\right] \left[\begin{array}{cc} Q_k & Q_k^\perp \end{array}\right]^T,$$

com $W_k \in \mathbb{R}^{(k+p)\times k}$ e $Q_k \in \mathbb{R}^{(k+p)\times k}$ ortogonais e $\hat{\Sigma}_k \in \mathbb{R}^{k\times k}$ diagonal. Então, a melhor aproximação de posto k de B é dada por:

$$B_k = \left[\begin{array}{cc} U_k & \hat{U}_k \end{array}\right] W_k \, \hat{\Sigma}_k \left(\left[\begin{array}{cc} V_k & 0 \\ 0 & I_p \end{array}\right] Q_k \right)^T.$$

Por fim, obtemos $B_k = U_B \Sigma_B V_B^T$ tais que:

$$U_B = \left[\begin{array}{cc} U_k & \hat{U}_k \end{array}\right] W_k, \qquad V_B = \left[\begin{array}{cc} V_k & 0 \\ 0 & I_p \end{array}\right] Q_k \qquad \text{e} \qquad \Sigma_B = \hat{\Sigma}_k.$$

Atualização de Pesos de Termos

Sejam j o número de termos cujos pesos serão modificados, $Z \in \mathbb{R}^{n\times j}$ uma matriz que especifica as diferenças entre os pesos antigos e novos dos j termos, e $Y \in \mathbb{R}^{m\times j}$ uma matriz de seleção [5] que indica os j termos de A_k a serem ajustados. Considere as fatorações QR com pivotamento de $(I - U_k U_k^T)Y$ e de $(I - V_k V_k^T)Z$:

$$\begin{aligned} (I - U_k U_k^T) Y P_Y &= Q_Y R_Y, \\ (I - V_k V_k^T) Z P_Z &= Q_Z R_Z, \end{aligned}$$

[5] Seja A uma matriz de seleção. Dizemos que a_{ij} é selecionado se $a_{ij} = 1$. Caso contrário, $a_{ij} = 0$.

com R_Y e R_Z triangulares, P_Y e P_Z de permutação, e Q_Y e Q_Z ortogonais. Utilizando essas fatorações e sabendo que $A_k = U_k \Sigma_k V_k^T$, pode-se verificar que:

$$B = \begin{bmatrix} U_k & Q_Y \end{bmatrix} \left(\begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_k^T Y \\ R_Y P_Y^T \end{bmatrix} \begin{bmatrix} Z^T V_k & P_Z R_Z^T \end{bmatrix} \right) \begin{bmatrix} V_k^T \\ Q_Z^T \end{bmatrix}.$$

Assim, B é escrito como produto de três matrizes: a primeira, de dimensão $m \times (k+j)$ é ortogonal, a segunda é uma matriz diagonal $(k+j) \times (k+j)$ e a última é uma matriz $(k+j) \times n$ com linhas ortonormais. Seja $\hat{B}$ definido por:

$$\hat{B} \doteq \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_k^T Y \\ R_Y P_Y^T \end{bmatrix} \begin{bmatrix} Z^T V_k & P_Z R_Z^T \end{bmatrix}.$$

Se a decomposição SVD de $\hat{B}$ for dada por

$$\hat{B} = \begin{bmatrix} \hat{U}_k & \hat{U}_k^\perp \end{bmatrix} \begin{bmatrix} \hat{\Sigma}_k & 0 \\ 0 & \hat{\Sigma}_j \end{bmatrix} \begin{bmatrix} \hat{V}_k & \hat{V}_k^\perp \end{bmatrix}^T,$$

a melhor aproximação de B de posto k é dada quando $\hat{\Sigma}_j = 0$, i.e.,

$$B_k = U_B \Sigma_B V_B^T = \left(\begin{bmatrix} U_k & Q_Y \end{bmatrix} \hat{U}_k \right) \hat{\Sigma}_k \left(\begin{bmatrix} V_k & Q_Z \end{bmatrix} \hat{V}_k \right)^T.$$

5.8.3 Justificativas do SVD-Updating

Seja $A \in \mathbb{R}^{m \times n}$ a matriz original de termos $\times$ documentos e $D \in \mathbb{R}^{m \times p}$ a matriz cujas colunas correspondem aos p documentos adicionados. Durante a atualização de documentos, vimos que a nova matriz de termos $\times$ documentos é dada por $\begin{bmatrix} A_k & D \end{bmatrix}$ ao invés de $\begin{bmatrix} A & D \end{bmatrix}$. Nesta secção veremos que o uso da aproximação A_k de posto k ao invés de A é perfeitamente viável, não acarretando erros [ZS99]. Primeiro compararemos os valores singulares de $\begin{bmatrix} \mathrm{best_k}(A) & D \end{bmatrix}$ e $\begin{bmatrix} A & D \end{bmatrix}$. Para isso, enunciaremos três lemas úteis.

Lema 5.8.1. *Seja $A \in \mathbb{R}^{m \times n}$ e V ortonormal. Então:*

$$\sigma_i(AV^T) = \sigma_i(A), \qquad i = 1, ..., m.$$

Demonstração. Seja $A = U \Sigma V_1^T$ sua decomposição SVD. Temos que

$$AV^T = U \Sigma V_1^T V^T = U \Sigma (V V_1)^T = U \Sigma V_2^T.$$

Sabemos que V_2 é ortogonal, pois é produto de duas matrizes ortogonais. Pela unicidade do SVD, $U \Sigma V_2^T$ é a única decomposição SVD de AV^T. Desse modo, observa-se que os valores singulares de A e AV^T são iguais. □

Lema 5.8.2. *Seja $A \in \mathbb{R}^{m \times n}$ e $P \in \mathbb{R}^{n \times n}$ uma matriz de permutação. Então, $\sigma_i(A) = \sigma_i(AP)$ para todo $i = 1, ..., n$.*

Demonstração. Seja $A = U\Sigma V^T$ a decomposição SVD de A. Então, $AP = U\Sigma V^T P$. Considere $V_2 \doteq P^T V$. Como a decomposição SVD de uma matriz é única a menos de sinal, temos que $AP = U\Sigma V_2^T$ é o SVD de AP. Logo, $\sigma_i(A) = \sigma_i(AP)$ para todo $i = 1, ..., n$. $\square$

Lema 5.8.3. *Seja* $A = \begin{bmatrix} A^{(1)} & A^{(2)} \end{bmatrix}$*. Então, para* $i = 1, ..., n$*,* $\sigma_i(A^{(1)}) \leq \sigma_i(A)$*.*

Demonstração. Sejam $A^{(1)} \in \mathbb{R}^{m\times n}$ e $A^{(2)} \in \mathbb{R}^{m\times p}$. Vamos supor, por contradição, que existe k tal que $\sigma_k(A^{(1)}) > \sigma_k(A)$. Considere as melhores aproximações de $A^{(1)}$ e de A de posto $k-1$ dadas por $A^{(1)}_{k-1}$ e A_{k-1}. Pelo teorema 4.1.2, temos que

$$\|A^{(1)} - A^{(1)}_{k-1}\|_2 > \|A - A_{k-1}\|_2.$$

Usando a definição de norma de matriz,

$$\sup_{\|x\|_2=1} \|(A^{(1)} - A^{(1)}_{k-1})x\|_2 > \sup_{\|(x,y)^T\|_2=1} \|(A - A_{k-1})(x, y)^T\|_2.$$

No entanto,

$$\sup_{\|(x,y)^T\|_2=1} \|(A - A_{k-1})(x, y)^T\|_2 \geq \sup_{\|x\|_2=1} \|(A^{(1)} - (A_{k-1})^{(1)})x\|_2,$$

ou seja, $(A_{k-1})^{(1)}$ é uma melhor aproximação de A_1 de posto $k-1$ que $A^{(1)}_{k-1}$, o que é uma contradição. Logo, $\sigma_i(A^{(1)}) \leq \sigma_i(A)$ para todo $i = 1, ..., n$. $\square$

Teorema 5.8.1. *Seja* $A \in \mathbb{R}^{m\times n}$ *a matriz original de termos* $\times$ *documentos e* $D \in \mathbb{R}^{m\times p}$ *a matriz cujas colunas correspondem aos* p *documentos adicionados. Temos que:*

$$\sigma_i\big(\begin{bmatrix} \mathrm{best_k}(A) & D \end{bmatrix}\big) \leq \sigma_i\big(\begin{bmatrix} A & D \end{bmatrix}\big), \qquad i = 1, ..., m.$$

Demonstração. Considere o SVD de A como sendo

$$A = \begin{bmatrix} P_k & P_k^\perp \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ 0 & \Sigma_k^\perp \end{bmatrix} \begin{bmatrix} Q_k & Q_k^\perp \end{bmatrix}^T.$$

Para $i = 1, ..., m$, temos:

$$\begin{aligned} \sigma_i\big(\begin{bmatrix} A & D \end{bmatrix}\big) &= \sigma_i\left(\begin{bmatrix} \begin{bmatrix} P_k & P_k^\perp \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ 0 & \Sigma_k^\perp \end{bmatrix} \begin{bmatrix} Q_k & Q_k^\perp \end{bmatrix}^T & D \end{bmatrix}\right) \\ &= \sigma_i\left(\begin{bmatrix} \begin{bmatrix} P_k & P_k^\perp \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ 0 & \Sigma_k^\perp \end{bmatrix} & D \end{bmatrix} \begin{bmatrix} Q_k & Q_k^\perp & I \end{bmatrix}^T\right). \end{aligned} \tag{5.12}$$

Pelo lema 5.8.1, a equação 5.12 é igual a:

$$\sigma_i\left(\begin{bmatrix} \begin{bmatrix} P_k & P_k^\perp \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ 0 & \Sigma_k^\perp \end{bmatrix} & D \end{bmatrix}\right). \tag{5.13}$$

O lema 5.8.2, por sua vez, indica que 5.13 equivale a:

$$\sigma_i\big(\left[\begin{array}{ccc} P_k\Sigma_k & D & P_k^\perp\Sigma_k^\perp \end{array}\right]\big).$$

Novamente pelo lema 5.8.1, temos:

$$\sigma_i\big(\left[\begin{array}{cc} A & D \end{array}\right]\big) = \sigma_i\big(\left[\begin{array}{ccc} P_k\Sigma_k Q_k^T & D & P_k^\perp\Sigma_k^\perp \end{array}\right]\big) = \sigma_i\big(\left[\begin{array}{cc} \left[\begin{array}{cc} \mathrm{best_k}(A) & D \end{array}\right] & P_k^\perp\Sigma_k^\perp \end{array}\right]\big).$$

Já que $\left[\begin{array}{cc} \mathrm{best_k}(A) & D \end{array}\right]$ é submatriz de $\left[\begin{array}{cc} \left[\begin{array}{cc} \mathrm{best_k}(A) & D \end{array}\right] & P_k^\perp\Sigma_k^\perp \end{array}\right]$, completamos a prova usando o lema 5.8.3. □

Para compreender os próximos tópicos, enunciaremos um teorema e dois corolários importantes que envolvem matrizes simétricas positivas semi-definidas:

Definição 5.8.1. *Uma matriz $A \in \mathbb{R}^{m\times m}$ é simétrica positiva semi-definida (SPSD) se A é simétrica e $x^TAx \geq 0$ para todo $x \in \mathbb{R}^m$ não nulo.*

Teorema 5.8.2. *Se $A \in \mathbb{R}^{m\times m}$ é SPSD, então $B \doteq X^TAX$ é SPSD, para $X \in \mathbb{R}^{m\times k}$ de posto completo.*

Demonstração. Como A é SPSD, então, para qualquer $x \in \mathbb{R}^m$ não nulo, temos que $x^TAx \geq 0$. Assim, $x^TXBX^Tx \geq 0 \Leftrightarrow (X^Tx)^TB(X^Tx) \geq 0$. Se $y \doteq X^Tx$ (note que $x \neq 0 \Rightarrow X^Tx \neq 0$) concluímos que $y^TBy \geq 0$ para qualquer $y \in \mathbb{R}^k$ não nulo. □

Corolário 5.8.1. *Se $A \in \mathbb{R}^{m\times m}$ é SPSD, então $B \doteq X^TAX$ é SPSD, para $X \in \mathbb{R}^{m\times k}$ ortogonal.*

Demonstração. Se X é ortogonal, então $\det(X) = \pm 1 \neq 0$. Portanto, toda matriz ortogonal tem posto completo e a afirmação é válida pelo teorema 5.8.2. □

Corolário 5.8.2. *Se $A \in \mathbb{R}^{m\times m}$ é SPSD, então, todas as suas submatrizes principais são SPSD.*

Demonstração. Seja $X \in \mathbb{R}^{m\times k}$ com um elemento igual a 1 em cada coluna e zeros nas demais posições. Podemos escrever uma submatriz principal A_1 de A utilizando-se de tal matriz X, ou seja, $A_1 = X^TAX$. Pelo corolário 5.8.1, temos que A_1 é SPSD. □

Mostraremos agora a relação entre as melhores aproximações de $\left[\begin{array}{cc} \mathrm{best_k}(A) & D \end{array}\right]$ e de $\left[\begin{array}{cc} A & D \end{array}\right]$ de posto k.

Lema 5.8.4. *Considere o SVD de $A \in \mathbb{R}^{m\times n}$ dado por $A = \sum_{i=1}^{\min\{m,n\}} \sigma_i u_i v_i^T$, onde σ_i o i-ésimo valor singular de A e u_i, v_i são os i-ésimos vetores singulares de A à esquerda e à direita, respectivamente. Então, para $p \geq k$, temos:*

$$\text{best}_\text{k}(A) = \text{best}_\text{k}\left(A - \sum_{i=p+1}^{\min\{m,n\}} \sigma_i u_i v_i^T\right).$$

Demonstração. Basta observarmos que

$$A - \sum_{i=p+1}^{\min\{m,n\}} \sigma_i u_i v_i^T = \sum_{i=1}^{p} \sigma_i u_i v_i^T$$

e que $p \geq k$. $\square$

Teorema 5.8.3. *Seja* $\hat{B} = \begin{bmatrix} A & D \end{bmatrix}$, $B = \begin{bmatrix} \text{best}_\text{k}(A) & D \end{bmatrix}$, *onde* $A \in \mathbb{R}^{m\times n}$ *e* $D \in \mathbb{R}^{m\times p}$ *com* $m \geq n + p$. *Vamos assumir que:*

$$\hat{B}^T \hat{B} = X + \sigma^2 I, \qquad \sigma > 0,$$

para uma matriz X *simétrica positiva semi-definida com* $\text{posto}(X) = k$. *Desse modo,*

$$\text{best}_\text{k}(\hat{B}) = \text{best}_\text{k}(B).$$

Demonstração. A idéia é mostrar que as informações retiradas quando A é substituída por $\text{best}_\text{k}(A)$ também serão eliminadas quando $\text{best}_\text{k}(\hat{B})$ é computada de $\hat{B}$. Sabemos que:

$$\hat{B}^T \hat{B} = \begin{bmatrix} A^T \\ D^T \end{bmatrix} \begin{bmatrix} A & D \end{bmatrix} = \begin{bmatrix} A^T A & A^T D \\ D^T A & D^T D \end{bmatrix}.$$

Assim,

$$X = \hat{B}^T \hat{B} - \sigma^2 I = \begin{bmatrix} A^T A - \sigma^2 I & A^T D \\ D^T A & D^T D - \sigma^2 I \end{bmatrix}.$$

Como $\text{posto}(X) = k$, então $\text{posto}(A^T A - \sigma^2 I) \leq k$ e $\text{posto}(D^T D - \sigma^2 I) \leq k$. Considere a decomposição por valores singulares de $A^T A - \sigma^2 I$ e $D^T D - \sigma^2 I$ abaixo:

$$A^T A - \sigma^2 I = V_A \,\text{diag}(\Sigma_A^2, 0) V_A^T, \qquad D^T D - \sigma^2 I = V_D \,\text{diag}(\Sigma_D^2, 0) V_D^T,$$

onde $\Sigma_A \in \mathbb{R}^{k_1 \times k_1}$ e $\Sigma_D \in \mathbb{R}^{k_2 \times k_2}$ com $k_1 \leq k$, $k_2 \leq k$. Podemos escrever a decomposição SVD de A da seguinte maneira:

$$A = U_A \,\text{diag}(\Sigma_A, \sigma I_{t_1}) V_A^T = \begin{bmatrix} U_A^{(1)} & U_A^{(2)} \end{bmatrix} \begin{bmatrix} \Sigma_A & 0 \\ 0 & \sigma I_{t_1} \end{bmatrix} \begin{bmatrix} V_A^{(1)} & V_A^{(2)} \end{bmatrix}^T$$

onde $U_A^{(1)} \in \mathbb{R}^{m \times k_1}$ e $t_1 \doteq n - k_1$. Por sua vez, o SVD de D pode ser dado por:

$$D = U_D \,\text{diag}(\Sigma_D, \sigma I_{t_2}) V_D T = \begin{bmatrix} U_D^{(1)} & U_D^{(2)} \end{bmatrix} \begin{bmatrix} \Sigma_D & 0 \\ 0 & \sigma I_{t_2} \end{bmatrix} \begin{bmatrix} V_D^{(1)} & V_D^{(2)} \end{bmatrix}^T,$$

com $U_D \in \mathbb{R}^{m\times k_2}$ e $t_2 \doteq p - k_2$. Para prosseguir com a demonstração, mostraremos as relações de ortogonalidade entre os vetores singulares à esquerda de A e D. Para isso, considere a matriz $V_A^T A^T D V_D$ particionada na seguinte forma:

$$V_A^T A^T D V_D \doteq \begin{array}{cc} \begin{array}{cc} k_2 & t_2 \end{array} & \\ \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} & \begin{array}{c} k_1 \\ t_1 \end{array} \end{array}$$

Seja $F \doteq \mathrm{diag}(V_A, V_D)$. Temos que:

$$\begin{aligned} F^T X F &= \begin{bmatrix} V_A^T & 0 \\ 0 & V_D^T \end{bmatrix} \begin{bmatrix} V_A \,\mathrm{diag}(\Sigma_A^2, 0) V_A^T & A^T D \\ D^T A & V_D \,\mathrm{diag}(\Sigma_D^2, 0) V_D^T \end{bmatrix} \begin{bmatrix} V_A & 0 \\ 0 & V_D \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{diag}(\Sigma_A^2, 0) & V_A^T A^T D V_D \\ V_D^T D^T A V_A & \mathrm{diag}(\Sigma_D^2, 0) \end{bmatrix} = \begin{bmatrix} \Sigma_A^2 & 0 & S_{11} & S_{12} \\ 0 & 0 & S_{21} & S_{22} \\ S_{11}^T & S_{21}^T & \Sigma_D^2 & 0 \\ S_{12}^T & S_{22}^T & 0 & 0 \end{bmatrix}. \end{aligned}$$

Sabemos que X é simétrica positiva semi-definida (SPSD) e que F é ortogonal. Logo, pelo corolário 5.8.1, $F^T X F$ também é SPSD. Sabemos também, pelo corolário 5.8.2, que as submatrizes principais de uma matriz SPSD são SPSD. Ou seja,

$$\begin{bmatrix} \Sigma_A^2 & S_{12} \\ S_{12}^T & 0 \end{bmatrix}, \begin{bmatrix} 0 & S_{21} \\ S_{21}^T & \Sigma_D^2 \end{bmatrix}, \begin{bmatrix} 0 & S_{22} \\ S_{22}^T & 0 \end{bmatrix}$$

são SPSD. Por definição de SPSD, temos que para qualquer $x \doteq \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$,

$$\begin{aligned} \begin{bmatrix} x_1^T & x_2^T \end{bmatrix} \begin{bmatrix} \Sigma_A^2 & S_{12} \\ S_{12}^T & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq 0 \quad &\Leftrightarrow x_1^T \Sigma_A^2 x_1 + x_2^T S_{12}^T x_1 + x_1^T S_{12} x_2 \geq 0 \\ &\Leftrightarrow x_1^T S_{22}^T x_1 + 2 x_2^T S_{12}^T x_1 \geq 0. \end{aligned}$$

Como $x_1^T \Sigma_A^2 x_1 \geq 0$, temos que $S_{12} = 0$. Analogamente, podemos provar que $S_{21} = 0$ e $S_{22} = 0$. Além disso, $k_1 + k_2 \geq \mathrm{posto}(X) = k$. Com esses resultados e usando o SVD de A e de D, temos:

$$\begin{bmatrix} U_A^{(1)} \Sigma_A & \sigma U_A^{(2)} \end{bmatrix}^T \begin{bmatrix} U_D^{(1)} \Sigma_D & \sigma U_D^{(2)} \end{bmatrix} = \begin{bmatrix} S_{11} & 0 \\ 0 & 0 \end{bmatrix}.$$

Ou seja,

$$\begin{bmatrix} (U_A^{(1)} \Sigma_A)^T U_D^{(1)} \Sigma_D & (U_A^{(1)} \Sigma_A)^T \sigma U_D^{(2)} \\ \sigma (U_A^{(2)})^T U_D^T \Sigma_D & \sigma (U_A^{(2)})^T \sigma U_D^{(2)} \end{bmatrix} = \begin{bmatrix} S_{11} & 0 \\ 0 & 0 \end{bmatrix}.$$

Obtemos, assim, as relações de ortogonalidade [6] abaixo:

$$U_A^{(1)} \perp U_D^{(2)}, \qquad U_A^{(2)} \perp U_D^{(1)}, \qquad U_A^{(2)} \perp U_D^{(2)}.$$

[6] Denotamos $A \perp B$ quando $A^T B = 0$.

Tais relações serão usadas para obtermos a relação entre B e $\hat{B}$. Definimos $\hat{U}$ como a base ortonormal de:

$$\mathrm{span}\Big\{ \begin{bmatrix} U_A^{(1)} & U_D^{(1)} \end{bmatrix} \Big\} \cap \mathrm{span}\Big\{ \begin{bmatrix} U_A^{(2)} & U_D^{(2)} \end{bmatrix} \Big\}^{\perp}$$

onde span{.} denota o espaço gerado pelas colunas de uma matriz (conforme vimos no capítulo 2) e $\mathrm{span}\{.\}^{\perp}$ é o complementar do espaço gerado pelas colunas de uma matriz. Com isso, podemos escrever $\hat{B}$ da seguinte forma:

$$\hat{B} = \begin{bmatrix} A & D \end{bmatrix} = \begin{bmatrix} \hat{U} & U_A^{(2)} & U_D^{(2)} \end{bmatrix} \begin{bmatrix} \tilde{B} & 0 & 0 \\ 0 & \sigma I_{t_1} & 0 \\ 0 & 0 & \sigma I_{t_2} \end{bmatrix} \begin{bmatrix} (V_A^{(1)})^T & 0 \\ 0 & (V_D^{(1)})^T \\ (V_A^{(2)})^T & 0 \\ 0 & (V_D^{(2)})^T \end{bmatrix}.$$

onde $\tilde{B} \in \mathbb{R}^{k \times k}$ e que todos os seus valores singulares são maiores que σ. Portanto,

$$\begin{aligned} \hat{B} &= \begin{bmatrix} \hat{U} & U_D^{(2)} \end{bmatrix} \begin{bmatrix} \tilde{B} & 0 \\ 0 & \sigma I_{t_2} \end{bmatrix} \begin{bmatrix} (V_A^{(1)})^T & 0 \\ 0 & (V_D^{(1)})^T \\ 0 & (V_D^{(2)})^T \end{bmatrix} + \sigma\, U_A^{(2)} \begin{bmatrix} (V_A^{(2)})^T & 0 \end{bmatrix} \\ &= B + \sigma\, U_A^{(2)} \begin{bmatrix} (V_A^{(2)})^T & 0 \end{bmatrix}. \end{aligned}$$

Assim, pelo lema 5.8.4, temos que $\mathrm{best_k}(\hat{B}) = \mathrm{best_k}\, B$. □

Provamos, dessa maneira, que a adição de documentos no *SVD-Updating* é feita adequadamente. A demonstração para atualização de termos é feita de modo análoga. No teorema 5.8.3, tivemos a suposição de que $\hat{B}^T\hat{B}$ possui uma estrutura específica [XK94, XZGK94, ZZ98]. Sabe-se que tal estrutura é adequada no contexto que trabalhamos. Maiores detalhes em relação a isso podem ser obtidos em [ZMS98, ZS99].

Capítulo 6

Resultados Obtidos e Conclusões

Apesar do caráter desse projeto de iniciação científica ser essencialmente teórico, algumas implementações e certos experimentos foram realizados. A maior parte das implementações estão associadas a algoritmos de álgebra linear computacional. Elas serviram para melhorar a compreensão de tais algoritmos, além de permitir observar as dificuldades existentes apenas na prática. A seguir, listamos os principais algoritmos implementados, todos em linguagem Octave [1].

1. Fatoração QR por Reflexões de Householder
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$.
   Saída: Matrizes $R \in \mathbb{R}^{m \times n}$ triangular superior e $Q \in \mathbb{R}^{m \times m}$ ortogonal tais que $A = QR$.

2. Fatoração QR por Rotações de Givens
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$.
   Saída: Matrizes $R \in \mathbb{R}^{m \times n}$ triangular superior e $Q \in \mathbb{R}^{m \times m}$ ortogonal tais que $A = QR$.

3. Fatoração QR com Pivotamento
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$.
   Saída: Matrizes $R \in \mathbb{R}^{m \times n}$ triangular superior, $Q \in \mathbb{R}^{m \times m}$ ortogonal e $P \in \mathbb{R}^{n \times n}$ de permutação, tais que $AP = QR$.

4. Redução à Forma Hessenberg
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times m}$ quadrada.
   Saída: Uma matriz $H \in \mathbb{R}^{m \times m}$ no formato Hessenberg superior tal que $H = Q^T AQ$, onde Q é o produto de refletores de Householder.

[1] Linguagem de computação numérica semelhante ao popular MATLAB, com a diferença de ser Software Livre. Informações detalhadas podem ser obtidas na página `http://www.octave.org`.

5. Bidiagonalização por Golub-Kahan
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$.
   Saída: Matrizes $B \in \mathbb{R}^{n \times n}$ bidiagonal, $U \in \mathbb{R}^{m \times n}$ e $V \in \mathbb{R}^{n \times n}$ ortogonais tais que $B = U^T AV$.

6. Bidiagonalização por Lawson-Hanson-Chan
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$.
   Saída: Matrizes $B \in \mathbb{R}^{n \times n}$ bidiagonal, $U \in \mathbb{R}^{m \times n}$ e $V \in \mathbb{R}^{n \times n}$ ortogonais tais que $B = U^T AV$.

7. Bidiagonalização em Três Fases
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$.
   Saída: Matrizes $B \in \mathbb{R}^{n \times n}$ bidiagonal, $U \in \mathbb{R}^{m \times n}$ e $V \in \mathbb{R}^{n \times n}$ ortogonais tais que $B = U^T AV$.

8. Algoritmo QR
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times m}$ simétrica e um escalar ϵ, múltiplo pequeno da precisão da máquina.
   Saída: Uma matriz $B \in \mathbb{R}^{m \times m}$ diagonal com os elementos convergindo para os auto-valores de A.

9. Golub-Kahan SVD
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$ sem elementos nulos na diagonal e na superdiagonal.
   Saída: Matrizes $B \in \mathbb{R}^{m \times n}$ bidiagonal, $U \in \mathbb{R}^{m \times n}$ e $V \in \mathbb{R}^{n \times n}$ ortogonais tais que $B = U^T AV$.

10. Decomposição por Valores Singulares
   Entrada: Uma matriz $A \in \mathbb{R}^{m \times n}$ com $m \geq n$ e um escalar ϵ, múltiplo pequeno da precisão da máquina.
   Saída: Matrizes $\Sigma \in \mathbb{R}^{n \times n}$ diagonal com valores singulares em ordem decrescente, $U \in \mathbb{R}^{m \times n}$ e $V \in \mathbb{R}^{n \times n}$ ortogonais correspondentes aos vetores singulares à esquerda e à direita de A, respectivamente.

Outras implementações visaram a compreensão dos principais resultados de recuperação de informações com o uso de ferramentas de álgebra linear computacional. Em particular, duas implementações em Octave foram úteis para verificarmos a qualidade e a performance do LSI (*Latent Semantic Indexing*):

1. Aproximação de Posto de Matriz com Fatoração QR

Entrada: Uma matriz de termos $\times$ documentos $A \in \mathbb{R}^{m \times n}$, um vetor de pesquisa $p \in \mathbb{R}^m$ e um inteiro positivo k que representa a aproximação desejada da matriz.
Saída: Vetor de similaridades entre p e as colunas da matriz A original, vetor de similaridades entre p e as colunas da aproximação de A de posto k, e a mudança relativa entre a matriz de posto k e a original.
Observação: Os algoritmos 1, 2 e 3 listados acima puderam ser aproveitados neste experimento.

2. Aproximação de Posto de Matriz com Decomposição SVD
Entrada: Uma matriz de termos $\times$ documentos $A \in \mathbb{R}^{m \times n}$, um vetor de pesquisa $p \in \mathbb{R}^m$ e um inteiro positivo k que representa a aproximação desejada da matriz.
Saída: Vetor de similaridades entre p e as colunas da matriz A original, vetor de similaridades entre p e as colunas da aproximação de A de posto k, e a mudança relativa entre a matriz de posto k e a original.
Observação: O algoritmo 10 listado acima (juntamente com 5, 6, 7 e 9) pôde ser aproveitado neste experimento.

Utilizando esses programas, alguns experimentos foram realizados com uma matriz de termos $\times$ documentos e algumas pesquisas fornecidas pelo artigo de Berry et al [BDJ99]. Outros, semelhantes a esse, foram criados e os resultados de um deles foram apresentados nas secções 5.3, 5.4 e 5.5 desta monografia. Ademais, adquirimos uma coleção disponível no FTP público da Universidade de Cornell [FTP04], denominado ADI, com 3561 termos, 82 documentos e 35 pesquisas para testes.

Os resultados dos testes com a coleção ADI foram comparados com os obtidos pelo sistema SMART, (*System for the Mechanical Analysis and Retrieval of Text*) [SM83]. Dessa forma, verificou-se que o uso de uma aproximação do banco de dados pode melhorar significativamente os resultados. No entanto, alguns resultados obtidos foram díspares em relação ao que o SMART oferecia. Isso se torna evidente se levarmos em consideração as várias técnicas e heurísticas que não foram abordadas no nosso programa.

O SMART - assim como outros sistemas reais de IR - utiliza a maioria das técnicas que vimos: aproximação de posto de matriz, agrupamento de termos, agrupamento de documentos, expansão de pesquisa, folding-in e SVD-Updating. Outras técnicas que não foram vistas aqui podem também ser importantes para que tenhamos um sistema de boa performance. Visto que o objetivo deste trabalho foi a verificação de como a Álgebra Linear Computacional pode ser aplicada à Recuperação de Informações, tanto o estudo teórico, quanto os experimentos práticos foram valiosos e bem aproveitados.

Parte II

Experiência Pessoal

Capítulo 7

O BCC e a Iniciação Científica

7.1 Desafios e Frustrações

A iniciação científica, realizada de Julho de 2003 a Dezembro de 2004, proporcionou uma boa experiência acadêmica. Inicialmente, o trabalho envolvia relembrar conceitos vistos na disciplina *MAC300 - Métodos Numéricos de Álgebra Linear* — e estudar outros tópicos mais avançados, necessários para compreender bem o artigo de Berry et al. [BDJ99], que foi a base de todo o estudo realizado. Essa atividade inicial foi simples e a monitoria que fazia na época — justamente de MAC300 — facilitou ainda mais tal estudo.

Em seguida, realizei algumas implementações de algoritmos de álgebra linear computacional com a linguagem Octave. Isso possibilitou compreender melhor tais algoritmos e observar as dificuldades existentes apenas na prática. Com esses programas, algumas experiências que envolviam recuperação de informações tornaram-se viáveis, bem como comparações com o sistema SMART. A primeira frustração está justamente relacionada a esse sistema: não obtive sucesso em uma simples instalação do SMART na rede Linux e na rede IME. O meu orientador ajudou-me nesse processo, mas por algum motivo, a realização só foi bem sucedida em seu computador. Para facilitar o andamento do trabalho, ele criou uma conta para mim em sua máquina.

O próximo desafio ainda estava relacionado ao SMART, desta vez com a sua utilização. A falta de boas documentações fizeram com que eu consumisse um tempo considerável apenas para descobrir como utilizá-lo. Após várias pesquisas e abusando da intuição, experimentos puderam finalmente ser realizados. Vimos que o uso da aproximação do banco de dados de fato pode melhorar o resultado. Apesar disso, com a coleção que usamos, tal melhora foi pequena. Isso foi um pouco frustrante, mas motivou o estudo de diversas técnicas e heurísticas utilizadas em sistemas reais de Recuperação de Informações.

Durante o ano de 2004, a iniciação basicamente tratava do estudo dessas técnicas. A partir de cada tópico do artigo base, procurou-se outras referências. Era um desafio encontrar certos artigos ou livros por serem raros ou antigos. Esse problema foi parcialmente contornado com a busca por outros textos que possivelmente tratavam do assunto desejado. Entretanto, nem sempre tais buscas tinham retornos imediatos. Como conseqüência, visitas à biblioteca do IME e a diversas bibliotecas digitais, que pouco utilizava até então durante minha graduação, tornaram-se freqüentes.

Em relação aos artigos, notei que nem todos os autores têm a preocupação de escrever claramente seu texto. Muitas vezes, no entanto, isso é ocasionado pela quantidade limitada de páginas a que eles dispõem para publicação. Foi uma pena ver idéias interessantes, mas mal escritas. Um artigo em particular foi bem interessante, por envolver uma demonstração de três páginas usando inúmeros conceitos de álgebra linear computacional, como decomposição SVD, decomposição por auto-valores, matrizes simétricas definidas semi-positivas e ortogonalidade. Infelizmente, vários pontos dessa demonstração foram escritos de maneira pouco clara, o que exigiu grande esforço para compreendê-la.

Alguns lemas e pontos específicos dessa demonstração foram provados por mim e outras tornaram-se possíveis com a ajuda do meu orientador. No entanto, um enunciado em particular consumiu grande parte do nosso tempo. Muitas vezes, chegávamos em resultados quase promissores, faltando apenas um detalhe para o que queríamos. Por fim, depois de algumas semanas, enviei um e-mail para os autores do artigo e um deles gentilmente me deu uma pequena dica. Isto foi o suficiente para eu conseguir completar a prova. Toda essa trajetória obviamente não foi feliz, principalmente quando vimos que o detalhe que faltava era razoavelmente simples. Porém, encarei tudo isso como uma boa experiência.

Outro desafio que relato aqui envolve o próprio assunto que trabalhamos. Em uma certa semana, li um artigo mostrando resultados encorajadores para uma determinada técnica. Na semana imediatamente após, li um outro artigo que critica justamente aqueles resultados, mostrando uma metodologia alternativa, que segundo o autor, é melhor. De fato, sob certos pontos de vista, a abordagem anterior pode ser considerada pior. Observando de outra forma, entretanto, aquela pode ser considerada melhor que esta. A área de Recuperação de Informações lida com diversas incertezas do banco de dados, probabilidades e subjetividade dos usuários do sistema. Todos os modelos e metodologias dependem fortemente do tipo de coleção com que se está trabalhando e do resultado que se quer durante a pesquisa. É uma área rodeada por "dependes", não havendo, portanto, a exatidão que tanto almejamos.

Finalmente, um dos maiores desafios foi, sem dúvida, conciliar as atividades do bacharelado em Ciência da Computação com a iniciação científica. Alguns assuntos poderiam ter sido estudados com maior profundidade e outros experimentos poderiam ter sido feitos. Considero que o aprendizado dado por uma iniciação é grande, e uma outra frustração foi de não tê-la começado um pouco mais cedo. Antes disso, trabalhei durante um ano fora do IME e isso prejudicou meu caminho durante a graduação. Possivelmente, poderia ter adiantado algumas disciplinas nessa época e com isso teria mais tempo para o projeto no último ano. Fiz algumas escolhas erradas durante a minha graduação, mas felizmente a iniciação científica não foi uma delas.

7.2 Disciplinas Mais Relevantes

Em meados do primeiro semestre do terceiro ano, procurei meu orientador para uma possível iniciação científica envolvendo assuntos vistos em

MAC0300 - Métodos Numéricos de Álgebra Linear.

Esta foi a disciplina base do meu projeto. Para compreender bem todos os conceitos vistos nessa matéria, algumas outras foram particularmente úteis:

MAT0111 - Cálculo Diferencial e Integral I
MAT0121 - Cálculo Diferencial e Integral II
MAT0139 - Álgebra Linear Para a Computação

Ao contrário da maior parte dos meus colegas, quando entrei no IME as noções que tinha de programação eram nulas. Por esse motivo, considero as disciplinas básicas de computação essenciais, tanto para o meu projeto (que envolveu programação em Octave e um pouco em C), quanto para a minha formação:

MAC0110 - Introdução à Computação
MAC0122 - Princípios de Desenvolvimento de Algoritmos
MAC0211 - Laboratório de Programação
MAC0323 - Estrutura de Dados

Na iniciação científica em si, conceitos de diversas disciplinas foram extremamente úteis. Os estudos do modelo probabilístico, do modelo booleano, das diversas maneiras de se criar classe de termos semelhantes, do algoritmo do clique para construção de *thesaurus* e as análises de complexidade de tempo e espaço de algoritmos mostraram a importância das seguintes matérias:

MAE0121 - Introdução à Probabilidade e à Estatística I
MAC0328 - Algoritmos em Grafos
MAC0329 - Álgebra Booleana e Aplicações
MAC0338 - Análise de Algoritmos

As disciplinas mencionadas acima foram, de fato, as mais relevantes para a iniciação científica. Já as matérias a seguir, apesar de não estarem relacionadas diretamente ao projeto, foram certamente essenciais para despertar interesse em um futuro mestrado:

MAC0315 - Programação Linear
MAC0427 - Programação Não-Linear (feita como ouvinte)
MAC0453 - Princípios de Pesquisa Operacional e Logística
MAC0325 - Otimização Combinatória
MAC5792 - Otimização Global (feita como aluna especial)

De maneira geral, acredito que as disciplinas obrigatórias do Bacharelado em Ciência da Computação foram importantes para a minha formação. Infelizmente, algumas delas poderiam ter sido dadas de forma mais adequada, contribuindo mais na formação de um cientista. Ter cursado disciplinas optativas em diversas áreas também me permitiu ter uma visão mais ampla dessa ciência, além de poder distinguir bem as áreas de maior e de menor interesse. Além disso, com as disciplinas optativas livres, foi possível conhecer um pouco algumas áreas diferentes. Em particular, a disciplina de História da Fotografia e Arte foi muito interessante, contribuindo não na formação profissional, mas na pessoal.

7.3 Interação com o Supervisor

A procura de uma iniciação científica começou em meados do quinto semestre da minha graduação. O interesse por métodos numéricos de álgebra linear motivou-me a procurar o professor que havia ministrado essa disciplina para a minha turma. Na primeira conversa que tive com ele, era clara sua disposição em me arranjar assuntos relacionados. Um dos temas em particular me pareceu muito interessante e, em Julho de 2003, o professor Paulo José da Silva e Silva tornou-se meu orientador.

Desde o começo da iniciação, tive reuniões semanais com ele em horários e dias fixos. No começo, era ele que me direcionava aos estudos, estabelecendo os tópicos que eu deveria abordar durante a semana. Com o tempo, ele acabou me deixando mais livre para que eu mesma pudesse verificar as atividades. Em relação à escrita de textos, ajudava-me com as correções e dava-me dicas de como escrever adequadamente um texto científico. Muitas vezes, opinava sobre determinados trechos e indicava seus argumentos, mas sempre dando liberdade de fazer o que eu julgava melhor. Acredito que essa relação tenha me proporcionado maior maturidade em relação à pesquisa, o que possivelmente me ajudará no mestrado, no doutorado e na carreira acadêmica que pretendo seguir.

Além de ajudar na organização do projeto e na correção dos textos, discutíamos durante as reuniões o que eu havia estudado. Algumas reuniões eram curtas, devido à facilidade do tópico estudado. Outras, no entanto, duraram algumas horas. Na maioria das vezes que isso ocorreu tínhamos demonstrações de álgebra linear computacional mais sofisticadas e ficávamos um bom tempo pensando nelas. Ele mostrava algumas idéias ou soluções que não apareciam nos artigos e livros, e ensinava-me a pensar e a questionar os resultados.

Durante esses três semestres, ele também me ajudou na escolha de algumas disciplinas optativas. Foi por sua influência que cursei as disciplinas de Pesquisa Operacional, Análise

Real (do MAT) e Otimização Global (da pós-graduação). Mesmo antes de decidir minha área de mestrado, ele me incentivava a assistir seminários que ajudariam a conhecer um pouco as diversas áreas existentes. Em particular, seu incentivo em assistir seminários de Otimização Contínua estimulou minha escolha por tal área.

O Paulo foi um bom orientador. Sua dedicação pôde ser observada inclusive quando foi assistir algumas apresentações do trabalho de formatura (incluindo a minha). Somando-se a isso, sua área de estudo me pareceu bem interessante e por isso continuarei sob sua orientação no mestrado a partir do ano que vem.

7.4 Os Próximos Passos

Durante a iniciação científica, alguns tópicos de Recuperação de Informações associados ao modelo vetorial foram selecionados para estudo. Inúmeras outras metodologias e heurísticas não foram vistas e algumas ainda estão sendo elaboradas pelos pesquisadores da área. Alguns artigos mais recentes, i.e., dadas a partir do final da década de 90, me pareceram interessantes para estudo. Alguns desses assuntos são listados abaixo:

- Métodos probabilísticos de expansão de pesquisa [Din99].
- Técnicas que lidam com a esparsidade da matriz [Ber92].
- Decomposição SVD para matrizes esparsas: Arnoldi [LS96], Lancsos [Lan50], iteração de subspaços [Rut70], minimização de traço [Ber92].
- Problema dos quadrados mínimos em Recuperação de Informações [JB00].
- Filtro de informações usando SVD riemanniana (R-SVD) [JB98].

Meu interesse pela Recuperação de Informações não se anula com o término da iniciação científica. Apesar de querer estudar outras coisas no mestrado, gostaria de ver alguns desses tópicos ao longo do tempo e pretendo acompanhar, mesmo que não tão profundamente, as novidades e tecnologias envolvendo essa área, em particular o *Latent Semantic Indexing*. É interessante ver, por exemplo, que essa metodologia está sendo abordada inclusive em filtros de *spams* [Ker04].

7.5 Considerações Finais

Até o segundo ano do bacharelado sempre havia dúvidas se eu tinha escolhido o curso certo. O pensamento de mudar para o curso de Matemática foi eliminado quando vi que a Ciência da Computação também engloba áreas mais teóricas. Sem dúvida, o curso estimulou muito o raciocínio, não apenas com as disciplinas de matemática que tanto gostava, mas também com as disciplinas de programação e de sistemas.

A iniciação científica me permitiu entrar em contato com o meio acadêmico, influenciando-me em certas escolhas profissionais. Procurar e ler diversos artigos, escrever textos científicos e discutir idéias novas são apenas algumas dentre tantas coisas que se aprende durante uma iniciação. Apesar do meu tema de mestrado não envolver diretamente os assuntos abordados no projeto, a relação com o orientador despertou meu interesse pela área de Otimização.

Por fim, agradeço aos meus pais, amigos, colegas e professores do IME, por todo o apoio que me deram durante esses quatro anos de graduação.

Referências Bibliográficas

[AF77] R. Attar and A.S. Frankel. Local feedback in full-text retrieval systems. *Journal of the ACM*, 24(3):397–417, 1977.

[AGB00] J. Aitchison, A. Gilchrist, and D. Bawden. *Thesaurus Construction and Use - A Practical Manual*. Aslib, London, 4th. edition, 2000.

[AP75] H. Andrews and C. Patterson. Outer product expansions and their uses in digital image processing. *American Math.*, 82:1–13, 1975.

[BC89] N. Belkin and W. Croft. Retrieval techniques. *Annual Review od Information Science and Technology*, pages 109–145, 1989.

[BDJ99] M.W. Berry, Z. Drmac, and E.R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, 1999.

[BDO95] M.W. Berry, S.T. Dumais, and G.W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[BDS95] M.W. Berry, S.T. Dumais, and A.T. Shippy. A case study of latent semantic indexing. Technical Report UT-CS-95-271, 1995.

[Ber92] M. Berry. Large scale singular value computations. *Internat. J. Supercomputer Applications*, (6):13–49, 1992.

[Ber01] M. Berry. *Computational Information Retrieval*. SIAM Proceedings in Applied Mathematics, Philadelphia, 1st. edition, 2001.

[BF96] M.W. Berry and R.D. Fierro. Low-rank orthogonal decompositions for information retrieval applications. *Numerical linear algebra with applications*, 3(4):301–327, 1996.

[BSAS95] C. Buckley, G. Salton, J. Allan, and A. Singhanl. Automatic query expansion using SMART: TREC-3. pages 69–80, Gaithersburg, MD, April 1995. Overview of the Third Text REtrieval Conference.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1st. edition, 1999.

[Coh95] J. Cohen. Highlights: language and domain independent automatic indexing terms for abstracting. *Journal of the American Society for Information Science*, 46(3):162–174, 1995.

[Cro77] W.B. Croft. Clustering large files of documents using the single link method. *Journal of the ASIS*, 28(6):341–344, 1977.

[Cro83] W.B. Croft. Experiments with representation in a document retrieval system. *Information Technology: Research and Development*, 2(1):1–21, 1983.

[Cro90] C.J. Crouch. An approach to the automatic construction of global thesauri. *Information Processing & Management*, 26(5):629–640, 1990.

[CY92] C.J. Crouch and B. Yong. Experiments in automatic statistical thesaurus construction. pages 77–87, Denmark, 1992. SIGIR 92, 15th Int. ACM ISIGIR Conf. on R&D in Information Retrieval.

[Dam95] M. Damashek. Gauging similarity with n-grams: language independent categorization of text. *Science*, 267:843–848, 1995.

[DDF+90] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by lantent semantic analysis. *Journal of American Society of Information Science*, 41:391–407, 1990.

[Dem97] J.W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[Din99] C.H.Q. Ding. A similarity-based probability model for latent semantic indexing. pages 59–65. Proc. of 22nd ACM SIGIR'99 Conference, August 1999.

[Dum91] S. Dumais. Improving de retrieval of information from external sources. *Behavior Research Methods, Instruments & Computers*, 23:229–236, 1991.

[Eat02] J.W. Eaton. *GNU Octave Manual*. Network Theory Ltd., 2002.

[ERW92] F.C. Ekmekcioglu, A.M. Robertson, and P. Willett. Effectiveness of query expansion in rankedoutput document retrieval systems. *Journal of Information Science*, 18(2):139–147, 1992.

[For97] D.J. Forkett. Thesaurus. In K. Sparck Jones and P. Willet, editors, *Readings in Information Retrieval*, pages 111–134. Morgan Kaufmann Publishers, Inc., 1997.

[FTP04] Cornell University Public FTP. - SMART. ftp://ftp.cs.cornell.edu/pub/smart, December 2004.

[GB65] G.H. Golub and P.A. Businger. Linear least squares solutions by householder transformations. *Numerical Math.*, 7:269–276, 1965.

[GK65] G.H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis*, 2:205–224, 1965.

[GL96] G.H. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd. edition, 1996.

[Goo04] Google. http://www.google.com, December 2004.

[Gre92] G. Grefenstette. Use of syntactic context to produce term association lists for retrieval. pages 89–97, Denmark, 1992. SIGIR 92, 15th Int. ACM ISIGIR Conf. on R&D in Information Retrieval.

[HB92] M. Hancock-Beaulieu. Query expansion: advances in research in on-line catalogues. *Journal of Information Science*, 18(2):99–103, 1992.

[HSD01] P. Husbands, H. Simon, and C.H.Q. Ding. On the use of the singular value decomposition for text retrieval. *SIAM Reviews*, pages 145–156, 2001.

[HV96] D. Harman and E. Voorhees. Overview of the fifth text retrieve conference TREC-5. pages 1–28, Gaithersburg, MD, November 1996. The Fifth Text REtrieval Conference (TREC-5).

[Ide71] E. Ide. New experiments in relevance feedback. In G. Salton, editor, *The SMART Retrieval System*, pages 337–354. Prentice Hall, 1971.

[JB98] E.P. Jiang and M.W. Berry. Information filtering using the riemannian svd. pages 386–395, Spring-Verlag, London, August 1998. Proceedings of the Fifth International Symposium on: Solving Irregularly Structured Problems in Parallel.

[JB00] E.P. Jiang and M.W. Berry. Solving total least squares problems in information retrieval. *Linear Algebra and Its Application*, 316:136–157, 2000.

[Ker04] F.J. Kermadec. The fight against spam, part 2, May 2004.

[KM00] G. Kowalski and M.T. Maybury. *Information Storage and Retrieval Systems, Theory and Implementation*. Kluwer Academic Publishers, 2nd. edition, 2000.

[Lan50] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards*, 45:255–282, 1950.

[LS96] R. Lehoucq and D. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM J. Matrix Anal. Appl.*, 17:789–821, 1996.

[Mat95] Inc. Mathworks. *MATLAB, High-performance Numeric Computation and Visualization Software: Reference Guide*, 1995.

[Mir60] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math.*, 11:50–59, 1960.

[MW68] R.S. Martin and J.H. Wilkinson. Similarity reduction of a general matrix to hessenberg form. *Numer. Math.*, 12:349–368, 1968.

[MWZ72] J. Minker, G.A. Wilson, and B.H. Zimmerman. An evaluation of query expansion by the addition of clustered terms for a document retrieval system. *Information Storage and Retrieval*, 8(6):329–348, 1972.

[O'B94] G. O'Brien. Information management tools for updating an SVD-encoded indexing scheme. Master's thesis, University of Tennessee, Knoxville, TN, 1994.

[Ove01] M.L. Overton. *Numerical computing with IEEE floating point arithmetic.* Society for Industrial and Applied Mathematics, Philadelphia, 2001.

[PP73] B.N. Parlett and W.G. Poole. A geometric theory for the QR, LU and power iterations. *SIAM Jornal Num. Anal.*, 10:389–412, 1973.

[QF93] Y. Qiu and H.P. Frei. Concept based query expansion. pages 160–169. Proceedings of ACM SIGIR International Conference on Research and Development in Information Retrieval, 1993.

[QF95] Y. Qiu and H.P. Frei. Applying a similarity thesaurus to a large collection for information retrieval. Technical report, Department of Computer Science, Zurich, Switzerland, 1995.

[Qiu92] Y. Qiu. ISIR: An integrated system for information retrieval. pages 55–66, Spring-Verlag, London, 1992. Proceedings of the 14th Information Retrieval Colloquium.

[Roc71] J.J. Rochio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System - experiments in automatic document processing.* Prentice Hall, 1971.

[Rut70] H. Rutishauser. Simultaneous iteration method for symmetric matrices. *Numer. Math.*, 16:205–223, 1970.

[Sal72] G. Salton. Experiments in automatic thesaurus construction for information retrieval. *Information Processing*, 71:115–123, 1972.

[Sal75] G. Salton. *Dynamic information and library processing.* Prentice Hall, Englewod, New Jersey, 1975.

[Sal80] G. Salton. Automatic term class construction using relevance - a summary of work in automatic pseudoclassification. *Information Processing & Management*, 16(1):1–15, 1980.

[SB88] G. Salton and C. Buckley. Term-weighting approaches in automatic retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[SJ71] K. Sparck-Jones. *Automatic keyword classifications for information retrieval.* Buttersworths, London, 1971.

[SJ72] K. Sparck-Jones. A statistical interpretation of term specificity and its applications in retrieval. *J. Documentation*, 28:11–21, 1972.

[SJ91] K. Sparck-Jones. Notes and references on early classification work. *SZGZR Forum*, 25(1):10–17, 1991.

[SJB71] K. Sparck-Jones and E.B. Barber. What makes an automatic keyword classification effective? *Journal of the ASIS*, 18:166–175, 1971.

[SM83] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[SvCR83] A.F. Smeaton and van C.J. Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal*, 26(3):239–246, 1983.

[TB97] L.N. Trefethen and D. Bau. *Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, 1997.

[TRE04] TREC. http://trec.nist.gov, December 2004.

[vR79] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, 1979.

[Wat91] D.S. Watkins. *Fundamentals of Matrix Computations.* John Wiley & Sons, 1991.

[Wil68] J.H. Wilkinson. Global convergence of tridiagonal QR algorithm with origin shifts. *Linear Algebra and Its Applic.*, I:409–420, 1968.

[XK94] G. Xu and T. Kailath. Fast subspace decomposition. *IEEE Transactions on Signal Processing*, 42:539–551, 1994.

[XZGK94] G. Xu, H. Zha, G. Golub, and T. Kailath. Fast algorithms for updating signal subspaces. *IEEE Transactions on Circuits and Systems*, 41:537–549, 1994.

[ZMS98] H. Zha, O. Marques, and H. Simon. A subspace-based model for information retrieval with applications in latent semantic indexing. *Proc. of Irregular '98, Lecture Notes in Computer Science*, pages 29–42, 1998.

[ZS99] H. Zha and H.D. Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791, 1999.

[ZZ98] H. Zha and Z. Zhang. On matrices with low-rank-plus-shift structures: Partial SVD and latent semantic indexing. Technical report, Department of Computer Science and Engineering, Pennsylvania State University, 1998.