UMA FERRAMENTA DE MINERAÇÃO DE TEXTO EM BANCOS DE DADOS DE UM HOSPITAL UNIVERSITÁRIO UTILIZANDO DECOMPOSIÇÕES MATRICIAIS

Thiago Madureira Braga

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Amit Bhaya

Rio de Janeiro
Agosto de 2011

UMA FERRAMENTA DE MINERAÇÃO DE TEXTO EM BANCOS DE DADOS DE UM HOSPITAL UNIVERSITÁRIO UTILIZANDO DECOMPOSIÇÕES MATRICIAIS

Thiago Madureira Braga

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

_____

Prof. Amit Bhaya, Ph.D.

_____

Prof. Eugenius Kaszkurewicz, D.Sc.

_____

Prof. Afrânio Lineu Kritski, M.D., Ph.D.,

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2011

Braga, Thiago Madureira

Uma Ferramenta de Mineração de Texto em Bancos de Dados de um Hospital Universitário Utilizando Decomposições Matriciais/Thiago Madureira Braga. – Rio de Janeiro: UFRJ/COPPE, 2011.

XV, 87 p.: il.; 29, 7cm.

Orientador: Amit Bhaya

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2011.

Referências Bibliográficas: p. 86 – 87.

1. mineração. 2. automática. 3. texto. 4. banco de dados. 5. médico. 6. decomposição. 7. matricial. I. Bhaya, Amit. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

"Once you have flown, you will walk the earth with your eyes turned skywards, for there you have been, and there you long to return."

Leonardo da Vinci

Agradecimentos

Em primeiro lugar agradeço a Deus pai e a Nossa Senhora, mãe primeira. Pela pela vida com saúde, pela sabedoria e inteligência essenciais ao alcance desta graça. Pela proteção, pela atenção com as vontades e ansiedades compartilhadas, pelas queixas proferidas, pelas falhas e por todas as pessoas boas encaminhadas.

Agradeço intensamente a minha família, célula principal da minha origem. Ao meu pai, em especial, pelo amor e pela força serena nas cartas ainda manuscritas que chegavam de surpresa e pela paciência na escuta de assuntos diversos algumas vezes até intangíveis. Agradeço a minha mãe, pelo amor, carinho, zelo e atenção incondicionais dispensados, meu exemplo primeiro na engenharia. Aos meus irmãos Vítor e Cláudio, pelo companheirismo e fidelidade inabaláveis e também por serem as testemunhas mais presentes na minha história e, sem dúvida nenhuma o mais forte vínculo com meu passado.

Agradeço fraternalmente ao meus avós Alípia, Maria, Olívio e Albides, pois tenho certeza que os passos de hoje se tornaram possíveis pelas idéias e estruturas construídas antecipadamente. Creio que esta etapa é motivo de alegria e orgulho para todos, onde quer que estejam. Agradeço de coração a Nathália e Laura pelo carinho, amizade, estímulo e presença ativa. Agradeço aos meus tios e tias que participaram desta conquista no suporte e torcida, em especial a tia Gilza.

Em especial, agradeço aos meus colegas de graduação mais próximos e hoje também exemplos para mim de engenheiros e profissionais capazes. Agradeço ao Leandro e Nataly pela amizade, apoio e suporte no Rio e principalmente em Niterói. Agradeço ao Saulo pela amizade sincera, pela parceria, pelas conversas construtivas e por ajudar a manter o foco nas coisas importantes. Agradeço ao Fiche e ao Diego pelos conselhos e pelo estímulo ao empreendedorismo além da amizade.

Agradeço também aos meus amigos no mestrado. Em especial ao Andrei, pela amizade, pelas inúmeras conversas conflitantes, mas muito engrandecedoras, por me ensinar a metodologia de fabricação de uma cerveja pilsen, por mostrar como reconhecer uma boa vodka, por me apresentar o cheiro agora inconfundível do Jack Daniels e também pelos momentos "desesperadores" em que estudamos juntos. Agradeço ao Lécio e a Mery, pela amizade e companheirismo que extrapolaram as salas de aula. Agradeço a Priscilla pela amizade e entrosamento no novo ambiente.

Agradeço ao Michael e a Lúcia por tudo que representaram na minha vida pessoal e profissional nos últimos anos. Pelas várias conversas e conselhos a respeito dos caminhos a serem tomados, pela amplitude de visão conquistada no convívio diário. Pelas oportunidades de trabalho e de morar no Rio de Janeiro, que culminaram no início deste mestrado.

Agradeço de coração ao Hermes e a Maria José pelo carinho e apoio fraternais quando em visita a Belo Horizonte e também pelo prestígio de os terem participando da minha caminhada de forma tão ativa. Ao Felipe e a Carla pela amizade mineira conquistada em solo carioca, pelos momentos de descontração e estímulo.

Agradeço fortemente ao Amit, além de orientador e conselheiro, um mestre no sentido mais natural da palavra, com sua arte de ensinar provocando as idéias de forma sutil e tranquila numa busca constante por conhecimento aprimorado. Agradeço ao Leonardo Torres pela boa semente plantada que tento irrigar diariamente. Agradeço imensamente ao NACAD na amplitude de toda sua equipe que desde o início foi o laboratório desejado. Agradeço a Myrian pelos conselhos, ao Padilha, mestre da arte da convivência, ao Orlando pelos posicionamentos "sentimentais" em relação a tudo, nas horas avançadas da noite no laboratório, e ao Jonas pelo conhecimento compartilhado. A dona Sandra, por manter todo o laboratório limpo e a Mara por administrar este ambiente.

Agradeço aos colaboradores do Hospital Universitário Clementino Fraga Filho. Nominalmente ao professor Paulo Bahia e a professora Elise Tonomura que incentivaram a idéia e permitiram acesso aos dados para mineração. Ao pessoal do CPD especialmente Ana Rangel, Ana Braga, Gislane e Gisele, pela atenção e parceria.

Agradeço ao meu time da Cybersolda, acho que vai ser sempre asssim. Ao Ivan pelo primeiro suporte no Rio, ao Leonardo, Rodrigo e Rodolfo pelo companheirismo e pelas boas conversas em momentos difícieis. Ao Paulo pelas piadas peculiares e a todos os outros componentes desta equipe.

Agradeço a Thau, empresa fundada por várias idéias e outros tantos sonhos, que aos poucos vem tomando forma real. Ao Fabrício e Pedro, minha primeira equipe de trabalho neste projeto, dos quais tenho amizade e orgulho. Ao pessoal da contabilidade pela agilidade e colaboração no tratamento burocrático.

Agradeço a banca que me avalia e certamente colabora em engrandecer meu aprendizado e conhecimento com sugestões e apontamentos. Agradeço a Daniele na Secretaria da Elétrica pela paciência e suporte nas matrículas. Finalmente, agradeço ao Rio de Janeiro, na sua completude, por ter me proporcionado tantas experiências engrandecedoras e as vezes turbulentas para a vida. Do preconceito e receio no primeiro momento transformados na crença de que a maioria de pessoas boas superam em muito qualquer outro contratempo.

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

UMA FERRAMENTA DE MINERAÇÃO DE TEXTO EM BANCOS DE DADOS DE UM HOSPITAL UNIVERSITÁRIO UTILIZANDO DECOMPOSIÇÕES MATRICIAIS

Thiago Madureira Braga

Agosto/2011

Orientador: Amit Bhaya

Programa: Engenharia Elétrica

Apresenta-se, nesta dissertação, uma ferramenta para mineração automática de texto em bancos de dados médicos de grande porte utilizando decomposições matriciais. Técnicas para extração, transformação e carga de documentos em formato padronizado, retirada de palavras de pouco valor semântico (*stop-words*) e extração de raízes (*stemming*) são abordadas. A montagem de matrizes termo-documento a partir da vetorização de coleções de textos e sua posterior decomposição em valores singulares, possibilitando aproximação de dimensão reduzida com menor erro possível são implementadas e aferidas. Como resultado, a recuperação da informação e análise de relevância é abordada sob a perspectiva de comparação de ângulos entre vetores representando documentos. Questões relativas à implementação em software, bem como algoritmos disponíveis e utilização de hardware são consideradas simultâneamente com intuito de otimizar o desempenho do algoritmo, que é o motor da máquina de busca proposta, bem como viabilizar a utilização do mesmo em bancos de dados de grande porte que surgem em aplicações reais.

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

A TOOL FOR TEXT MINING IN A UNIVERSITY HOSPITAL DATABASE USING MATRIX DECOMPOSITIONS

Thiago Madureira Braga

August/2011

Advisor: Amit Bhaya

Department: Electrical Engineering

This dissertation describes the theoretical and practical details of a tool for text mining in large medical databases using matrix decompositions. Techniques to extract, transform and load documents in standardized format are presented. Words of little semantic value (stop-words) are removed from documents and extraction of roots of words (stemming) are addressed using algorithms. The assembly of a term-by-document matrix from vectorized collection of texts and its subsequent singular value decomposition in order to obtain a low dimensional approximation achieving the lowest possible error are implemented and verified. Information retrieval and analysis of relevance is thus addressed from the perspective of comparison of angles between vectors that represent documents in the collection. Issues related to software implementation, as well as use of available algorithms and hardware are considered in order to optimize the algorithm that lies at the heart of the proposed search engine.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ACID	*Atomicity Consistency Isolation Durability*, p. 46
ARPACK	*ARnoldi Package*, p. 36
ASCII	*American Standard Code for Information Interchange*, p. 12
CCS	*Compressed Column Storage*, p. 32
CRS	*Compressed Row Storage*, p. 32
ETL	*Extract Transform and Load*, p. 12
FLOPS	*Floating Point Operations Per Second*, p. 41
HIS	*Hospital Information System*, p. 2
HITS	*Hyperlink-Induced Topic Search*, p. 9
HUCFF	Hospital Universitário Clementino Fraga Filho, p. 45
JDBC	*Java Database Connectivity*, p. 49
JRE	*Java Runtime Environment*, p. 37
LSI	*Latent Semantic Indexing*, p. 11
NLQ	*Natural Language Query*, p. 19
PCA	*Principal Component Analysis*, p. 29
RMI	*Remote Procedure Call*, p. 37
SGBD	Sistema de Gerência de Banco de Dados, p. 45
SQL	*Structured Query Language*, p. 7
SSD	*Solid State Drive*, p. 42
SVD	*Singular Value Decomposition*, p. 26
UTF-8	*8-bit Unicode Transformation Format*, p. 12

Capítulo 1

Introdução

A sociedade moderna e digitalizada produz continuamente cada vez mais dados. O avanço da eletrônica, já em escala nanométrica, viabilizou sensores diversos, a transmissão e também o armazenamento dos dados numa escala antes impensada.

1.1 Motivação

Os antigos e raros manuscritos, a carta convencional, jornais, artigos, livros, e-mails, mensagens e conversas através do celular ou Internet, sites, blogs, tweets, músicas on-line e os vídeos em alta definição exemplificam o vigor dessa onda definida por Wurman em [19] como "tsunami de dados", nas seguintes palavras:

"This is a tidal wave of unrelated, growing data formed in bits and bytes, coming in an unorganized, uncontrolled, incoherent cacophony of foam. It's filled with flotsam and jetsam. It's filled with the sticks and bones and shells of inanimate and animate life. None of it is easily related, none of it comes with any organizational methodology."

Originados de fontes naturalmente diferentes, distribuídas e aleatórias, esses dados apresentam baixa organização e correlação além de estrutura heterogênea na maioria dos casos. Garimpar sistematicamente este conteúdo dinâmico em busca de novas informações úteis e aparentemente implícitas parece imprecindível e, ao mesmo tempo, humanamente impossível através de metodologias convencionais devido ao abundante volume.

1.1.1 Mineração Automática de Texto

Neste cenário, a mineração automática de dados aparece como alternativa visando equiparar as velocidades de processamento de informação e criação do conteúdo. Para compreender melhor e simplificar o desafio é notável que boa parte dos dados no cotidiano estão sob forma de textos, espalhados nos mais diversos meios

de comunicação e armazenamento existentes. Textos, por sua vez, se caracterizam por aspectos vantajosos de compacidade, densidade de informação e padronização linguística, natural de cada idioma, sendo facilmente manipulado em computadores.

Considerar os computadores no tratamento de textos é desejável por vários motivos. Mesmo máquinas modestas mostram bastante poder de processar texto em formatos e codificações diversos. O armazenamento digital, seja ele magnético ou óptico, alcança hoje patamares respeitáveis. A transmissão de grandes quantidades de dados se torna a cada dia mais flúida e confiável. Não bastasse isso, a simplicidade de programação para execução de tarefas repetitivas eleva a escala da solução para um nível compatível ao desafio mencionado.

Por definição, minerar é extrair de mina, geralmente um recurso de valor. No ambiente de sistemas de informação, essa atividade tem os objetivos de realizar recuperação de informação, reconhecimento de padrões, filtragem de informação, agrupamentos, recomendação etc. Na maioria dos casos é realizada sobre grande massa de dados digital, que pode estar contida em um ambiente completamente mapeado ou irrestrita e não estruturada como no caso da Internet.

Atualmente, técnicas e métodos variados de mineração são estudados e utilizados em diversas áreas. Algoritmos com esse propósito constituem o cerne de praticamente todas a máquinas de busca na Internet disponíveis hoje, além de programas de recomendação e softwares de aprendizado estatístico. Embarcado nesta conjuntura, este trabalho visa estudar, desenvolver e avaliar técnicas de mineração automática de textos na área da saúde, adaptando e contextualizando ferramentas existentes, assim como inovando quando necessário.

1.1.2 Bancos de Dados Médicos de Grande Porte

A produção de dados em larga escala e velocidade é realidade tanto nos grandes centros de saúde, representados por hospitais e clínicas, quanto em consultórios médicos isolados. Estabelecimentos majoritariamente digitalizadas geram todos os dias muitos bytes em forma de cadastros de pacientes, medicamentos, anamneses, procedimentos clínicos, diagnósticos, laudos, exames e resultados. A dinâmica da informação é proporcional ao grau de modernização da instituição.

Por características técnicas, os dados no ambiente médico ocorrem de forma mais concentrada e organizada. A inserção e edição das informações é geralmente feita através do sistema de gestão da informação utilizado. Em hospitais esses sistemas são conhecidos como HIS[1] e contam geralmente com servidor central e acesso remoto via rede. Em clínicas menores e consultórios é comum encontrar um sistema análogo, porém local, para gerir, garantir persistência[2] e segurança da informação.

[1] *Hospital Information System*, em português Sistema de Informação Hospitalar

[2] Armazenamento permanente em um dispositivo físico, como um disco rígido.

Em [16] Rubin defende o projeto de sistemas de informação médica orientados ao paciente. Neste conceito, a informação clínica dos pacientes é agregada e expandida em registros vinculados ao longo de toda sua vida, o que facilita o acesso e consequentemente a tomada de decisão. O efeito colateral dessa abordagem é a tendência de crescimento vertiginoso da quantidade de informação armazenada dado que nunca são descartadas.

Em contexto mais científico parte significativa dos dados também estão sob forma de texto com características especiais condicionadas ao vocabulário médico visando objetividade e clareza. Somado a isso, o formato digital dos bancos de dados relacionais, bastante populares, concentra e submete toda informação à uma lógica matemática que facilita e agiliza qualquer acesso.

1.1.3 Decomposições Matriciais

A matemática tem se mostrado o caminho mais objetivo, simples e eficiente para realizar mineração automática de informações em bancos de dados de grande porte. As bases teóricas são bem conhecidas, robustas e eficazes, podendo escalar largamente quando isso se fizer necessário. Somado a isso, a conversão de dados textuais em objetos matemáticos segue metodologia bem sedimentada e de rápida execução. Sobre estes objetos gerados, operações algébricas permitem estabelecer quantitativos objetivos e embutir lógica, aprimorando o processo. Aliado a matemática o poder computacional disponível é bastante útil no tratamento numérico sendo ainda relativamente simples de ser implementado.

Em uma estrutura projetada, toda informação é vetorizada. Através de processos bem definidos e específicos para cada idioma, os documentos da coleção de interesse são mapeados em vetores. O conjunto dos vetores, ponderados e organizados em forma de matriz passam a constituir uma nova base matemática gerada a partir da coleção. Em uma transformação bem projetada, a matriz termo-documento, como é conhecida, retém, semanticamente, características relevantes do conjunto original com dimensões também proporcionais e pode ser utilizada como entrada para algoritmos matemáticos de mineração.

Partindo da disponibilidade de objetos matemáticos (matrizes e vetores) coerentes, as decomposições matriciais fornecem tratamento matemático especial visando salientar ou descobrir informações presentes, ainda que implícitamente, de forma ponderada e ordenada. Neste aspecto, recuperar informações antes não estruturadas remete a comparações vetoriais agora simples. Outras funcionalidades relacionadas como filtros de conteúdo, agrupamentos e reconhecimento de padrões, por exemplo, são mero desdobramento dessa lógica.

1.2 Objetivo Deste Trabalho

Este trabalho tem como objetivos estudar, desenvolver, implementar e testar um conjunto estruturado de práticas para a mineração automática de texto em bancos de dados médicos de grande porte utilizando decomposições matriciais, oferecendo *insights* para futuro desenvolvimento de softwares inteligentes na área da saúde.

No estudo são pesquisados métodos e alternativas existentes para minerar automaticamente textos levando em consideração as particularidades relacionadas ao formato, a codificação e principalmente ao idioma. Todo aspecto envolvendo a ética médica e a utilização de informações sigilosas são resguardados no acesso anônimo a grandes coleções de documentos médicos. Estratégias para estruturação matricial e sua posterior decomposição são comparadas na forma de algortimos matemáticos já implementados e testados.

O desenvolvimento e projeto foca em funcionalidades úteis ao meio como recuperação eficiente de informações não estruturadas no banco de dados, agrupamento de elementos semelhantes, análise estatística de textos (laudos) e reconhecimento de padrões. São cogitados futuros desenvolvimentos em pesquisa acadêmica para evolução de patologias na população e a criação de uma base de conhecimento a partir de tratamento de laudos.

Na implementação buscou-se uma integração das ferramentas disponíveis de forma eficiente e interessante para compatibilizar as melhores soluções em cada área. São avaliadas as plataformas de execução, as linguagens de codificação, algoritmos já estabelecidos, os requisitos de hardware e tempo necessários, mantendo atenção especial para a viabilidade e utilidade da solução proposta.

Nos testes, o produto da implementação é submetido à uma base conceitual capaz de estabelecer métricas de desempenho realistas para trabalho realizado. Em momento posterior um banco de dados geral oriundo do hospital universitário é utililizado, para avaliação de desempenho e carga em situações usuais oferecendo ao corpo clínico uma máquina de busca simples com interface web.

O esquema mostrado em seguida ilustra as etapas principais no processo de pesquisa às informações mineradas. Outros profissinais devidamente autorizados são considerados como usuários potenciais além do próprio médico. Visando **realizar uma consulta** e **acessar a informação** previamente indexada, é necessário **proceder à uma autenticação** que credencia a efetuar buscas de forma prática, sem necessariamente, conhecer a estrutura organizacional interna dos dados.

$$\left.\begin{array}{r}\text{médico}\\ \text{enfermeira}\\ \text{secretária}\end{array}\right\} \rightarrow \begin{array}{c}\text{procede}\\ \text{autenticação}\end{array} \rightarrow \begin{array}{c}\text{realiza}\\ \text{consulta}\end{array} \rightarrow \begin{array}{c}\text{acessa}\\ \text{informação}\end{array}$$

1.3 Pesquisa Bibliográfica Específica

O baixo custo e alta disponibilidade da informática alcançados com o avanço da eletrônica tornaram os sistemas de gestão da informação populares em centros de saúde. As soluções convencionais e mais populares seguem o modelo dos antigos arquivos físicos, mantidos ainda hoje em alguns casos, migrando para o meio digital as mesmas funcionalidades inicialmente projetadas para o papel:

- cadastro de pacientes
- alocação de recursos humanos e físicos
- registro de consultas, procedimentos e internações
- armazenamento de anamneses, exames e diagnósticos
- controle de medicamentos e estoque

Neste contexto, a maioria destes sistemas não tira proveito extra do potencial implícito dos dados. O armazenamento digital serve apenas ao propósito de agilizar o acesso e melhorar segurança dos dados. Avanços relacionados a novas informações extraídas ainda são frutos de longo e árduo garimpo manual, caso a caso. Pesquisas e correlações simples são prejudicadas por não ser possível encontrar algo quando o repositório é desconhecido.

Convencionalmente existem três estruturas para acesso aos aplicativos: local, remota e web. No caso local mais trivial, os dados de cada setor são confinados localmente com acesso restrito e o compartilhamento ocorre manualmente através de relatórios dificultando amplo entendimento. Com acesso remoto, os dados são concentrados em banco unificado utilizado por vários aplicativos locais permitindo que lógicas não necessariamente coerentes operem os mesmos dados originando falhas de integridade. Sendo tendência atual, a abordagem web é mais robusta ao manter os dados e a lógica unificados em um sistema centralizado e amplamente coerente, expondo apenas a interface através do navegador.

Como referência, neste trabalho foram avaliadas experiências realizadas visando imprimir mais inteligência aos sistemas convencionais em funcionamento. Abordagens mais simples objetivam a construção manual de base de conhecimento digital e tratamento estatístico a campos de dados bem definidos em aplicativos. Textos são tomados como meros detalhadores da informação. Alternativas mais arrojadas arriscam a indexação e tratamento semântico para recuperar informação. Em contexto administrativo outras tentativas abordam inteligência negocial e agrupamento para potencializar a tomada de decisão.

Base de Conhecimento e Tratamento Estatístico

O aumento da disponibilidade de memória confiável e onipresença dos computadores criou a demanda pelo estabelecimento de bases de conhecimento digital em quase todos os negócios, não sendo diferente na saúde. Em trabalho publicado por Meenan e outros [11] foi implementado um sistema colaborativo estilo *wiki* para gestão do conhecimento da equipe de tecnologia da informação do setor de radiologia em um hospital norte-americano. Utilizando ferramentas livres, tinham como principal objetivo reduzir os tempos de diagnósticos e recuperação de falhas em sistemas de missão crítica.

A complexidade, heterogeneidade e o requisito de funcionamento ininterrupto dos sistemas presentes eram os principais desafios da equipe de suporte, superados com êxito pela criação, edição e acesso frequente às páginas criadas. A estratégia de gestão do conhecimento é apontada como cerne da sustentabilidade instituicional mais fortalecida contra alterações nas equipes e sugere essa iniciativa como boa alternativa também aos clínicos em necessidades específicas.

Outro estudo realizado por Hurlen em [8] utilizou dados disponíveis no prontuário eletrônico após integração com o sistema de informação radiológico para revelar aspectos comportamentais interessantes a respeito da geração e o consumo de informação hospitalar, tendo como objetivo principal verificar adaptação dos médicos utilizando novas rotinas de acesso aos laudos partindo da aquisição da imagem.

Foi observado que 12% dos laudos radiológicos finais não tinham sido acessados até 4 semanas após inserção no sistema, alguns poucos por razões técnicas. Para o restante verificou-se um tempo médio disponível de 1 hora nos preliminares e 4 horas para os finais. Na conclusão o autor aponta que o grande potencial do sistema não é utilizado plenamente pelo corpo clínico, por diversas razões, dentre elas a conduta médica e a usabilidade.

Experiência conduzida por Ramon e outros em [15] relacionada à mineração de dados provenientes de pacientes em tratamento intensivo aponta existência de ferramentas confiáveis para prever futuras complicações patológicas baseados nos dados disponíveis e normalmente não integrados. Com fontes heterogêneas como informações pessoais, histórico e aferições, a classificação e validação da informação traduz trabalho intensivo e árduo, devido a falta de padronização e sigilo.

Como desafio, Ramon salienta que, naturalmente, o intelecto humano é capaz de lidar com menor número de parâmetros que os disponíveis em unidades intensivas. Aponta uma solução razoável em um processo integrado que mescle a inteligência humana com parâmetros peculiares e subjetivos aliada à potência estatística e escalada de sistemas computacionais inteligentes, eficientes em filtrar ruídos e construir conhecimento assistido.

Recuperação da Informação

Um trabalho realizado por Erinjeri e outros [6] utilizou o software Google Desktop Enterprise[3] em servidor dedicado, para mineração de laudos radiológicos. Foram indexados aproximadamente 2,9 milhões de documentos a uma taxa média de 25.000 por hora. A primeira indexação durou aproximadamente 90 horas e as subsequentes mensais consomem por volta de 1 hora.

Como resultados uma média de 76 pesquisas são realizadas por dia com 85% retornando pelo menos um resultado e 15% falhando provavelmente devido a falhas sintáticas na busca ou erros gramaticais. O tempo médio de cada pesquisa foi 1,56 segundos e não se mostrou vinculado ao número de resultados retornados mas ao número de termos pesquisados e a unicidade de cada um. Segundo os autores, a evolução do sistema no processamento de linguagem natural potencializaria pesquisas por idéias representadas pelos termos ao invés de uma lista simples de palavras. Observou-se rápida adoção do sistema de buscas mesmo com pouca divulgação no corpo clínico, obtendo cerca de 64% das pesquisas no horário de trabalho.

Na mesma linha de trabalho outro projeto implementado por Voet e outros [18] usando ferramenta livre indexou 104 milhões de palavras distribuídas em 1,8 milhão de documentos, independente do idioma. O sistema foi desenvolvido como um módulo paralelo para agilizar o acesso e reduzir impacto no sistema hospitalar sendo sincronizado via consultas SQL[4] . Devido a dificuldade de integração com soluções comerciais pela ausência de uma interface bem definida e demora no tempo de indexação, foi utilizada a ferramenta *Swish-e* que atendia aos requisitos e apresentava tempo de indexação razoável.

O procedimento de importação é longo envolvendo pré-processamento e conversão dos documentos importados, com duração inicial de 14 dias. Em seguida a indexação total, que é refeita toda noite, consome 76 minutos de processamento. Como desafios foram relatados problemas com hifenização de palavras durante a importação causado pela segregação das mesmas. Duas funcionalidades não puderam ser utilizadas devido a ausência de *hyperlinks*: robôs para atualização da base e classificação de popularidade (*rank*) de documentos. Como pontos negativos, a falta de semântica nas pesquisas por palavras-chave somada a impossibilidade de estabelecer vínculo significativo entre palavras de raíz comum limitam a importante funcionalidade de busca por conceitos implícitos e significativos ao corpo clínico. Foi bem utilizado por residentes que realizavam nove em cada dez pesquisas por palavra-chave muito específicas. O uso extensivo de ferramentas livres foi apontado como redutor do tempo de implementação e custo do sistema.

[3] http://desktop.google.com/enterprise/index.html

[4] *Structured Query Language*, ou Linguagem de Consulta Estruturada

Inteligência Gerencial

Objetivando adquirir conhecimento detalhado sobre os fatores envolvidos em processo particular ampliando a visão da interação inter-processual, o trabalho de Prevedello e outros em [14] descreve conceitos originais de inteligência negocial aplicados à radiologia moderna pela integração de bases de dados. Neste cenário, a informação disponível em vários sistemas é combinada para fornecer informações úteis ao trabalho cotidiano, sem gerar inconsistência e degradação do desempenho.

Uma vez carregados, os dados seriam organizados para representar informações multidimensionalmente em cubos ao invés de tabelas permitindo análise rápida e interpretável do dado bruto. Com testes em bases fictícias, o autor aponta conceitos genéricos plausíveis de serem aplicados à radiologia e salienta a escolha de indicadores de desempenho ponderada pela perspectiva técnica e financeira.

1.4 Organização do Texto

Esta dissertação está estruturada em seis capítulos.

Este **Capítulo 1** contextualiza o cenário moderno da abundância de dados e motiva solução proposta. A revisão bibliográfica traz aspectos convencionais das ferramentas atuais e funcionalidades pesquisadas para embasar este texto.

No **Capítulo 2** a mineração de texto é abordada em suas etapas. As seções exploram métodos de extração transformação e carga, retirada de *stop-words* e raízes de palavravas no *stemming*. Ao final é mostrada a montagem de matrizes termo-documento e aperfeitçoamentos conhecidos para aumentar a eficiencia do processo.

O **Capítulo 3** estuda a decomposição em valores singulares para matrizes esparsas. A esparsidade é considerada no armazenamento e convergência do algoritmo de Arnoldi-Lanczos. São avaliadas implementações em software e plataformas. Requisitos de hardware como memória, processamento, e transmissão são discutidos enfatizando a característica da carga de trabalho.

No **Capítulo 4** o quesito legal do ambiente médico é tratado abordando dificuldades técnicas e implementações desenvolvidas. O processamento, dividido nos módulos de extração e indexação é abordado juntamente com a recuperação da informação pelas interfaces de busca.

O **Capítulo 5** mostra resultados práticos de desempenho em métricas objetivas definidas. Uma prova de conceito é utililizada seguida de experimentos práticos reais e comentários referentes a experiência de uso.

O **Capítulo 6** traz aspectos conclusivos relacionados à viabilidade e utilidade da ferramenta desenvolvida e do conhecimento produzido neste trabalho. Os aspectos inovadores alcançados e trabalhos futuros a partir desta obra finalizam o texto.

Capítulo 2

Mineração de Texto

O processo de minerar texto pode ser, essencialmente, manual ou automático. Manualmente, uma pessoa qualificada lê, interpreta e indexa os documentos seguindo uma lógica convencionada. Naturalmente, essa abordagem é cara, demorada e dificilmente escala em grandes massas, envolvendo ainda aspectos subjetivos relacionados ao tipo de instrução e personalidade do indexador. Como exemplo, uma experiência conduzida por Cleverdon em [2] relata a indexação feita por grupos distintos de pessoas numa base comum na qual apenas 60% dos termos indexados coincidiram. Quando automático, o trabalho é realizado por computadores em rotinas objetivas e programadas. O processamento rápido, de baixo custo e escalonado transformou este processo em tendência atual.

2.1 Indexação por Estrutura de Links

Com base na idéia que a estrutura em rede, originada a partir de documentos interligados por referências mútuas, é uma rica fonte de informação a respeito do conteúdo daquele ambiente, alguns algoritmos modernos de mineração foram projetados tendo a estrutura de grafo da web (*links*) como pano de fundo.

2.1.1 HITS

Formalizado em 1998 por Jon Kleinberg, o algoritmo HITS[1] particiona o conjunto de documentos em "detentores" (*authorities*) que são documentos que possuem informação significativa e "apontadores" (*hubs*) que possuem pouca ou quase nenhuma informação, servindo mais como pontos de passagem no estabelecimento dos vínculos na rede. Nesse cenário, é suposto que bons detentores são referenciados por bons apontadores e bons apontadores referenciam bons detentores. A indexação trata da construção de um grafo (matriz de vizinhanças) que mapeia os vínculos entre

[1] *Hyperlink-Induced Topic Search*

documentos. Consultas são realizadas pela utilização do grafo (lista invertida, por exemplo) de vizinhança associado aos termos presentes na pesquisa desejada, sendo em seguida calculados os pesos de cada detentor e apontador presentes no grafo para retornar ao usuário uma lista com aqueles de maior relevância. A máquina de busca Teoma (www.teoma.com) é exemplo prático de utilização deste algoritmo.

Como principal vantagem, este método apresenta ao usuário listas distintas dos detentores e apontadores, permitindo identificar com mais clareza as fontes da informação ampliando a noção sobre disponibilidade da mesma. Como contrapontos este algoritmo realiza parte significativa do processamento após a consulta, por ser conceitualmente dependente desta, tendo como consequência tempo de resposta inaceitável na maioria dos casos. A adequação da matriz de vizinhanças nas atualizações tem alto custo sendo ainda facilmente burlada pela criação de documentos parasitas com referências mútuas para elevar a relevância das mesmos. Somado a isso, problemas de desvio do foco original (*topic drift*) podem acontecer quando boas autoridades com elevados índices apontam erroneamente para outros documentos de assunto descorrelacionado.

2.1.2 PageRank

Cerne da máquina de busca do Google, o algoritmo do PageRank foi inicialmente desenvolvido por seus fundadores, Page e Brin. Exposto em [12] apresenta a vantagem de indexar previamente documentos (páginas) independentemente das consultas futuras a serem realizadas. Neste novo modelo a importância de determinado documento é estabelecida pela quantidade de referências importantes que o apontam e ao mesmo tempo pelo número de referências realizadas por estes apontadores.

Através da indexação antecipada de documentos recuperados pela exploração exaustiva e robotizada dos vínculos (*links*) uma grande matriz não negativa (BigTable) que relaciona os termos aos documentos é gerada. Para contornar dificuldades com documentos que não referenciam nenhum outro, perturbações são inseridas na matriz gerando uma cadeia de Markov irredutível[2] garantindo convergência do algoritmo de ranqueamento nas consultas. Os resultados são avaliados através de operações matriz-vetor de rápida execução.

Como vantagens, esta estratégia permite a paralelização de multiplicações esparsas entre matrizes e vetores, o particionamento da matriz em blocos além do uso de técnicas de extrapolação e agregação para acelerar a convergência nas pesquisas. Nas desvantagens, este algoritmo também apresenta a dependência estrita da estrutura de vínculos entre os documentos e baixa eficiência na recuperação de informações menos populares ou fracamente referenciadas. Uma consequência natural

[2]Estrutura na qual qualquer estado (documento) pode ser alcançado a partir de qualquer outro em um número finito de passos.

deste fato é que documentos publicados porém não referenciados são desconhecidos na rede dificultando estimar parte dos dados disponíveis sendo reconhecida pelo jargão "*dark web*".

2.2 Indexação Semântica Latente

A praticidade e eficiência dos métodos baseados em estrutura de *links* são pontos estabelecidos pelas inúmeras aplicações disponíveis. No entanto, parte significativa da informação é produzida fora desta estrutura e requer um tratamento especializado para mineração. Neste contexto a indexação semântica latente ou LSI[3] é um método que busca identificar padrões e relacionamentos entre termos e conceitos contidos em uma coleção não estruturada de documentos, baseado no conceito que palavras utilizadas no mesmo contexto tendem a ter significados semelhantes.

Inicialmente desenvolvido no final dos anos 1980 nos laboratórios Bell, este método utiliza decomposições matriciais para filtrar e extrair significado latente de textos em resposta a consultas conceituais, ainda que estes não compartilhem as mesmas palavras. O tratamento matemático dispensado traz vantagens claras e tem contraponto no maior custo computacional requerindo ponderações na implementação para bases de dados significativamente grandes. Neste cenário, otimizações algébricas (utilizando esparsidade e multiplicações matriz-vetor) e computacionais (paralelismo e distribuição) colaboram em ampliar o potencial desta ferramenta.

LSI apresenta vantagens significativas no tratamento de vários ítens:

- **sinônimos e polissemia:** respectivamente múltiplas palavras com mesmo significado e palavras com múltiplos significados.
- **classificação:** uso de correlação para categorizar documentos estabelecendo padrões semelhantes ao processo humano.
- **agrupamento:** dinâmico e baseado no conteúdo semântico similar dos documentos, sem necessitar de modelos de referência.
- **independência de linguagem:** abordagem matemática estrita desobriga a utilização de dicionários e considera palavras presentes em idiomas variados.
- **irrestrito a palavras:** qualquer objeto passível de representação vetorial (imagens, por exemplo) pode ser considerado, viabilizando entradas diversas.

Neste contexto, focado na mineração de textos médicos em grandes coleções completamente mapeadas, o método de LSI foi eleito para implementação deste trabalho.

[3] *Latent Semantic Indexing*, também referido como *Latent Semantic Analysis (LSA)*

2.3 Preparação dos Documentos

Intuitivamente é aceitável que a grande maioria dos documentos não é concebida com propósitos estritos de mineração de dados. Isso traduz-se em ausência de padrões, grande variedade de formatos e codificações, dando origem, em alguns casos, a coleções heterogêneas interna e externamente. Nesse contexto a preparação de documentos é uma sequência de processos visando homogenizar os dados de entrada preparando-os para a etapa posterior de indexação propriamente dita.

2.3.1 Análise

No desenvolvimento de mineradores a etapa de análise é conhecida pelas três fases principais que a compõe: extração, transformação e carga ou simplesmente ETL[4]. Por extração entende-se o trabalho de recuperar os dados do seu repositório original, levando em consideração a distribuição estrutural e geográfica. A fase de transformação envolve três processos principais relacionados:

- **Formatação:** Adequação sistemática de codificação (ASCII[5], UTF-8[6] , etc) padronizada para tratar textos nos possíveis formatos de entrada e estabelecimento de regras para a manipulação de imagens, tabelas e outros objetos de interesse vinculados.
- **Validação:** Verificação estrutural, após formatação, quanto ao atendimento de padrões específicos, verificação de campos e inconsistências. Intuitivamente esse trabalho é focado em partes pré-estabelecidas como alvo de indexação (*zoning*), sendo menos rigoroso com o restante do conteúdo.
- **Normalização:** Processo realizado sobre a menor e mais filtrada parte dos documentos e remete a identificação de termos essenciais e tratamento de sinais especiais comumente presentes como espaços em branco, formatadores e pontuação em geral.

A fase de carga segue a transformação. Seu objetivo principal é estabelecer um novo armazenamento estruturado, confiável e de rápido acesso a massa de dados homogenizada e preparada para indexação. Neste ambiente, a velocidade no acesso ao novo conteúdo é decisiva no desempenho do algoritmo. A noção de transparência no acesso oferecido ao usuário final é resultado da eficiência desta fase e colabora em relativizar a fonte real da informação. Uniformemente preparada e armazenada, a massa de dados está pronta para o início real da mineração.

[4] *Extract Transform and Load*
[5] *American Standard Code for Information Interchange*, codificação para idioma inglês de 8 bits.
[6] *8-bit Unicode Transformation Format* é uma codificação de caracteres de comprimento variável.

2.3.2 *Stop-Words*

Conceitualmente, esta etapa é considerada o início prático da mineração. Partindo de uma coleção de documentos homogenizada, tanto em relação ao conteúdo quanto ao acesso, o próximo passo trata da retirada/filtragem de palavras (*stop-words*) com valor semântico desconsiderável em consultas. Tais palavras são específicas para cada idioma e servem na maioria das vezes como conectores sendo geralmente dispostas em listas (*stop-list*).

A extração de *stop-words* dos documentos na formação dos índices oferece duas vantagens principais. A primeira está relacionada à economia de espaço no armazenamento da estrutura de indexação pela eliminação de partes frequentes e pouco significantes. A segunda e mais importante remete a otimizações no tratamento matemático posterior, uma vez que retirar as *stop-words* colabora reduzir a ordem do problema e consequentemente aprimorar os resultados.

O execução do processo é simples. Cada documento é serializado como uma sequência pura de palavras. A presença de cada palavra desta sequência é verificada na *stop-list*. Caso negativo nada é modificado e caso positivo aquela palavra deixa de fazer parte do conteúdo do documento **preparado**. O documento-exemplo: "matemática na computação e aplicações computacionais" teria o seguinte resultado:

matemática	→	matemática
na	→	
computação	→	computação
e	→	
aplicações	→	aplicações
computacionais	→	computacionais

Tabela 2.1: Demonstração do algoritmo de *stop-words*.

Neste trabalho foi utilizada a *stop-list* disponibilizada pelo projeto Snowball[7], com poucas alterações, para atender melhor requisitos específicos da área médica. É importante salientar que a palavra "**não**" está ausente desta lista, apesar de ser frequênte. Neste caso especial existe a idéia de negação envolvida que é muito significativa. Cada idioma opera com uma *stop-list* particular e o projeto Snowball disponibiliza listas para o português, inglês, alemão e francês, entre outros idiomas.

Caso particular que merece consideração especial são as palavras que aparecem somente uma vez em toda a coleção, denominados *singletons*. Nas experiências práticas foram observados percentuais significativos desses casos, na maioria das vezes vinculados a erros ortográficos. Até esta etapa tais particularidades não receberam tratamento diferenciado visando avaliação inicial simplificada.

[7] http://snowball.tartarus.org/

2.3.3 *Stemming*

Partindo de documentos mais compactos e semanticamente mais objetivos, resultado da retirada das *stop-words*, o processo de *stemming* trata da extração das raízes semânticas de cada palavra originando os "termos". A raiz de uma palavra é encontrada, na maioria das vezes, pela adequação do plural e retirada de prefixos e sufixos, reduzindo cada uma ao valor semântico essencial. Intuitivamente, este processo envolve um número considerável de regras linguísticas estando, dessa forma, fortemente vinculado ao idioma.

Bem definido, o processo de *stemming* traz benefícios na mineração. Permite corrigir erros gramaticais simples e avaliar plurais em consultas. Reduz consideravelmente o tamanho do índice e também o esforço computacional através da aglutinação de várias palavras em apenas um termo menor. Otimiza na maior parte das vezes o conteúdo semântico aumentando a precisão dos resultados, salvo quando a retirada de prefixos e sufixos desvia a palavra da sua essência original.

Fazer *stemming* é mais complexo e custoso que retirar *stop-words*. Nesse sentido a sequência desses processos é estratégica, aliviando carga onde é mais crítico. Existem basicamente duas formas de extrair raízes: através de regras gramaticais programadas que consideram plurais e o posicionamento de vogais e consoantes nas palavras e utilizando dicionários nos quais cada palavra é comparada numa lista buscando o termo "mais adequado" para aquele caso. O mesmo documento-exemplo anterior mostra o resultado após a retirada de *stop-words* seguido do *stemming*:

matemática	→	matemátic
na	→	
computação	→	**comput**
e	→	
aplicações	→	aplicaç
computacionais	→	**comput**

Tabela 2.2: Demonstração do algoritmo de *stemming* após *stop-words*.

O algoritmo utilizado para implementação do *stemming* também foi disponibilizado pelo projeto Snowball e usa a estratégia de programação de regras gramaticais em tempo de compilação o que tornou a execução mais rápida comparada ao uso de dicionários. O projeto citado usa uma estrutura prática e robusta para criação dos algoritmos. Inicialmente as regras linguísticas de cada idioma são escritas em um *script* intuitivo e neutro em relação a codificação computacional. Isso feito, um aplicativo interpretador traduz todas as regras em código fonte C (ANSI) ou Java prontos para serem adapatados e compilados atendendo as particularidades de cada aplicação. Ponto interessante viabilizado por esta ferramenta é o suporte a vários idiomas em uma codificação neutra, reutilizável e independente de plataforma.

2.4 Modelo de Espaço Vetorial

De acordo com Berry em [1], o modelo de espaço vetorial é uma representação matemática de **termos** e **documentos** numa **coleção** de textos. Nesta abordagem, cada componente do vetor-documento traduz numericamente a importância semântica de um termo presente no mesmo. A coleção é modelada por meio de uma matriz chamada **termo-documento**.

2.4.1 Construção da Matriz Termo-Documento

Uma coleção composta de **n** documentos indexados por **m** termos pode ser representada por uma matriz termo-documento **A** de ordem $\mathbf{m}\times\mathbf{n}$. Os vetores-documento estão dispostos como colunas na matriz **A** e cada elemento $\mathbf{a_{i,j}}$ representa a frequência ponderada que o termo **i** ocorre no documento **j**.

Neste contexto, o espaço coluna de **A** determina essencialmente o conteúdo semântico da coleção. Entretanto não existe uma interpretação específica para cada vetor representado no espaço coluna, uma vez que a combinação de quaisquer dois documentos não produz necessariamente um documento viável. Utilizar esta estrutura possibilita explorar poderosas relações geométricas e algébricas entre termos e documentos (vetores) para avaliar semelhanças e diferenças semânticas de conteúdo. Como exemplo, a tabela 2.3 mostra a preparação (análise + *stop-words* + *stemming*) de uma pequena coleção fictícia de documentos.

Doc.		Conteúdo Original		Termos
D1	$\rightarrow$	Matemática para Controle	$\rightarrow$	matemát control
D2	$\rightarrow$	Controladores Computadorizados	$\rightarrow$	control comput
D3	$\rightarrow$	Matemática nos Computadores	$\rightarrow$	matemát comput
D4	$\rightarrow$	Controle Matemático e Computacional	$\rightarrow$	control matemát comput

Tabela 2.3: Exemplo de preparação em pequena coleção.

Uma vez preparados, os documentos são estruturados na matriz da equação (2.1).

$$A = \begin{array}{c} \\ matemát \\ control \\ comput \end{array} \begin{array}{c} \begin{array}{cccc} D1 & D2 & D3 & D4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \end{array} \tag{2.1}$$

Neste ponto, a frequência simples de ocorrência do termo no respectivo documento foi atribuída a célula da matriz termo-documento, sem outras considerações. No entanto, é intuitivo e defendido por Berry em [1] e Dumais em [3] que ponderar os termos otimiza potencialmente o desempenho na recuperação da informação.

Visando aprimorar este processo, Berry em [1] sugere a ponderação dos elementos da matriz termo-documento no formato mostrado na equação (2.2), para parametrizar aspectos relevantes na recuperação.

$$a_{ij} = l_{ij} g_i d_j \tag{2.2}$$

O fator l_{ij} representa o peso local para o termo i presente no documento j e regula a importância de cada termo internamente ao documento, salientando sua essencia semântica. O parâmetro g_i reflete a ponderação global do termo i na coleção considerando conteúdos individuais no ambiente mais amplo da coleção e atuando como moderador da heterogenidade da base. Por sua vez, d_j especifica a normalização aplicada nos documentos e estabelece um patamar homogêneo na avaliação dos documentos. Modelos reconhecidos para equacionamento dos elementos foram disponibilizados por Berry em [1], sendo a_{ij} o produto final.

Com esse entendimento, neste exemplo básico será usada a frequencia de ocorrência simples no peso local sem considerar ponderação global e normalização, assumindo portanto valor unitário. Neste contexto, o valor de a_{ij} é dado pelo número de ocorrências do termo i no documento j. Intuitivamente esta configuração simples atende bem a abordagens iniciais em bases de dados desconhecidas e serve de parâmetro de comparação na avaliação de outros modelos mais sofisticados.

Exemplos de ordem três (ou menor) permitem visualização gráfica da coleção de documentos mapeada em espaço vetorial como na figura 2.1.

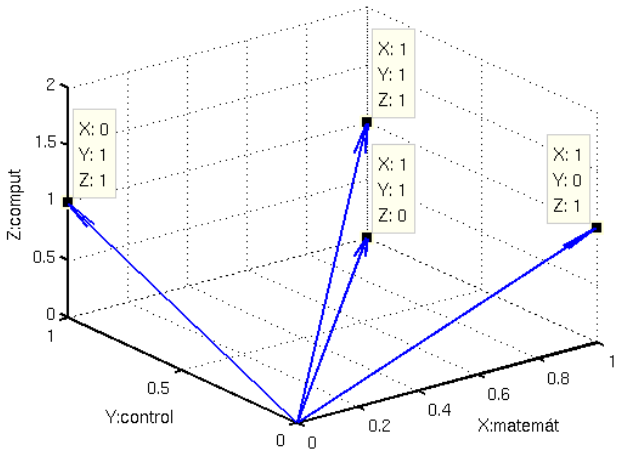


Figura 2.1: Modelo visual do espaço vetorial exemplificado.

Nesta figura é perceptível a separação dos vetores-documento mapeados no espaço vetorial. como resultado de suas particularidades.

2.4.2 Processamento de Consultas

Para coleções de documentos de dimensão elevada, o trabalho desde a preparação até a construção da matriz termo-documento é computacionalmente custoso e independe das consultas que venham a ser feitas. Por esse motivo é executado previamente em batelada nos indexadores. De outra forma, inviabilizaria a recuperação de informação pelo processamento sob demanda a cada consulta. Por este motivo, máquinas de busca populares realizam esta tarefa durante as madrugadas consumindo horas de processamento em supercomputadores.

O processamento de consultas, por sua vez, depende da entrada fornecida pelo usuário e sua computação inicia imediatamente após a submissão da pesquisa. Pela prática comum do desenvolvimento desacoplado de indexadores e buscadores, vale salientar necessidade de coerência operacional (mesmas regras linguísticas e algébricas) entre estes mecanismos. Uma consulta é uma mineração com objetivo específico de recuperar informação de interesse e pode ser dividida em três fases conceituais.

- **Formulação:** O usuário formula a consulta e submete ao sistema atentando para funcionalidades disponíveis. Como exemplo será submetida a consulta:

  algoritmos computacionais para controle de computadores.

- **Conversão:** A máquina de busca converte a consulta em termos válidos e gera o vetor-consulta. Por termo válido deve ser entendido todos os termos presentes simultaneamente na consulta e no índice (linhas) da matriz termo-documento construída durante o processo de indexação. No exemplo fornecido:

$$\begin{array}{lcl} \text{algoritmos} & \rightarrow & \\ \text{computacionais} & \rightarrow & \text{comput} \\ \text{para} & \rightarrow & \\ \text{controle} & \rightarrow & \text{control} \\ \text{de} & \rightarrow & \\ \text{computadores} & \rightarrow & \text{comput} \end{array} \quad \Longrightarrow \quad \begin{array}{c} \\ \\ freq \\ matemát \\ control \\ comput \end{array} \begin{pmatrix} \\ \\ 0 \\ 1 \\ 2 \end{pmatrix} = q$$

  Importante notar que a palavra "algoritmo" presente na consulta não seria considerada, mesmo sem pertencer à *stop-list* pelo fato de não ter sido indexada (pertence à uma dimensão desconhecida) previamente.

- **Recuperação:** O vetor-consulta (q) construído é uma entidade matemática semelhante a um vetor-documento podendo ser comparado aos demais na base para avaliar a relevância da consulta. Calcular o ângulo entre vetores é uma forma eficiente e usual para avaliar proximidade (semelhança) entre eles. Ângulos menores significam vetores mais próximos e representam portanto conteúdo semântico semelhante. O valor do cosseno, dado pela equação (2.3) é geralmente utilizado no lugar do ângulo em si pela eficiência computacional.

$$\cos(\theta_j) = \frac{a_j^T q}{||a_j||_2 ||q||_2} = \frac{\sum_{i=1}^{m} a_{ij} q_i}{\sqrt{\sum_{i=1}^{m} a_{ij}^2} \sqrt{\sum_{i=1}^{m} q_i^2}} \tag{2.3}$$

Na equação (2.3) o cálculo do cosseno do ângulo entre os vetores requer a avaliação das normas de cada vetor-documento ($||a_j||_2$), a norma do vetor-consulta ($||q||_2$) e um produto interno simples ($a_j^T q$) entre o vetor-documento considerado e o vetor-consulta. Estrategicamente, todas as normas são conhecidas a priori sendo unitárias, caso todos vetores-documento tenham sido normalizados durante a indexação e o vetor-consulta também o foi durante a fase de conversão. Resta portanto o cálculo do produto interno com custo linear, proporcional ao número de termos indexados.

Para o exemplo na tabela 2.3 também é possivel ilustrar graficamente a avaliação dos ângulos na consulta, como mostrado na figura 2.2

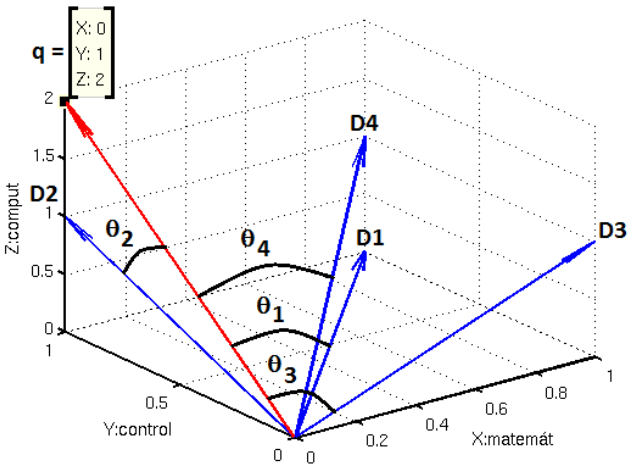


Figura 2.2: Modelo visual da avaliação dos documentos por ângulo.

Visualmente nota-se que o vetor-consulta (q) é mais semelhante ao vetor-documento 2. Algebricamente esse resultado é confirmado pelo cálculo dos cossenos:

$$\cos\theta_1 = 0.3162; \quad \cos\theta_2 = 0.9487; \quad \cos\theta_3 = 0.6325; \quad \cos\theta_4 = 0.7746$$

Valores de cosseno próximos de 1 (≈ 0 rad) indicam semelhança e próximos de 0 ($\approx \pi/2$ rad) indicam o contrário. Em máquinas de busca reais, com milhares de documentos indexados, geralmente é tomado um valor limite no comparador. Acima deste gatilho os documentos são retornados em ordem de relevância, sendo considerados irrelevantes caso contrário. A escolha deste valor é fundamental para alcançar uma relevância adequada e será discutida oportunamente.

2.4.3 Tipos de Consultas

Constituídas essencialmente por palavras as consultas se diferenciam em estrutura, na maneira com que são construídas e avaliadas, considerando seu objetivo.

Consultas Lógicas ou Booleanas

São pesquisas por palavras vinculadas usando operadores lógicos, do tipo "E" (adição), "OU" (combinação) e "NÃO" (exclusão). Cada operador acrescenta lógica às palavras que afeta, na sequência e prioridade de utilização. Como exemplo, uma pesquisa por <uma palavra> "E" <outra palavra> exige que ambas estejam presentes nos resultados, ao passo que uma consulta por <uma palavra> "OU" <outra palavra> admite que apenas uma delas esteja presente no resultado podendo ainda haver combinação das mesmas. O operador "NÃO" geralmente força a exclusão de resultados que contenham a palavra argumento.

Como desvantagem estas consultas aparentemente não representam boa estratégia para imprimir significado nas pesquisas. Korfhage em [10] sugere que a maioria dos usuários de sistemas de informação não estão bem treinados no uso deste artifício, a menos que sejam cientistas ou matemáticos. Em sistemas baseados no modelo de espaço vetorial geralmente esta abordagem não é utilizada.

Consultas em Linguagem Natural

Consultas em linguagem natural ou NLQ[8] são formuladas como perguntas ou afirmações simples. Um exemplo típico numa máquina de busca moderna seria: "Como instalar driver de rede no Linux?". Para processar pesquisas assim é necessário primeiro extrair os termos de valor semântico significativo e depois buscá-los no índice, geralmente desconsiderando sua ordem.

Como desvantagem, também segundo Korfhage em [10], esta abordagem esbarra na "dificuldade computacional de extrair termos mantendo sua integridade sintática, semântica e pragmática intacta". Em outras palavras, quando termos são extraídos de uma NLQ perde-se o contexto no qual a palavra foi utilizada. Este efeito é mais acentuado em palavras polissêmicas[9].

Consultas Utilizando Vocabulário Controlado (Thesaurus)

Uma consulta com vocabulário controlado é aquela na qual o usuário seleciona palavras/termos para pesquisa a partir de uma lista predefinida pelo sistema. Esta estratégia tem a vantagem explícita de oferecer termos já preparados ao usuário, podendo expandir conceitos relacionados.

[8] *Natural Language Query*
[9] Múltiplas palavras com mesmo significado

Como contraponto esse modelo limita pesquisas a palavras do vocabulário previamente mapeado (podendo utilizar dicionário) e não permite a escolha livre pelo usuário. Apresentam também tendência de utilização de termos genéricos não necessariamente indexados naquela coleção.

Consultas Difusas ou *Fuzzy*

São consideradas uma extensão da lógica booleana que admite valores lógicos intermediários e engloba, de certa forma, conceitos estatísticos principalmente na área de inferência. Pesquisas difusas em sistemas de recuperação da informação remetem a capacidade de lidar com erros ortográficos e variações nas mesmas palavras. Estas consultas podem também ser interpretadas como um conjunto de documentos resultado de uma busca convencional e expostos a uma avaliação de relevância em alguma lógica difusa particular de interesse do usuário.

Consultas Por Termos

De acordo com Berry em [1], talvez a consulta por termos seja a mais prevalente, especialmente na Internet, quando o usuário fornece algumas palavras ou frases para pesquisa. Neste aspecto, o uso de frases induz a adição semântica dos conceitos presentes em cada termo (eixos principais), semelhante ao operador "E" lógico, porém sem implicar na presença obrigatória dos termos. Este método apresenta melhor precisão relativa tendo como efeito colateral a possível perda de resultados menos específicos. Consultas maiores tendem a retornar melhor resultado em modelos de espaço vetorial enquanto as menores funcionam melhor no caso lógico.

Quando disponível, usuários experientes podem também utilizar operadores de proximidade, escolhendo as distâncias máximas entre os termos fornecidos e melhorando o entendimento de frases. Essa funcionalidade exige armazenamento estruturado dos documentos indexados sendo também mais custosa computacionalmente.

Consultas Probabilísticas

Estão relacionadas a forma de avaliação da relevância dos documentos para uma dada pesquisa. Métodos mais convencionais como consultas lógicas (coincidência exata) ou pesquisa por termos (espaço vetorial) oferecem critérios rígidos e sistemáticos nesse julgamento. Esta modelagem tem a premissa de estabelecer uma forma de avaliar também as incertezas no processo de recuperar informações.

Como vantagem sobre as consultas difusas (*fuzzy*) esta abordagem oferece um conjunto de métodos bem estabelecidos e testados para calcular probabilidades e frequências. Aos métodos mais sistemáticos e inflexíveis traz certa suavização nos resultados.

2.4.4 Aproximações de Baixa Ordem

Modelar coleções de documentos em espaço vetorial é uma ferramenta poderosa e fonte aquecida de pesquisas. Dado o crescente volume de informação produzida, o custo do tratamento em larga escala e a eficiência dos métodos merecem atenção. Neste ambiente, as aproximações de baixa ordem traduzem artifícios matemáticos eficientes e bons em desempenho, principalmente em grandes massas.

No exemplo abordado, a coleção de documentos foi mapeada numa pequena matriz tratada com rapidez num computador moderno. Operações vetoriais duraram frações de segundos, ocupando pouca memória. Em aplicações práticas porém, as proporções tendem a ser maiores elevando consideravelmente o processamento e o requisito de memória. Sistemas de gestão simples, disponíveis em hospitais, acumulam centenas de milhares de documentos com milhares de termos indexáveis.

Matematicamente as aproximações de baixa ordem visam resolver os mesmos desafios anteriores utilizando soluções matriciais mais compactas e leves. Uma forma simples de "reduzir" a carga, simplificando o problema, seria desprezar alguma(s) dimensões julgadas irrelevantes por algum critério. A figura 2.3 mostra graficamente o espaço vetorial ao desprezar o eixo "X:matemát". Visualmente o novo problema bidimensional requer operações em vetores menores e computacionamente mais leves.

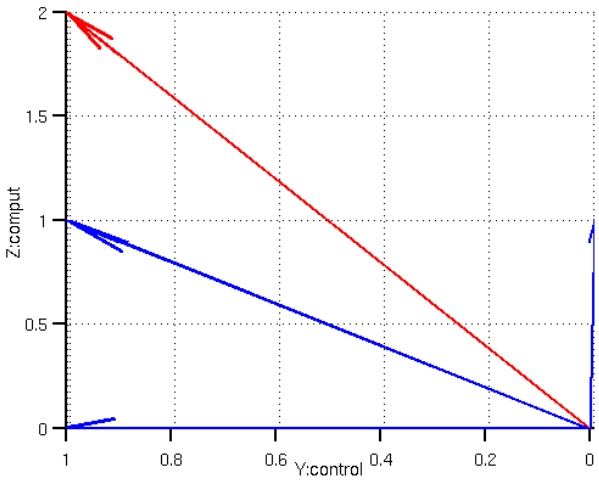


Figura 2.3: Exemplo de espaço vetorial de ordem reduzida.

Visualmente, a redução de dimensões ignorando termos/eixos causa problemas semânticos graves entre documentos que se diferenciavam naquelas dimensões e até o desaparecimento de documentos que possuem componentes exclusivamente nos eixos ignorados. Técnicas mais elaboradas e eficiêntes para reduzir dimensões existem, são largamente utilizadas e serão detalhadas oportunamente.

2.4.5 Atualização da Coleção

Documentos que constituem uma coleção são naturalmente criados, editados e apagados ao longo do tempo, fenômeno que se denomina volatilidade. Estas alterações contínuas na coleção implicam na necessidade constante de adequar a abstração matemática gerada na indexação para refletir mais fielmente as características do novo conjunto. Por este prisma, a mineração se configura como um processo recorrente e intimamente vinculado aos dados.

Sistemas mineradores podem trabalhar de forma *off-line* ou *on-line*. No primeiro caso, são implementados geralmente em módulos independentes e operam sobre uma cópia privada da coleção real atualizada periodicamente com baixo impacto na coleção original, porém com maior risco de inconsistência dos dados. No segundo caso, implementações integradas de estruturas mais elaboradas dispensam a criação da cópia privada atuando concorrentemente na coleção original com melhor a eficiência temporal e capacidade de reagir sob demanda à alterações mais significativas.

Independente da forma operacional, mas parametrizadas pela volatilidade dos dados e custo total as atualizações podem operar sobre porções selecionadas do conjunto otimizando tempo e recursos consumidos sem perder desempenho considerável no tratamento da informação.

- **Completa:** A medida que o conjunto de documentos é modificado significativamente, uma nova indexação toma lugar refazendo todos os índices e atualizando por completo o modelo abstrato anterior. Esse processo é comparativamente mais simples e preciso sendo porém mais custoso e demorado.

- **Segmentos:** Coleções pouco voláteis nas quais a informação é pouco editada e frequêntemente adicionada é razoável avaliar a segmentação do conjunto para mineração. Idealmente particionada em sub-conjuntos de documentos (antigos e novos) de acordo com as alterações ocorridas. A parte antiga e muito pouco volátil é mantida indexada sendo atualizada com menor frequência, enquanto a parte mais nova e menor é atualizada frequêntemente. Apesar da complexidade adicional em lidar com mais índices (segmentos) essa abordagem oferece baixo custo computacional e boa eficiência nos resultados.

- **Aproximações:** Considerando o uso de aproximações redutoras de ordem, é razoável avaliar a atualização dos índices já calculados pela "inserção" de informações relativas apenas aos documentos adicionados ou modificados sendo computacionalmente mais leve. Esta abordagem, no entanto, deve ser avaliada com cautela para coleções altamente voláteis uma vez que resulta numa indexação inexata (aproximada) da nova coleção.

2.5 Avaliação de Eficiência

Avaliar desempenho em mineradores de texto é uma questão complexa dada a inexistência de critérios universalmente reconhecidos e somada a subjetividade inerente dos parâmetros passíveis de serem coletados. Para o caso mais específico de máquinas de busca, a pergunta fundamental a ser respondida é: A informação necessária foi encontrada em tempo hábil? Uma tentativa de fornecer reposta mais objetiva para essa indagação originou métricas quantitativas aferíveis também em coleções menores.

2.5.1 Precisão

A precisão de um método de busca é definida como mostra a equação (2.4).

$$\text{precisão} = \frac{\text{documentos relevantes recuperados}}{\text{total de documentos recuperados}} \tag{2.4}$$

Notadamente o denominador da equação (2.4) é uma grandeza objetiva e simples de aferir, bastanto contar os ítens na lista de retorno da consulta. Este número, por sua vez, está intimanente vinculado ao valor limite adotado nos cossenos dos ângulos da equação (2.3). Quanto mais próximo de um estiver o valor limite do cosseno mais rígido será o critério de semelhança (vetores mais próximos) e portanto menos documentos constarão na lista retornada.

Por outro lado, a avaliação do numerador da equação (2.4) é muito subjetiva e dependente do julgamento pessoal do avaliador. Como exemplo, dois usuários podem submeter exatamente a mesma consulta (mesmos termos) em um dado momento e receber a mesma lista como resultado. A quantificação dos documentos relevantes no conjunto está submetida a critérios individuais de cada um, podendo diferir significativamente.

Um tratamento estatístico adequado, realizado com um grupo maior de usuários diferentes é uma estratégia razoável para contornar problemas de subjetividade e utilizar a média como métrica. Em máquinas de busca modernas, principalmente as que operam na web, é comum perceber tentativas de colher alguma classificação do usuário sobre a pesquisa feita, como forma de aprimorar a precisão e consequentemente obter melhores resultados.

Coleções de documentos menores e humanamente tratáveis são também utilizadas com finalidade de testes. Neste método é definida uma coleção completamente conhecida e mapeada, em consenso com uma equipe qualificada, sobre a qual são executadas consultas específicas para avaliação da precisão do método. O resultado, ainda que influenciado pelo tamanho da coleção, serve como indicativo interessante da precisão do algoritmo implementado.

2.5.2 Relevância

A relevância de um método de busca é definida como mostra a equação (2.5).

$$\text{relevância} = \frac{\text{documentos relevantes recuperados}}{\text{total de documentos relevantes na coleção}} \tag{2.5}$$

Diferente do critério de precisão, o denominador da equação (2.5) é uma grandeza subjetiva e difícil de aferir. A quantidade total de documentos relevantes para uma determinada pesquisa é característica intrínseca da coleção e independe, teoricamente, dos algoritmos usados na mineração. Desvinculado do valor limite adotado nos cossenos dos ângulos da equação (2.3), a estimação desse número somente é razoável por análise individual em coleções fechadas e menores.

Como as equações de precisão e relevância compartilham o numerador, as mesmas características se aplicam neste caso. A consequência direta da subjetividade inerente ao processo exige utilização cautelosa (estatística) deste critério para comparar desempenho entre sistemas diferentes.

2.5.3 Realimentação

Segundo Korfhage em [10], o diálogo entre usuário e sistema é fator primordial para o bom desempenho na recuperação da informação. O cerne da idéia de interatividade está em aproveitar cada consulta realizada pelo usuário para coletar a opinião (julgamento) deste a respeito da precisão e relevância do resultado anterior. Com essa realimentação é possível ponderar dinamicamente conteúdos por peso e popularidade aumentando a inteligência do sistema para, em conjunto com algoritmos sofisticados, obter desempenho aprimorado.

Um sistema minerador para recuperação de informação ideal deve apresentar alta precisão em todos os níveis de relevância como defende Berry em [1]. Neste aspecto, a abordagem por espaço vetorial oferece vantagem significativa quando classifica e pondera semanticamente os documentos evitando pesquisas com retorno vazio e fazendo com que os primeiros resultados forneçam um bom ponto de partida para exploração da informação.

Na prática a realimentação dos sistemas de busca pode ocorrer de forma manual ou automática, não sendo, necessariamente, excludentes. No caso manual, o usuário escolhe novos termos para pesquisas subsequentes baseado nos resultados recuperados na tentativa anterior, aprimorando com isso o conjunto de termos significativos explorados. No modo automático, geralmente alguns resultados são marcados para visualização de documentos semelhantes, o que permite o sistema ponderar vetorialmente novas consultas e evitar resultados repetitivos. Em princípio esse processo converge no encontro da informação desejada com menor esforço.

2.6 Aprimoramentos

As estratégias citadas para mineração de dados apresentam resultados satisfatórios em usos genéricos. No entanto, aplicações mais específicas e exigentes estimularam o desenvolvimento de métodos aprimorados em eficiência (precisão e relevância) e desempenho (rapidez e confiabilidade) na recuperação da informação.

2.6.1 Dicionário de Sinônimos

A presença de sinônimos e palavras polissêmicas é comum em grandes massas de dados de conteúdo não específico. Tais palavras degradam a eficiência das consultas pois possuem o mesmo conteúdo semântico sob diferentes formas ou conteúdos diferentes em formato (vetor) semelhante.

Um dicionário de sinônimos bem implementado funciona como uma lista de palavras estrategicamente definidas que podem eventualmente substituir ou complementar termos em uma consulta amenizando ou enfatizando diferenças semânticas tendendo a melhorar a eficiência na mineração dos dados.

2.6.2 Paralelização

Grandes coleções de documentos, além de ocuparem mais espaço ainda requerem maior carga de processamento e consequentemente consomem mais tempo durante a indexação e o avaliação de consultas. Nesse cenário, a paralelização destes processos colabora em estabilizar o desempenho na mineração frente a carga de trabalho crescente. O contraponto de aumento da complexidade do sistema é compensado pela maior escalabilidade alcançada.

2.6.3 Segmentação de Bases

Aumentando a complexidade, a segmentação do conjunto de documentos adiciona ao minerador a capacidade de distribuição de responsabilidades. Vinculada a paralelização, esta abordagem foca no tratamento de coleções extremamente grandes nas quais a paralelização local unificada não escalou para a indexação e consulta. Neste caso, geralmente é adotada uma estrutura distribuída em rede, com pontos de processamento trabalhando em sincronia preservando a integridade do todo.

Sub-conjuntos menores da coleção completa são tratados em máquinas distintas e depois unificados para fornecer um resultado homogêneo e coerente. Intuitivamente esta abordagem potencializa sistemas globais mas requer atenção para aspectos externos ao processo de mineração propriamente dito, como por exemplo, latência, largura e confiabilidade de banda da rede de interconexão.

Capítulo 3

Decomposição em Valores Singulares para Matrizes Esparsas

Modelar uma coleção de documentos em espaço vetorial é um artifício valioso. Neste cenário, decomposições matriciais apresentam custo variado sob condições específicas e aumentam potencialmente a eficiência na mineração trazendo vantagens:

- **Base Cartesiana Aprimorada:** Os principais métodos de decomposição visam essencialmente estabelecer bases cartesianas melhores para representar o conjunto de vetores (documentos). Características mais significativas ficam salientadas em eixos principais encontrados através da decomposição.
- **Redução de Dimensão:** Em base cartesiana otimizada, menos dimensões (eixos) são necessários para caracterizar cada elemento. Isso permite selecionar eixos principais significativos e operar em dimensão reduzida com eficiência e menor custo computacional nas manipulações algébricas.
- **Compactação da Informação:** Em dimensão reduzida (truncada) a informação permanece compactada uma vez que ocupa menos espaço (eixos) e mantém as características principais do seu conteúdo original.
- **Filtro de Ruído:** As perdas decorrentes da redução de dimensões afetam mais significativamente características indistinguíveis dos elementos que geralmente são tomadas como ruído. Neste contexto, a decomposição trucanda também funciona como um filtro seletivo sobre a base.

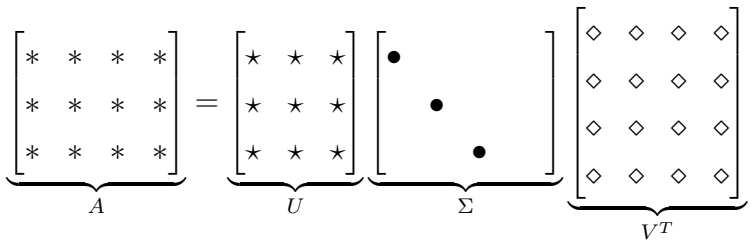
A decomposição em valores singulares ou SVD[1] opera com linhas e colunas de forma simétrica, salientando características dominantes ordenadamente. Esta qualidade, somada a eficiência e flexibilidade na redução dimensional tornou esta ferramenta muito útil na mineração de dados. Mesmo com custo comparativo superior, é frequentemente utilizada em algoritmos de LSI, como neste trabalho.

[1] *Singular Value Decomposition*

3.1 Decomposição em Valores Singulares

De acordo com Strang em [17], a SVD está intimamente associada à fatoração $Q\Lambda Q^T$ em autovalores e autovetores de uma matriz positiva definida. Sem as mesmas restrições (dimensionais e de positividade) este algoritmo essencialmente define vetores singulares que formam nova base ortogonal (ótima) ponderada por valores singulares. O tratamento semântico através deste método algébrico otimizado evidencia características e viabiliza a redução dimensional com a menor perda possível.

Formalmente, dada uma matriz $A \in \mathbb{R}^{m \times n}$, existem duas matrizes ortogonais $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ e uma matriz $\Sigma \in \mathbb{R}^{m \times n}$ que possui apenas a diagonal principal possivelmente não-nula composta de números não negativos denominados **valores singulares**, σ_i, de A, como mostra (3.1).



(3.1)

As colunas de U, **vetores singulares**, u_i de A são também autovetores de AA^T, enquanto as colunas de V são autovetores de A^TA. Na matriz Σ os valores singulares estão dispostos em ordem decrescente ($\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$) e são raízes quadradas dos autovalores não nulos de ambas AA^T e A^TA. Para matrizes positivas definidas $U\Sigma V^T$ é idêntica a $Q\Lambda Q^T$ No caso complexo, Σ permanece real mas U e V se tornam unitárias e deve ser tomado o conjugado complexo em $U\Sigma V^H$.

Nas matrizes U e V, a SVD fornece bases ortogonais para os quatro sub-espaços fundamentais, tomando os r valores singulares não nulos, como mostrado na tabela 3.1. Observando $AV = U\Sigma$ uma coluna por vez temos que a multiplicação de A por uma coluna v_j de V produz σ_i vezes a coluna u_i de U.

primeiras	r	colunas de U	→	espaço coluna de A
últimas	m-r	colunas de U	→	espaço nulo a esquerda de A
primeiras	r	colunas de V	→	espaço linha de A
últimas	m-r	colunas de V	→	espaço nulo de A

Tabela 3.1: SVD e os quatro sub-espaços fundamentais

A estabilidade numérica computacional desta decomposição é ponto chave que vale destaque. Multiplicações das matrizes ortogonais U e V por vetor não alteram o comprimento do produto ($||Ux||^2 = x^TU^TUx = ||x||^2$) e por isso não destroem a escala. Apesar de operações com valores singulares extremos potencialmente causarem transbordamento, estes valores ainda são os melhores possíveis e sua razão máxima ($\sigma_{max}/\sigma_{min}$) revela o número de condição (κ) de uma matriz inversível.

3.2 Aproximações de Posto Reduzido

Supondo uma matriz termo-documento A, grande e de posto relativamente pequeno. Decomposta, temos tipicamente um decaimento acentuado na magnitude dos valores singulares. Inspecionando este conjunto é possível estimar um posto reduzido de forma a aproximar a matriz original por outra mais compacta. Um número escolhido destes valores maiores é geralmente considerado o posto numérico da matriz. Além de compactar informação, o truncamento da SVD ainda reduz simultaneamente o ruído original presente e representa uma maneira eficiente de reduzir o posto.

Formalmente, e segundo Eckart e Young em [4] se $A \in \mathbb{R}^{m \times n}$ possui r valores singulares não-nulos e ordenados decrescentemente como $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r$, então para cada $k < r$, a matriz de posto k que **melhor** aproxima A é dada por:

$$A_k := \sum_{i=1}^{k} \sigma_i u_i v_i^T \quad \approx \quad \sum_{i=1}^{n} \sigma_i u_i v_i^T = A \tag{3.2}$$

As aproximações matriciais são avaliadas geralmente através de normas. Estas, capturam noções essenciais de distância em espaço vetorial, medindo a convergência numérica através de um valor real sujeito aos axiomas de positividade, desigualdade do triângulo e homogeneidade. Normas vetoriais, como a Frobenius, tomam uma matriz $m \times n$ como um vetor em dimensão mn, conforme mostrado em (3.3).

$$||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} \tag{3.3}$$

A distância da matriz original A à matriz mais próxima de posto k pela norma Frobenius é dada pela norma quadrática dos valores singulares descartados, conforme (3.4), não sendo muito prática em casos reais com matrizes grandes.

$$\min_{posto(B)=k} ||A - B||_F = ||A - A_k||_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_{rA}^2} \tag{3.4}$$

Normas induzidas, por sua vez, definidas em termos do comportamento da matriz avaliada como transformação entre um espaço normado (domínio) e outro (imagem), são mais úteis. O caso particular da norma-2 induzida é exposto em (3.5).

$$||A||_2 = \max_{x \neq 0} \frac{||Ax||}{||x||} = \sqrt{\lambda_{max}(A^* A)} = \sigma_{max}(A) \tag{3.5}$$

Esta norma afere a distância conforme mostrado em (3.6), tomando o primeiro (maior) valor singular descartado na aproximação. Tratamento mais detalhado a esse respeito é formalizado por Eldén em [5].

$$\min_{posto(B)=k} ||A - B||_2 = ||A - A_k||_2 = \left|\left| \sum_{i=k+1}^{r} \sigma_i u_i v_i^T \right|\right| = \sigma_{k+1} \tag{3.6}$$

3.3 Algoritmos e Aplicabilidade

Procedimentos para decomposição em valores singulares podem ser analíticos ou numéricos. Manualmente e em dimensões reduzidas o cálculo analítico requer a multiplicação a da matriz A a ser decomposta formando AA^T e A^TA. Na sequência são encontrados os autovalores e autovetores para cada caso. Os valores singulares são as raízes quadradas dos autovalores não nulos e os vetores singulares são os próprios autovetores encontrados. Claramente, este manuseio é cansativo e propenso a erro, não escalando para dimensões maiores.

Numericamente existem duas abordagens interativas distintas para calcular a SVD. Para matrizes densas (poucos valores nulos) a decomposição é computada tipicamente em dois estágios. Primeiro a matriz é reduzida a forma bidiagonal através de transformações "*householder*"[2] ao custo $O(mn^2)$. Depois os autovalores e autovetores são calculados por uma variante do algoritmo QR custando $O(n)$ iterações. O pacote LAPACK [3] e o software MATLAB [4] implementam essa rotina.

Matrizes esparças (muitos valores nulos) merecem considerações especiais no cálculo da SVD uma vez que as transformações compactas citadas destruiriam completamente a esparsidade aumentando drasticamente o requisito de espaço e complexidade computacional. O método de Arnoldi e Lanczos é uma alternativa robusta e atrativa para a decomposição em valores singulares uma vez que toma a matriz como operador em cálculos matriz-vetor otimizados. O pacote ARPACK[5] e derivados além do MATLAB, implementam essa rotina, escolha deste trabalho.

Devido ao alto valor agregado no tratamento algébrico de propriedades genéricas de sistemas lineares e também pela flexibilidade, a SVD é utilizada com vários objetivos em aplicações distintas que merecem destaque. Na **indexação semântica latente**, que matematicamente se traduz em decompor uma matriz termo-documento mapeada a partir de uma coleção utilizando a decomposição em valores singulares truncada. O resultado (U, Σ e V) é utilizado na mineração eficiente (dimensão reduzida) e seletiva (filtrada) dos dados.

Outras aplicações como a solução de sistemas lineares pelo **cálculo de matrizes inversas** utilizam com frequencia algoritmos numéricos baseados em SVD ao invés de procedimentos analíticos. Ainda, na **análise em componentes principais** ou PCA[6] dados de séries temporais, provenientes de processos industriais, podem ser estruturados em forma de vetores (coluna) numa matriz, posteriormente decomposta visando diferenciar amostras e salientar as características mais relevantes aferidas.

[2] en.wikipedia.org/wiki/Householder_reflection
[3] Linear Algebra PACKage, biblioteca de código para álgebra linear numérica.
[4] http://www.mathworks.com/
[5] ARnoldi PACKage, biblioteca de código para decomposição em valores singulares esparsa.
[6] *Principal Component Analysis*

Exemplo Simplificado

Aproximações de baixa ordem foram consideradas no exemplo simplificado apresentado. Naquele ponto, a redução dimensional pela simples exclusão de eixos (termos) diminuiu o requisito computacional mas trouxe graves problemas de redução semântica e perda de indexação em alguns documentos.

Como mostrado, a SVD oferece aprimoramento significativo definindo uma nova base abstrata de resíduo mínimo e ortogonal (otimizada para cálculo de relevância) para mineração de dados. Neste novo cenário, a redução dimensional criteriosa traduzida no descarte de eixos menos significativos vinculados aos menores autovalores, oferece tratamento algébrico eficiente, reduz carga computacional e ruído. A equação (3.7) expõe o resultado númerico calculado via Matlab pelo código anexo.

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}}_{A} \approx \underbrace{\begin{bmatrix} -0.58 & 0.02 \\ -0.58 & 0.70 \\ -0.58 & -0.72 \end{bmatrix}}_{U_k} \underbrace{\begin{bmatrix} 2.66 & 0 \\ 0 & 1.00 \end{bmatrix}}_{\Sigma_k} \underbrace{\begin{bmatrix} -0.44 & -0.44 & -0.44 & -0.65 \\ 0.72 & -0.02 & -0.70 & -0.00 \end{bmatrix}}_{V_k^T} \tag{3.7}$$

Graficamente, o exemplo simplificado foi reduzido do espaço tridimensional para o plano bidimensional otimizado. Na figura 3.1 é possível notar a distinção entre os vetores-documento no novo plano e a viabilidade da redução dimensional.

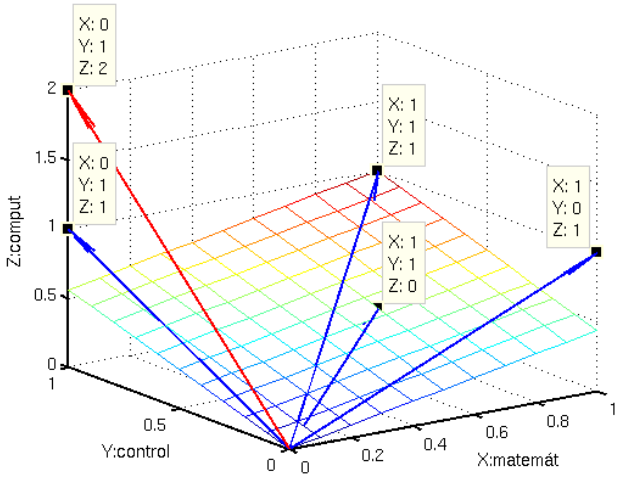


Figura 3.1: Exemplo otimizado pela decomposição em valores singulares.

A norma Frobenius dada por (3.4) é usada para quantificar a informação perdida na redução dimensional através da diferença entre a norma da matriz original e após a decomposição. Neste exemplo, a redução dimensional de 33% ocasionou perda de apenas 6% da informação original, segundo esta métrica.

3.4 Matrizes Esparsas

Uma matriz é dita **densa** quando possui valores diferentes de zero atribuídos a grande maioria dos seus elementos. Por outro lado, uma matriz é **esparsa** quando possui uma grande quantidade relativa de elementos nulos, ou ausentes. Esta abstração têm aplicações em problemas de engenharia, física, computação além de outras áreas. A matriz **termo-documento** modelada e construída neste trabalho possui característica significativamente esparsa, que merece atenção e tratamento diferenciado tanto no armazenamento quanto no processamento.

Além dos parâmetros convencionais para classificação de uma matriz como o número de linhas, número de colunas, estrutura e posto, é comum observar a densidade para tratar matrizes esparsas. Analogamente ao conceito físico, a densidade matricial é definida como a razão entre a quantidade de elementos não nulos e o tamanho total da matriz, como mostrado na equação (3.8). Este valor real é importante para caracterizar a quantidade de informação presente, bem como estimar a carga de processamento e espaço em memória necessários para tratar tais objetos computacionalmente.

$$\text{densidade} = \frac{\text{número de elementos diferentes de zero}}{\text{número de linas} \times \text{número de colunas}} \tag{3.8}$$

3.4.1 Armazenamento Otimizado

Diferente das matrizes densas, o armazenamento de grandes matrizes esparsas em memória requer tratamento particular. Para matrizes densas e não estruturadas inexistem alternativas significativamente vantajosas na compactação do espaço utilizado uma vez que cada elemento é armazenado indistintamente. Neste caso, armazenar uma matriz $A^{m \times n}$ consome memória proporcional a $(m)(n)$(tamanho do elemento) aproximadamente.

Em outro cenário, operando matrizes esparsas, muitos elementos são nulos e não tem necessidade explícita de serem armazenados. Persistir tais objetos de forma eficiente requer considerar também da densidade da matriz, sendo espaço dimensionado para a mesma matriz A segundo $(m)(n)(\textit{tamanho do elemento})(\textit{densidade})$. Estratégias comuns para armazenar matrizes esparças envolvem uso de listas vinculadas ou tabelas *hash* especiais. Frequentemente em mineração de dados as matrizes apresentam densidade inferir a 5%, o que resulta em ganho expressivo de desempenho e eficiência quando tratadas de forma especial. Uma abordagem diferenciada envolve aspectos particulares de memória permanente ou de trabalho (volátil).

Memória Permanente

Uma das principais características da memória permanente é o acesso esporádico e frequentemente sequencial do conjunto completo de dados. Memória permanente geralmente se traduz em discos rígidos acessados durante o início da aplicação ou em intervalos conhecidos espaçados de tempo. Neste conceito é fundamental atingir a maior compacidade possível, mesmo em sacrifício de parte do processamento, dado que neste ponto o espaço é fator restritivo da capacidade operacional. Visando otimizar essa abordagem algumas técnicas desenvolvidas operam somente sobre a massa útil de maneira diferenciada.

Para armazenar apenas os elementos não nulos de uma matriz de forma simples, seria necessário conhecer apenas seus índices (linha e coluna) além do seu valor. Nessa estrutura uma matriz esparsa poderia ser mapeada numa lista de tríades gerando uma indexação **linha-coluna** mostrada em (3.9). Intuitivamente simples e genérica esta estrutura permite particionamento com facilidade e oferece vantagens para tamanho e esparsidade elevados.

$$\begin{pmatrix} 0 & 0,256 & 0 \\ 0 & 0 & 0,48 \\ 0,6 & 0 & 0 \end{pmatrix} \quad \rightarrow \quad \begin{array}{l} (1,2):\ 0,256 \\ (2,3):\ 0,48 \\ (3,1):\ 0,6 \end{array} \tag{3.9}$$

Visando otimizar o modelo baseado em linha-coluna, na implementação é possível adotar tamanho fixo para os campos de índice (linha e coluna) e também de valor. O armazenamento de **largura fixa** sequencial otimiza espaço ao dispensar cabeçalhos e marcações especiais em cada elemento. Na demonstração (3.10) foram adotados índices com largura dois e largura três para o valor. Entre as desvantagens este método apresenta dificuldade para acesso aleatório aos elementos além de pouca robustez na leitura e durante o particionamento.

$$\begin{pmatrix} 0 & 0,256 & 0 \\ 0 & 0 & 0,48 \\ 0,6 & 0 & 0 \end{pmatrix} \quad \rightarrow \quad \overbrace{1\square 2\square 0,256}\,\overbrace{2\square 3\square 0,48\square}\,\overbrace{3\square 1\square 0,6\square\square} \tag{3.10}$$

Evolutivamente métodos genéricos baseados na compressão de linhas (CRS[7]) e colunas (CCS[8]) foram criados sem restrições estruturais armazenando apenas elementos necessários. O formato CCS, também conhecido como **Harwell-Boeing**, armazena valores não-nulos das colunas continuamente em um vetor dedicado utilizando outros dois vetores como apontadores. Analogamente o formato CRS realiza

[7] *Compressed Row Storage* (http://netlib.org/linalg/html_templates/node91.html)
[8] *Compressed Column Storage* (http://netlib.org/linalg/html_templates/node92.html)

compressão por linhas e equivale ao CCS da matriz transposta. Pela alta eficiência mostrada em (3.11) e tendo desvantagem apenas no requisito de endereçamento indireto, o formato de Harwell-Boeing é amplamente utilizada, inclusive neste trabalho.

$$\begin{pmatrix} 0 & 0,256 & 0 \\ 0 & 0 & 0,48 \\ 0,6 & 0 & 0 \end{pmatrix} \quad \begin{array}{lllll} & 1 & 2 & 3 & 4 \\ \rightarrow & 3 & 1 & 2 & \\ & 0{,}6 & 0{,}256 & 0{,}48 & \end{array} \quad \begin{array}{ll} \rightarrow & \text{col\_ptr} \\ \rightarrow & \text{row\_ind} \\ \rightarrow & \text{value} \end{array} \tag{3.11}$$

Memória de Trabalho

O acesso aleatório e frequente em porções menores de dados são as principais características desta memória, geralmente alocada na RAM do computador. Por razões algébricas é interessante dispor de formas práticas para acesso independente a colunas e linhas específicas da matriz operada. Neste conceito é fundamental alcançar a maior velocidade possível, sem comprometer significativamente o processamento ocupado nos cálculos inerentes. Neste ambiente mais volátil, técnicas semelhantes ao caso permanente são utilizadas com devidas ponderações.

Aplicações mais antigas e baseadas em linguagens procedurais habitualmente implementam a estratégia de linha-coluna pela simplicidade ou vetores para Harwell-Boeing quando o desempenho é crítico. O acesso indireto é realizado através de procedimentos otimizados para cada arquitetura. Modelos mais modernos e ainda em estudos, como o projeto COLT[9], utilizam tabelas *hash* e métodos de alto nível que encapsulam as implementações tendendo a ser mais práticos e úteis futuramente.

3.4.2 Algoritmo de Arnoldi-Lanczos

Algoritmos para realizar SVD em matrizes densas são aplicáveis a matrizes menores, sendo o tamanho limite pautado pela potência computacional disponível. Porém, frequentemente em mineração de dados as matrizes são muito grandes e/ou esparsas, apesar de suas decomposições manterem características densas. Com requisitos de espaço e processamento proibitivos neste contexto, a utilização de métodos convencionais não é razoável.

Alternativamente, métodos que utilizam a matriz como operador caixa preta, a exemplo do método de Arnoldi-Lanczos, ao invés de modificar sua estrutura são preferíveis e escalam bem. Trabalhos anteriores de Golub em [7], Eldén em [5] e Júnior em [9] detalham bem o conceito e a implementação desta estratégia, também utilizada neste trabalho, servindo de base para seguinte abordagem sucinta.

[9] http://acs.lbl.gov/software/colt/

Método de Arnoldi

Tomando a matriz $A \in \mathbb{R}^{n\times n}$ grande, esparsa e não-simétrica usaremos uma tranformação de similaridade ortogonal para a forma de Hessenberg superior ($V^TAV = H$) visando calcular a decomposição de Schur ($A = URU^T$, U ortogonal e R triangular superior). Apesar de transformações *householder* atenderem este objetivo apresentam efeito colateral do preenchimento de elementos nulos da matriz esparsa. Por isso, alternativamente esse cálculo é computado recursivamente através de operações matriz-vetor como mostrado na tabela (3.2).

1.		Escolher v_1 tal que $\|\|v_1\|\|_2 = 1$
2.		**Para** $j = 1, 2, ...$
	(a)	$h_{ij} = v_i^T A v_j$, com $i = 1, 2, ..., j$
	(b)	$v = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
	(c)	$h_{j+1,j} = \|\|v\|\|_2$
	(d)	$v_{j+1} = (1/h_{j+1,j})v$
3.		**Fim-Para**

Tabela 3.2: Algoritmo de Arnoldi.

Neste método, segundo Eldén em [5], cada recursão implica em uma operação matriz-vetor retendo toda informação produzida ao longo do processo. Assumindo que V_k é uma aproximação razoável do autoespaço, podemos tomar $H_k = Z_k \hat{R} Z_k^T$ como a decomposição de Schur de H_k e partindo de $AV_k \approx V_k H_k$ temos $A\hat{U} \approx \hat{U}_k \hat{R}_k$ e $\hat{U}_k = V_k Z_k$. Os autovalores de $\hat{R}_k$ são, assim, aproximações dos autovalores de A.

O procedimento exposto apresenta problemas de perda da ortogonalidade exata nas operações de ponto flutuante e também aumento do processamento e espaço requerido nas iterações. Estes efeitos são contornados eficientemente pela reortogonalização e reinício implícito abordados oportunamente.

Método de Lanczos

Ao aplicar o método de Arnoldi a uma matriz A simétrica, a forma de Hessenberg superior se torna tridiagonal, conforme mostrado em [7] por Golub. Neste contexto, uma versão simétrica, mais econômica do algoritmo pode ser derivada, partindo de uma ortogonalização tridiagonal, como apresentado pela equação (3.12).

$$AV = (Av_1 Av_2 ... Av_n) = VT = (v_1 v_2 ... v_n) \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix} \quad (3.12)$$

Reorganizando para obter: $\beta_j v_{j+1} = Av_j - \alpha_j v_j - \beta_{j-1} v_{j-1}$ é possível usar a equação (3.12) recursivamente, com α_j e β_j determinados pelos vetores ortogonais normalizados. De acordo com Eldén em [5], a decomposição Lanczos definida por $AV_k = V_k T_k + \beta_k v_{k+1} e_k^T$ é gerada pela sequencia apresentada na tabela 3.3.

1.		Fazer $\beta_0 = 0$ e $v_0 = 0$ escolhendo v_1 tal que $\|\|v_1\|\|_2 = 1$
2.		**Para** $j = 1, 2, ...$
	(a)	$\alpha_j = v_j^T A v_j$
	(b)	$v = Av_j - \alpha_j v_j - \beta_{j-1} v_{j-1}$
	(c)	$\beta_j = \|\|v_j\|\|_2$
	(d)	$v_{j+1} = (1/\beta_j) v$
3.		**Fim-Para**

Tabela 3.3: Algoritmo de Lanczos.

Nota-se que em cada iteração apenas uma multiplicação matriz-vetor (de maior custo) é realizada sendo este um ponto forte do método de Arnoldi mantido em Lanczos. O procedimento exposto também apresenta problemas de perda da ortogonalidade exata nas operações de ponto flutuante e aumento do processamento e espaço requerido nas iterações, com solução também similar ao método de Arnoldi.

Reortogonalização

Este processo visa reparar a ortogonalização do vetores explicitamente a cada iteração ou quando não-ortogonalidades são detectadas por algum critério. Dentre as alternativas existentes para solução, está o processo de Gram-Schmidt.

Reinício Implícito

De acordo com Eldén em [5], sem esta estratégia, uma sobrecarga de processamento ou esgotamento de memória pode ocorrer antes que uma boa aproximação seja computada. Esse método foi desenvolvido para contornar esse problema através da redução dimensional durante uma decomposição Arnoldi-Lanczos pelo descarte de autovalores desnecessários, reiniciando implicitamente o algoritmo.

Convergência Rápida

O método otimizado apresentado para matrizes esparsas é mais complexo que seu análogo denso, sendo interessante quando apenas poucas tríades são requisitados em grandes matrizes. Neste caso, a convergência de algoritmos iterativos é medida pelo alcance de um limiar de precisão configurado. Trabalhos anteriores de Golub em [7] e Eldén em [5] apontam o rápido evidenciamento (convergência) para valores singulares extremos, detectado também em testes preliminares neste trabalho.

3.5 Software Disponível

3.5.1 Fortran

Criada na década de 1950 pela IBM e bem estabelecida atualmente, a liguagem procedural Fortran[10] tem boa reputação em algoritmos numéricos e conta com vasta biblioteca de funções. Os pacotes BLAS, LAPACK e ARPACK[11] são exemplos famosos e comumente utilizados em álgebra linear. Evolutivamente a linguagem foi aprimorada por várias versões com adição de funcionalidades e o pacote ARPACK de interesse neste projeto está implementado na versão Fortran-77 de 1977.

Sendo uma linguagem compilada e focada em algoritmos numéricos, o Fortran alcança níveis de desempenho superiores a qualquer outra solução. O uso de compiladores especiais, como o iFort[12], juntamente com bibliotecas dedicadas e altamente otimizadas como a MKL[13] proporcionam à aplicação o melhor aproveitamento de cada ciclo podendo ainda serem paralelizados através de rotinas em MPI[14].

A implementação disponível no pacote ARPACK foi testada neste trabalho apresentando elevado desempenho, como esperado. Pelo fato de ser baseada em uma versão antiga da linguagem Fortran, que não permitia carga dinâmica dos parâmetros, a recompilação completa do software era exigida cada vez que a massa de dados fosse modificada. A ausência de orientação a objetos e a exposição de funções otimizadas de baixo nível tornaram o código pouco flexível para outras funcionalidades.

Nos experimentos realizados, a alimentação da aplicação era feita através de arquivos pré-condicionados e a saída exposta em arquivos de texto formatados. Pelo foco em algoritmos numéricos e impossibilidade de atualização da versão devido ao legado de compatibilidade com o ARPACK, essa alternativa mostrou deficiência em lidar com outros dispositivos de entrada e saída mais genéricos, como os bancos de dados, que são fonte essencial da informação tratada.

A execução periódica do algoritmo implementado é primordial para atualização da base e mineração de novos dados. Usualmente funcionalidades temporais são desenvolvidas baseadas em serviços (*daemons*) executados no sistema operacional hospedeiro. Aplicativos com estas características não são suportados pela versão utilizada da linguagem Fortran o que exigiria controle cronometrado externo.

Considerando o desempenho atingido frente as dificuldades relacionadas a falta de flexibidade e clareza do código, utilização de entrada/saída pouco adequada, necessidade de recompilação frequente, controle temporal externo e manutenção trabalhosa foi tomada a decisão de avaliar outras alternativas mais completas.

[10] FORmula TRANslation
[11] *ARnoldi Package*, versão esparsa do pacote LAPACK
[12] Intel Fortran Compiler
[13] *Intel Math Kernel Library*
[14] *Message Passing Interface*

3.5.2 Java

A plataforma Java está entre as opções mais modernas diponíveis atualmente. Diferentemente das linguagens convencionais, compiladas para código nativo, a linguagem Java é compilada para um *bytecode* executado por uma máquina virtual (JRE[15]) . Oferece desenvolvimento multi-plataforma em linguagem orientada a objetos, simples e clara. Possui funcionalidade multi-tarefa (*multi-thread*) nativa e uma gama de ambientes que a tornam alternativa atrativa tanto para servidores de alto desempenho quando para aplicações móveis.

Projetada pela SUN Microsystems na década de 1990 a linguagem Java se estruturou em uma realidade conectada mantendo grande força no desenvolvimento web. Métodos otimizados, como o RMI[16] , para comunicação entre aplicativos em máquinas diferentes e distribuição de tarefas em rede fazem parte da API[17] básica. Para os algoritmos numéricos estudados neste trabalho, a praticidade na distribuição/paralelização aliada a flexibilidade na conexão direta ou gerenciada aos sistemas de bancos facilitam a obtenção e tratamento da informação minerada.

Na evolução da plataforma as JREs tiveram sua estabilidade e desempenho aprimorados atingindo patamares aceitáveis para execução de aplicações diversas. O consumo excessivo de memória, inicialmente detectado, foi superado e o baixo custo e alta disponibilidade deste recurso estimularam o desenvolvimento de projetos numéricos interessantes, como o IBM Ninja[18], que visa tornar a plataforma Java competitiva ao Fortran e C++ no domínio da computação científica.

Qualitativamente as vantagens abordadas até agora não comprometeram a facilidade de manutenção original desta plataforma. Diretivas e testes oriundos do compilador colaboram na detecção de erros durante a implementação e tornam o desenvolvimento mais gerenciável. Somado a isso a possibilidade de realizar teste unitário em porções menores do software viabilizadas por ferramentas automáticas diminuem o retrabalho e geram um produto final mais consistente.

Computacionalmente, bibliotecas de software numérico foram desenvolvidas ou traduzidas de versões anteriores construindo/portando funcionalidades para este ambiente. De forma sucinta, as alternativas mais populares serão abordadas com foco nas funcionalidades oferecidas e desempenho alcançado.

- **JARPACK:** Parte integrante do projeto "netlib-java"de domínio público esta biblioteca é uma tradução direta do Fortran realizada através da ferramenta F2J[19] ainda em estágio de desenvolvimento. Este pacote oferece uma camada

[15] *Java Runtime Environment*
[16] *Remote Procedure Call*
[17] *Application Programming Interface*
[18] http://www.research.ibm.com/ninja/
[19] Fortran to Java

de acesso a APIs tradicionais configuráveis para utilizar implementação nativa e otimizada ou linguagem Java pura e multi-plataforma. O legado de software codificado em Fotran é bastante amplo e as tentativas de modernização tecnológica direta como esta, são interessantes, não fossem a instabilidade e falta de robustez criadas no processo inicial de tradução automática. Dados estes fatos, esta biblioteca não foi considerada adequada para uso neste trabalho frente aos objetivos planejados.

- **JAMA:** Acrônimo de *Java Matrix Package* é um pacote para álgebra linear codificado puramente em Java que fornece funcionalidades para construir e manipular matrizes reais e densas. Essencialmente este projeto visa oferecer interfaces amigáveis e padronizadas e ao mesmo tempo com desempenho satisfatório para manipulações algébricas. Promissoramente apoiado pela empresa Mathworks[20] e também pelo NIST[21], oferece uma especificação bem estruturada mas uma implementação pouco amadurecida e incompleta para os requisitos deste trabalho no tratamento de matrizes esparças.

- **COLT:** Conjunto de bibliotecas de código livre para computação científica de alto desempenho utilizado primariamente no CERN[22]. Oferece implementação enxuta de algoritmos para álgebra linear básica com segregação entre dados e funções. Segundo os desenvolvedores[23], no projeto Ninja da IBM, alcançou 90% do desempenho do equivalente Fortran otimizado, atingindo a marca de 1,9 gigaflop[24]/s em um processador Intel Xeon de 2.8 GHz. Na avaliação realizada este pacote supriria da melhora forma os requisitos deste projeto, não fosse a ausência de métodos para tratar matriz esparças. Apesar de apresentar desempenho muito satisfatório em matrizes densas, a aplicação dos mesmos algoritmos em matrizes grandes e esparças ainda é inviável neste projeto.

Para fins de realizar a decomposição em valores singulares de matrizes grandes e esparças, as bibliotecas e algoritmos disponíveis na plataforma Java e avaliadas neste projeto demonstraram grau de maturidade e robustez inadequados a aplicação pretendida. Apesar de parte dessas soluções apresentarem especificações claras e desenvolvimento ativo os métodos disponíveis ainda focam tratamento de sistemas densos e menores. Estabelecidos estes fatos, o novo foco era estabelecer uma relação de compromisso entre a rapidez encontrada no Fortran e a flexibilidade contectividade da plataforma Java.

[20]Desenvolvedora do software Matlab.
[21]Instituto Nacional de Padrões e Tecnologia do governo norte-americano
[22]Organização Européia Para Pesquisa Nuclear
[23]http://acs.lbl.gov/software/colt/
[24]Bilhões de operações de ponto flutuante.

3.5.3 C (ANSI)

Desenvolvida em 1972 nos Laboratórios Bell, a linguagem procedural C é bem estabelecida e muito ampla atualmente. Criada para operar o hardware em baixo nível e eficiencia se mostrou uma solução muito rápida e compacta. Hoje é utilizada para programação de microcontroladores até grande aplicações críticas. Serve como base para implementação do próprio compilador Fortran e das máquinas Java mais populares. Ao longo de sua evolução passou por aprimoramentos, incremento de funcionalidades e padronização até o estabelecimento do C (ANSI)[25].

Aplicações multi-tarefa são desenvolvidas com uso de bibliotecas específicas em cada sistema operacional e permitem aplicações mais responsíveis e ágeis. A implementação do paralelismo também é viabilizada através do uso de rotinas MPI de código aberto, bem testadas e largamente utilizadas em ambiente científico. Operando em baixo nível com suporte a parte do código em linguagem natural de máquina (*assembly*) apresenta desempenho otimizado muito próximo ao Fortran quando não manipulando números complexos, implementados indiretamente.

Tomando o sistema Unix como primeiro balizador desta linguagem, sua característica genérica e flexível permite implementação tanto como aplicativo isolado ou serviço (*daemon*) executado sengundo diretivas e temporizações do próprio sistema operacional. Esta característica é muito útil nos requisitos deste projeto por viabilizar atualizações periódicas e configuráveis das bases de dados além de prover mecanismo de recuperação após falha durante a execução ou de hardware.

Os compiladores de código C construídos para vários sistemas operacionais tornaram esta linguagem portável para muitos ambientes, desde dispositivos portáteis até arquiteturas super-escalares em servidores. Detalhe importante a ser observado é a necessidade de recompilar o código fonte gerando novo binário nativo para cada arquitetura, diferente da plataforma Java que teoricamente utiliza o mesmo *bytecode* em ambienntes diferentes com a JRE instalada.

Muitos sistemas de gerenciamento de bancos de dados (SGBDs) modernos tem implementação direta ou indireta baseados na linguagem C. Naturalmente a conexão com estas aplicações é facilitada e ocorre por meio de *drivers*[26] fornecidos pelo fabricante. Alternativas de mais alto nível contam com acesso via camadas que unificam e padronizam leituras e escritas imprimindo mais agilidade e segurança. Esta característica oferece grande vantagem frente a restrição do Fortran e viabiliza largo acesso a massa de dados.

[25] American National Standards Institute

[26] Pequenas porções de software que fazem a ponte entre a applicação e o banco de dados.

SVDPACKC

Avaliando alternativas citadas em [1] o pacote SVDPACKC foi testado neste trabalho. Desenvolvido principalmente por Michael W. Berry esta biblioteca contém funções para decomposição em valores singulares de matrizes esparças baseada no método de Arnoldi Lanczos apresentado. Seguindo os algoritmos oriundos do pacote ARPACK, o autor traduziu para linguagem C (ANSI) as funcionalidades necessárias de forma organizada e comentada. Por se tratar de ambiente mais conhecido e bem estruturado a compilação e primeiros testes ocorreram sem dificuldades.

O tratamento da entrada e saída nas funções presentes neste pacote também são triviais e permitiram rápida adaptação para escrita em arquivos de texto convencionais de forma a serem avaliados e aproveitados em outros aplicativos que também interagem com a massa de dados.

O pacote disponibilizado oferece quatro métodos distintos em duas abordagens (cíclica e simétrica) para cálculo da decomposição:

- **las:** Lanczos por vetor simples, cíclico ou simétrico.
- **bls:** Lanczos por blocos, cíclico ou simétrico.
- **sis:** Iteração por subespaço, cíclico ou simétrico.
- **tms:** Minimização de traço, cíclico e simétrico

Internamente, cronômetros implementados servem de referência para aferir o desempenho individual de cada método, que variou segundo matriz utilizada na rotina de testes proposta pelo autor. Para massas de dados genéricas e não estruturadas, a rotina **las** demonstrou melhor desempenho na maior parte das vezes.

Mesclando as vantagens e desvantagens apresentadas em cada abordagem (Fortran, Java e C) com os requisitos principais deste projeto, foi escolhida a linguagem C (ANSI) e integração com a biblioteca SVDPACKC no método **las simétrico** neste trabalho. Neste cenário ficou prezervado bom desempenho na decomposição de grandes massas de dados, boa conectividade com sistemas de bancos de dados, possibilidade de implementação de serviços com controle temporal interno e manutenção flexível e falcilitada da solução desenvolvida.

3.6 Compatibilidade de Hardware

Cada etapa no trabalho de mineração estabelece requisitos de hardware distintos. Numa arquitetura simples, em todas as etapas são executadas no mesmo servidor, este deve atender minimamente todos os requisitos. Estruturas mais complexas e robustas geralmente utilizam máquinas dedicadas em cada caso.

3.6.1 Carga de Trabalho

Essencialmente o processo de extração se caracteriza pelo uso intenso de disco rígido durante a extração inicial e posterior uso moderado a cada atualização incremental. Mantendo acesso constante e confiável ao servidor principal e sem requisição elevada de processamento é razoável favorecer velocidade de armazenamento frente ao processamento poderoso. Para projetos realizados tendo mineração de dados nos requisitos, algumas rotinas internas (*procedures*) ao banco de dados podem substituir completamente o tratamento externo da atualizações.

Processamento e acesso a memória de trabalho (RAM) periódico e alto é a principal característica da carga de trabalho na indexação. Durante este processo, o poder computacional disponível e utilizado plenamente sendo o principal fator determinante do tempo consumido. Pelo fato de ocorrer somente em intervalos configurados, a capacidade de processamento permanece ociosa podendo inclusive ser alocada para outros trabalhos desde que seja estabelecido política de arbitragem segura e coerente com a periodicidade programada.

Consultas apresentam carga de processamento elevadas porém rápidas, ocupando processador apenas pelo breve intervalo de espera do usuário. Neste processo a utilização de disco rígido é praticamente nula. Em contrapartida, a memória de trabalho permanece constantemente alocada, mesmo na ausência de consultas para evitar recarga sob demanda gerando gargalo que tornaria impraticável uso da aplicação. O tratamento de pesquisas requer ambientes com alta disponibilidade dinâmica e baixa latência de rede.

3.6.2 Especificação de Equipamento

As características do equipamento devem considerar fortemente a carga de trabalho.

Potência de processamento em algoritmos matemáticos, como neste trabalho de mineração de dados é, geralmente medida pela quantidade de operações de ponto flutuante realizadas por segundo (FLOPS[27]) pelo processador. Esta capacidade influencia intimamente os tempos de indexação e pesquisa dado que são essencialmente multiplicações matriz-vetor. Computadores responsáveis por este trabalho devem oferecer além de larga disponibilidade um mecanismo de controle concorrente robusto para evitar interrupções no ciclo. Máquinas atuais, comumente utilizadas como servidores, operam aproximadamente um bilhão de FLOPS (GFLOP) sendo eficientes no tratamento de massas de dados relativamente grandes.

O tamanho da memória primária disponível é fator limitante na definição do tamanho máximo da massa de dados minerada. Dado que a matriz termo-documento, na estrutura proposta, deve ser completamente alocada para decom-

[27] *Floating Point Operations Per Second*

posição, seu tamanho total convertido em bytes não pode exceder a disponibilidade sob pena de utilização automática de memória virtual e consequente perda de desempenho. Além comportar a matriz termo-documento deve ser considerado espaço extra nesta memória utilizado pelo prórpio sistema operacional e também como área de trabalho dos procedimentos algébricos executados internamente no software.

O tamanho da base ou méria em cache (espelho) armazenado é definido pela quantidade de memória secundária disponível. Neste aspecto e dado o estágio atual de disponibilidade farta de memória, dificilmente os discos rígidos oferecerão restrição ao tamanho da base de dados pela sua capacidade. Para estes dispositivos, no entanto, é importante considerar a menor velocidade de acesso aos dados que afetam mais pronunciadamente os processos de extração inicial e indexação. Alternativas mais velozes e caras como os discos de estado sólido (SSD[28]) podem ser consideradas visando aumentar o desempenho gargalado neste processo.

3.6.3 Arquitetura Distribuída

Em grandes projetos que exigem alta confiabilidade é conveniente considerar a utilização de arquitetura distribuída para mineração de dados. Nesta estrutura mais complexa as responsabilidades são segregadas e adequadas em máquinas com perfis específicos para execução de determinadas tarefas. A operação adequada deste conjunto depende fortemente das características da rede e dos servidores utilizados.

Para interligar servidores dedicados numa malha de mineração de dados as redes devem apresentar três características fundamentais: alta banda disponível, baixa latência e alta confiabilidade. O primeiro aspecto é fundamental para acelerar a transmissão de informação diminuindo os tempos de atualização e sincronia. A baixa latência é importante para manter o acesso instantâneo e consequentemente os tempos de pesquisa dentro de patamares confortáveis à utilização. Sem alta confiabilidade todo o processo corre risco dado que as operações serializadas implicam indisposição de funcionalidades para uma falha isolada na ausência de redundância.

Cargas de trabalho bem caracterizadas permitem especificação de servidores dedicados e bem dimensionados para a tarefa alvo. Simplificadamente, operações de extração requerem máquinas com baixo processamento e memória primária, porém alta disponibilidade de memória secundária. A indexação por sua vez deve ser apoiada por alto poder de processamento e quantidade farta de memória primária visando extender sua capacidade de tratamento de volume de informação. Finalmente as consultas tem requisito de processamento moderado e alta disponibilidade de memória primária. Alguns requisitos podem ser reavaliados para utilizar de redundância tendo tarefas segregadas com menor granularidade.

[28] *Solid State Drive*

Capítulo 4

Implementação Prática em Software

O projeto e desenvolvimento do software, bem como utilização do hardware na estrutura disponível envolvem detalhes especiais que merecem destaque.

4.1 Legislação Médica

A profissão médica é regulamentada pela constituição federal tendo particularidades tratadas no Código de Ética Médica (CEM). Segundo este documento, é dever do médico elaborar prontuário para cada paciente atendido, no qual deverá constar os dados clínicos para a boa condução do caso, permancendo sob guarda do médico ou da instituição que assiste o paciente.

O Conselho Federal de Medicina (CFM) através da resolução número 1.638/02 define o prontuário como "o documento único constituído por um conjunto de informações, sinais e imagens registradas, geradas a partir de fatos, acontecimentos e situações sobre a saúde do paciente e a assistência a ele prestada, de caráter legal, sigiloso e científico, que possibilita a comunicação entre membros da equipe multiprofissional e a continuidade da assistência prestada ao indivíduo."

Conforme proposto, este trabalho minera informação médica de forma segura, anonimizada e automática. Para efeito de desenvolvimento e teste, o acesso aos documentos clínicos é realizado através de autenticação segura. Apenas informações clínicas presentes são utilizadas, sem recuperar qualquer vínculo identificador preservando anônimidade. A conversão da coleção em objetos matemáticos abstratos e o posterior tratamento é realizado sem interferência humana.

Os requisitos de segurança de um Sistema de Registro Eletrônico em Saúde (S-RES) são fundamentais para garantir a privacidade, confidencialidade e integridade da informação identificada em saúde. Mesmo não sendo proposta deste tra-

balho estabelecer um sistema como o citado, as decisões tomadas no desenvolvimento seguem as diretivas dos Níveis de Garantia de Segurança (NSG2) estabelecidos conforme convênio firmado entre a Sociedade Brasileira de Informática e o CFM.

4.2 Armazenamento

Na atualidade não existe um padrão estabelecido formalmente ou de fato para persistência da informação clínica em sistemas de bancos de dados. Em consequência, é comum encontrar dentro de uma mesma instituição, diversos bancos de dados com estruturas diferentes armazenando as mesmas informações. A replicação da informação dificulta o estabelecimento de uma fonte universal além de favorecer potencialmente o aparecimento de inconsistências temporais e até lógicas.

Na prática, a segregação não projetada dos sistemas de informação leva ao desconhecimento do modelo de dados utilizado e acarreta desequilíbrio no acesso a tabelas devido a baixa normalização global. Neste cenário, o acesso aos dados requer procedimentos complexos, sincronização e compatibilização entre as fontes disponíveis. Como a mineração opera sobre grande volume de dados, utilizados com frequência, é evidente que acessos nesse ambiente pouco estruturado, impactam negativamente no desempenho do sistema vigente. Efeitos colaterais como este comprometem significativamente a viabilidade destes projetos, caso não tratados.

Como alternativa estratégica para reduzir o impacto no acesso frequente à grandes quantidades de informação armazenadas no HIS, neste trabalho foi implementado um sistema de bancos de dados independente em servidor dedicado que permanece espelhado operando conforme mostrado pelo diagrama na figura 4.1. Nesta ilustração, o **banco de dados hospitalar** é espelhado no **banco de dados dedicado** visando **minerar dados**. Após se **autenticar**, o **médico** pode **realizar pesquisas** independente do **banco de dados hospitalar** copiado.

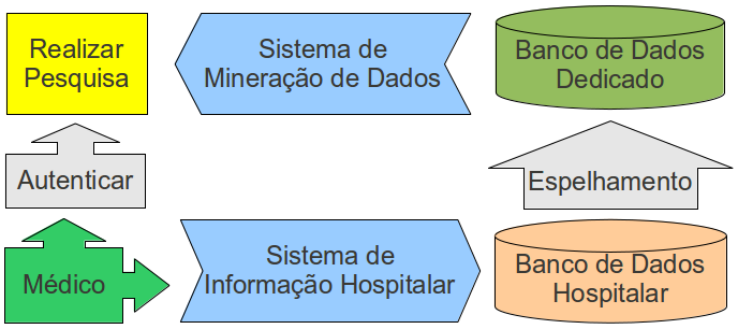


Figura 4.1: Diagrama esquemático para mineração de dados espelhados.

No processo apresentado na figura 4.1 o funcionamento do HIS presente na instituição de saúde permanece inalterado pela utilização do sistema de gestão corrente.

As informações de interesse científico são selecionadas e copiadas (espelhadas) periodicamente em outro sistema de gerência de banco de dados (SGBD) paralelo e independente. De forma desacoplada, é possível minerar os dados espelhados transparentemente e sem gerar inconsistências ou gargalos.

4.2.1 Banco de Dados Caché

O estudo de caso deste trabalho tem como fonte dos dados a serem minerados o principal sistema de bancos de dados mantido no Hospital Universitário Clementino Fraga Filho (HUCFF) da Universidade Federal do Rio de Janeiro (UFRJ). O sistema em uso é denominado Medtrak desenvolvido pela empresa TrakHealth. O SGBD chamado Caché desenvolvido pela empresa InterSystems é utilizado para persistência das informações no banco de dados hospitalar.

Analisar o modelo de dados para descobrir o mapa das informações de interesse é, teoricamente, o primeiro passo para espelhamento em um banco de dados dedicado. A pouca documentação específica, a inacessibilidade do software e a utilização incomum de múltiplas variáveis globais caracteriza o banco de dados do HUCFF como um banco no qual existem dificuldades significativas no acesso a informação.

Como particularidade e desafio adicional, o SGBD Caché não atende ao padrão formal da linguagem SQL para acesso aos dados. Atualmente, o tratamento rotineiro da informação pelo HIS é realizado através de um componente fechado e dedicado para o ambiente Delphi na plataforma Microsoft Windows. Segundo documentação, este componente utiliza internamente a linguagem MUMPS[1] nativa para o Caché.

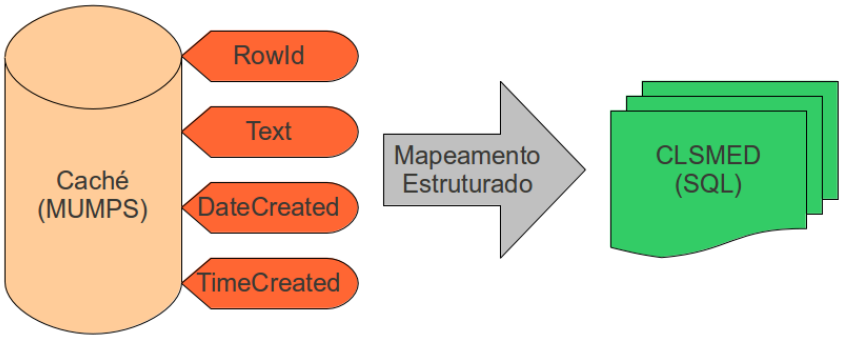


Figura 4.2: Mapeamento estruturado no Caché do hospital universitário.

Como alternativa e tomando base na experiência dos funcionários do departamento de informática, o acesso aos dados foi viabilizado em outra plataforma pela utilização de um mapeamento estruturado (e pouco conhecido) realizado em projeto anterior. Tal mapeamento vincula as variáveis globais do Caché a tabelas relacionais virtuais com informações gerais de laudos emitidos nos setores do hospital

[1]Massachusetts General Hospital Utility Multi-Programming System

que utilizam o sistema citado. Apesar de limitado este acesso atendia aos requisitos mínimos para extração dos dados e superficialmente os laudos apresentavam volume suficiente para consecução do estudo proposto. Em seguida foi realizado espelhamento unilateral (copia apenas com privilégio de leitura) destes dados para carregamento em outro SGBD mais padronizado, rápido e independente do Caché.

4.2.2 Banco de Dados PostgreSQL

Robustez, agilidade, flexibilidade no acesso e disponibilidade são as principais características desejadas em um sistema de banco de dados dedicado. Em avaliação sucinta e considerando a experiência do autor foi escolhido o software de código aberto PostgreSQL[2] como SGBD dedicado para espelhamento. Com experiências de sucesso reconhecidas na utilização deste software, ele apresenta suporte multiplataforma, velocidade e interface compatível com a aplicação.

Conceitualmente o PostgreSQL é um SGBD relacional e estrutura sua persistência em tabelas e linguagem SQL versão 2008. Oferece suporte a transações ACID[3] , controle de concorrência multi-versão MVCC[4] e sofisticado esquema de *backup*. Neste trabalho as experiências realizadas utilizaram principalmente a versão 9.0.4 estável, rodando em servidor independente sobre o sistema operacional Linux.

Como requisito para espelhamento e carga dos dados foi desenvolvido um modelo para o banco de dados espelhados abrigado no SGBD PostgreSQL. Sem exigências complexas este modelo simples contempla poucas tabelas, como pode ser observado na figura 4.3. A tabela **documents**, armazena cópias dos documentos clínicos extraídos e tratados. A tabela **terms** abriga a lista de termos produzida na indexação, com respectiva frequencia na coleção. As tabelas com nome iniciados por **svd** guardam as respectivas matrizes oriundas da decomposição em valores singulares, enquanto a tabela **norms** armazena as normas de cada documento (por linhas) calculada à partir da SVD, como será visto oportunamente. As tabelas **crawler**, **indexer** e **webapp** servem ao propósito de reter um log das operações realizadas e suas características principais.

4.3 Extrator: Crawler

Crawler, robô e *spider* são sinônimos para programas de computador que navegam numa massa de dados de forma metódica e automatizada visando manutenção, validação e obtenção de tipos específicos de informação. Em particular, máquinas de busca na Internet usam crawlers para manter um banco de dados *off-line* atualizado

[2] www.postgresql.org

[3] *Atomicity Consistency Isolation Durability*

[4] *Multiversion Concurrency Control*, atualiza a informação pela criação de versões da mesma

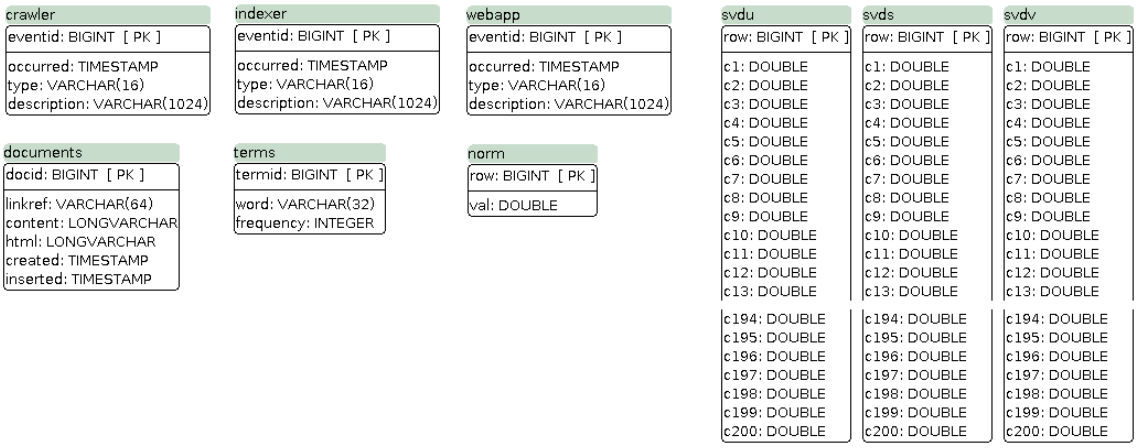


Figura 4.3: Modelo para banco de dados espelhado abrigado no PostgreSQL.

da web visível. Operando segundo regras pré-estabelecidas os crawlers da web visitam uma lista de sites pré-cadastrados coletando informações, mais estruturadas, e adiciondo na lista todos os *links* disponíveis para posterior visita e indexação.

O projeto deste módulo objetivou mapear laudos dos diversos setores do HUCFF contidos no Caché para um conjunto de documentos únicos no PostgreSQL. Como observado na figura 4.4, cada elemento extraído é identificado pela chave primária **docid** e referenciado externamente pelo identificador original do Medtrak na coluna **linkref**. O conteúdo útil (texto para mineração), fica armazenado no campo **content**. Visando aliviar carga de processamento na indexação posterior os textos são tratados e formatados para padronizar a codificação e uso de marcadores especiais presentes de maneira não uniforme nos documentos originais. A coluna **html** armazena uma versão formatada em código HTML[5] do documento original para utilização futura, caso necessário. Os campos **created** e **inserted** armazenam respectivas datas de criação original e cópia dos documentos no banco espelhado.

documents					
docid	**linkref**	**content**	**html**	**created**	**inserted**
1	369152||1||1	Esôfago Trânsito esofagiano livre...	<html> <head> <meta http-equiv="cont...	2000-06-27 10:17:00	2010-06-02 16:03:54.456
...	...	...	...	...	...
212174	5975021||1||1	ESÔFAGO-GASTRO-DUODEN...	<html> <head> <meta http-equiv="cont...	2010-06-02 16:45:38	2010-06-02 17:33:12.135

Figura 4.4: Detalhamento da tabela documents no banco de dados espelhado.

Para realizar essa tarefa as "variáveis globais" de interesse mostradas na figura 4.2, foram identificadas para posterior cópia consistente, sincronizada e estruturada atendendo ao modelo na figura 4.3 do banco de dados dedicado. Detalhes particulares relacionados ao mapa do banco de dados hospitalar e sua conexão de acesso foram omitidos visando preservar a segurança dos dados da instituição.

[5] *HyperText Markup Language*, que significa Linguagem de Marcação de Hipertexto.

Sem vínculo explícito e utilizando apenas referências aos documentos originais dentro do Medtrak, não identificados nos laudos, as informações complementares, eventualmente de interesse clínico, somente poderão ser acessadas buscando pela referência original no software de gestão e sujeito as regras vigentes, já estabelecidas pela instituição. O resultado final do trabalho do crawler é, portanto, uma lista de documentos padronizados e pré-formatados, identificados por referência externa numa longa tabela, mostrada na figura 4.4 e gerenciada pelo PostgreSQL.

4.3.1 Funcionamento

O fluxograma apresentado na figura 4.5 ilustra o funcionamento do crawler dedicado ao HUCFF.

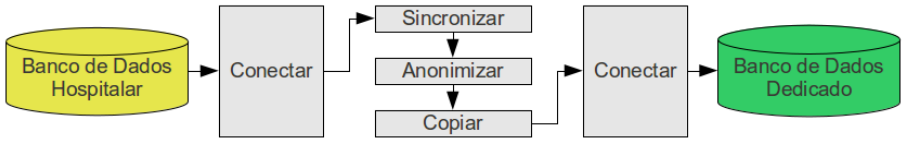


Figura 4.5: Fluxograma simplificado do crawler desenvolvido.

- **Conectar:** O servidor de banco de dados do Medtrak executa um sistema operacional Linux Red Hat em uma máquina com mais de cinco anos de idade. O procedimento de conexão padrão ao Caché não é trivial pois requer o mapeamento interno das variáveis globais em tabelas virtuais. Felizmente a realização anterior deste procedimento viabilizou e poupou esforço extra em lidar com a não padronização deste software. Não foram enfrentados outros problemas no acesso remoto ou instabilidade do SGBD.

- **Sincronizar:** Demarcar os dados a serem copiados tem elevada importância neste projeto. Como a legislação médica vigente proibe a alteração de registros passados, permitindo apenas inserção de novos registros, foi adotada a hipótese de imutabilidade dos dados passados, em curto período. Apoiados nisso torna-se necessário apenas conhecer o momento da última cópia dos dados e refazer este processo daquela data até o presente. Esta estratégia se mostrou leve e suficientemente eficaz para manter os dados sincronizados.

- **Anonimizar:** Não houve necessidade de implementar procedimento de anonimização explícita neste projeto. Na cópia eram tomados três campos de interesse nas variáveis globais do Caché, sendo eles: texto do laudo, data de criação e referência interna. Pelo fato de não mapear nenhuma identificação do paciente os dados já estão automaticamente anonimizados no PostgreSQL uma vez que a referência somente é válida internamente no Caché.

- **Copiar:** Estabelecidos os parâmetros anteriores o espelhamento do banco ocorreu sem problemas na banda de rede disponível. Durante este processo os marcadores especiais, quando encontrados foram filtrados dos textos coletados e a codificação foi padronizada em UTF-8 devido ao legado presente em outros formatos. O crawler apresentou baixa carga de trabalho periódica.

4.3.2 Implementação

O diagrama de caso de uso, padronizado segundo [13], ilustra na figura 4.6 os detalhamentos e dependência de cada processo interno deste módulo, bem como os participantes externos (atores) representados pelos SGBDs. Para **extrair os dados** no **banco de dados dedicado**, as tarefas **conectar**, **sincronizar**, **anonimizar** e **copiar** ocorrem em série espelhando o **banco de dados hospitalar**.

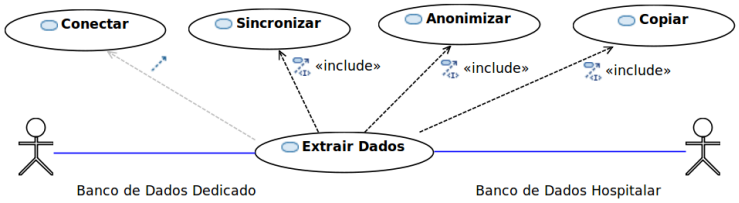


Figura 4.6: Caso de uso detalhado do crawler desenvolvido.

A documentação disponível para o Caché aborda procedimentos para conexão utilizando a linguagem C++ e também Java. A primeira alternativa requer o uso de bibliotecas proprietárias do desenvolvedor além do driver típico. Apesar de oferecer a vantagem do mapeamento, teoricamente transparente, entre variáveis globais internas ao banco e objetos instanciados na aplicação esta solução não funcionou na versão do SGBD disponível, além de apresentar muita complexidade.

Utilizando linguagem Java foi possível acessar as tabelas virtuais do Caché. Sem blibliotecas adicionais e utilizando apenas o driver padrão uma conexão JDBC[6] atendeu os requisitos deste projeto em particular. Mesmo sem vantagens do mapeamento transparente de objetos no banco, a implementação do crawler na plataforma Java se mostrou mais promissora pela simplicidade e característica multi-plataforma.

A implementação deste módulo tem como requisitos básicos a máquina virtual Java na versão 1.6 e o driver do Caché fornecido pelo desenvolvedor na instalação original do software. Na construção da aplicação este último requisito foi empacotado no arquivo binário gerado (jar) para facilitar utilização em outros servidores.

Os primeiros testes reais deste aplicativo expuseram o servidor do Caché a sobrecarga durante comparação de datas e horas em campos segregados. Neste processo

[6] *Java Database Connectivity* é uma interface Java padronizada para acesso a bancos de dados.

era realizada a mescla destes campos em variáveis de texto e posterior comparação interna no SGBD. Na ausência de alternativa nativa mais eficiente foi adotada outra estratégia para sincronização e cópia dos registros. Os intervalos passaram a ser avaliados em dias cheios, de forma a utilizar apenas o campo das datas. Apenas nas extremidades dos intervalos o campo de horas era calculado para alcançar a precisão requirida pela janela de tempo estabelecida na sincronização. Nesta abordagem as operações mais custosas foram reduzidas a apenas duas execuções no início e no final do intervalo, aliviando consideravelmente o servidor e tornando o *crawling* viável.

O computador rodando o crawler apresentou carga de processamento baixa e uso intenso de disco rígido somente na primeira sincronização, como era esperado. Operações de comparação e formatação simples não oneraram processamento enquanto a cópia de documentos requeriu armazenamento proporcional ao volume transferido. Decorrida a primeira e mais pesada sincronia, as subsequêntes foram programadas para ocorrer a cada dez minutos sem problemas na execução.

Estranhamente, durante a cópia de alguns registros, inconformidades no banco, incapazes de serem tratadas no processo de formatação abortaram o aplicativo durante a leitura, que somente reiniciava decorridos dez minutos a partir do último registro copiado. Esta característica indesejável e de causa ainda desconhecida também afeta o Medtrak, sendo desconsiderada neste projeto.

4.4 Indexador: Indexer

Indexadores, de forma geral, visam criar índices numa estrutura projetada para recuperar informação com agilidade, eficiência e precisão. Neste trabalho, em particular, foi implementada a **indexação semântica latente** para minerar uma coleção de documentos clínicos oriundos do HUCFF. Tratar estes dados implica um mapeamento para objetos matemáticos (matrizes e vetores) e sua posterior decomposição utilizando mecânica semelhante para recuperar informação.

Em teoria, indexadores são capazes de operar sobre qualquer coleção natural de documentos, sejam textos, imagens ou vídeos. No entanto, na prática, é necessário e comum realizar um pré-processamento para condicionar a entrada. Como apresentado neste trabalho, estes requisitos são preparados antecipadamente pelo crawler ao gerar um base padronizada e espelhada da coleção original.

O uso independente de cada componente é uma das vantagens do projeto modularizado. O **crawler** e o **indexer** permanecem vinculados e protegidos através SGBD na manutenção do paralelismo e consistência entre as operações. Como visto, as tarefas menores e mais frequentes do primeiro não afetam a sobrecarga de processamento e memória periódicos do segundo, devido aos mecanismos de concorrência nativos deste sistema. Ainda, uma estrutura modular permite segregação de funções

em máquinas dedicadas de perfil mais adequado conectadas via rede.

Para o indexador gerar a matriz termo-documento a partir da coleção, realizar sua decomposição e armazenar o resultado, algumas adaptações computacionais são necessárias. Devido a implementação baseada em tabelas *hash* verticais, os SGBDs atuais demonstram elevada eficiencia em armazenar grande número de linhas nas tabelas, administrando sua variação com robustez. Por outro lado, manter tabelas com muitas colunas e alterar sua quantidade é frequente mais custoso e frágil.

Neste sentido, a decomposição mostrada na equação (3.1), com vetores-documento como colunas da matriz A (conteúdo semântico no espaço coluna) exige elevado esforço do SGBD, limitando fortemente o tamanho da coleção pelo número variável de colunas das tabelas. Como solução, neste trabalho foi implementado a versão transposta da SVD, referida por Berry em [1] e exposta na equação (4.1), mantendo inalteradas as características matemáticas relevantes à mineração.

$$\underbrace{\begin{bmatrix} * & * & \dots & * & * \\ * & \ddots & \dots & * & * \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ * & * & \dots & \ddots & * \\ * & * & \dots & * & * \end{bmatrix}}_{A_{m,n}} \approx \underbrace{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \vdots & \vdots & \vdots \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}}_{U_{m,k}} \underbrace{\begin{bmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \end{bmatrix}}_{\Sigma_{k,k}} \left(\underbrace{\begin{bmatrix} \diamond & \diamond & \diamond \\ \diamond & \diamond & \diamond \\ \vdots & \vdots & \vdots \\ \diamond & \diamond & \diamond \end{bmatrix}}_{V_{n,k}} \right)^T \tag{4.1}$$

Ainda na equação (4.1), a persistência da matriz V na tabela do SGBD é favorecida na forma normal, não transposta. Pelos motivos mencionados, foi fixado, também no caso desta matriz, o número de colunas ao truncamento adotado, permitindo escalonamento flexível dos termos em linhas para viabilizar a operação. Como consequência, cada carga desta matriz implica também sua transposição, sem impacto significativo.

4.4.1 Funcionamento

O indexador, ilustrado no fluxograma da figura 4.7, usa como entrada o banco de dados espelhado e atualizado pelo crawler. A cor vermelha marca funcionalidades importadas por completo enquanto o amarelo traduz recurso externo utilizado.

O funcionamento deste módulo foi segregado em funções internas serializadas. A retirada de *stop-words*, foi integralmente implementada neste projeto, usando apenas a lista de palavras fornecida no projeto Snowball, citado na seção 2.3.2. Com a mesma origem, o método de *stemming*, foi importado com sua interface estudada e adaptada as necessidades vigentes. O algoritmo para decomposição de matrizes foi utilizado a partir de implementação pronta e disponível no SVDPACKC, mencionado na seção 3.5.3. Cada parte foi exaustivamente testada antes de sua integração.

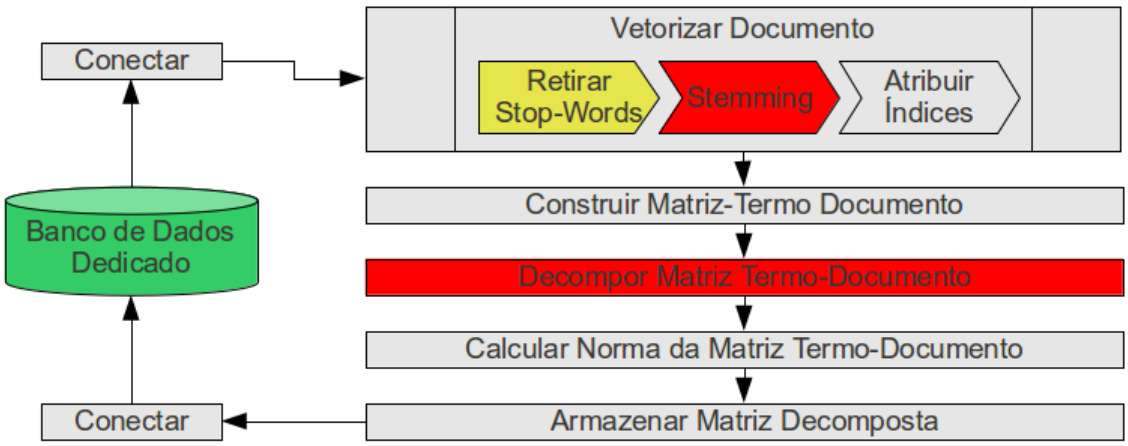


Figura 4.7: Fluxograma simplificado do indexador desenvolvido.

Construindo cada peça independentemente, o processo de integração requeriu adequação das entradas e saídas, principalmente das partes importadas, e a definição de uma interface flexível e otimizada. A função de decomposição, por ter entradas e saídas de tamanho relevante utilizava inicialmente arquivos binários gravados em disco que foram substituídos por versões maiores porém mais padronizados em texto puro. Após compilação deste módulo foi gerado apenas um arquivo binário de aproximadamente 100 KB e configuração externa via arquivo em texto puro.

- **Vetorizar Documento:** Esta tarefa toma lugar individualmente para cada documento em três etapas serializadas:
  - Retirar Stop-Words: Tomando como entrada um documento com codificação padronizada a exclusão das palavras com baixa relevância semântica é o primeiro processo e favorece uma redução considerável de tamanho e carga de trabalho para as próximas etapas.
  - Stemming: Recebendo uma lista enxuta de palavras não necessariamente únicas esta etapa extrai os radicais de cada uma segundo as regras configuradas para o idioma. O indexador desenvolvido manteve foco no idioma português do Brasil, mas pode ser facilmente extendido para tratar outras línguas. Eventuais tentativas de extrair raízes de palavras estrangeiras retornam, na maioria das vezes, a palavra original.
  - Atribuir Índices: Partindo da lista mais compacta de raízes classificadas como termos, nesta etapa são atribuídos índices para definir a linha (documento), coluna (termo) e valor (frequencia) de cada célula.
- **Construir Matriz Termo-Documento:** Cada documento transformado em vetor é empilhado horizontalmente passando a constituir uma linha da

matriz termo-documento. A medida que a coleção é mapeada, as colunas da matriz também aumentam em quantidade e tamanho. Este fato, no entanto, não afeta a integridade da matriz uma vez que o formato de Harwell-Boeing utilizado não atribui valor para as células não instanciadas.

- **Decompor Matriz Termo-Documento:** A decomposição da matriz montada no ítem anterior ocorre através da execução do algoritmo de Arnoldi-Lanczos sob sua forma compactada por colunas. Nesta estratégia as multiplicações internas necessárias da matriz transposta por vetores fica otimizada e evita gargalos de memória e processamento, mesmo para bases maiores.

- **Calcular Norma da Decomposição Truncada:** Decomposta e truncada em versões mais compactas (menos eixos), o conhecimento prévio da norma por linhas (documentos) agiliza as buscas dado seu requisito na equação (2.3). Para calcular a norma, uma aproximação da matriz original é computada gradativa e parcialmente, obtendo valores aproximados e satisfatórios, consumindo tempo significativo quando comparado com outras tarefas.

- **Armazenar Matriz Decomposta:** Uma vez decomposta e com norma por linhas previamente calculada faz-se necessário armazenar novamente o resultado tornando-o disponível para o próximo módulo de busca. Esta operação tem a característica de ser intensiva em memória permanente (disco rígido).

4.4.2 Implementação

No desenvolvimento do indexador os processos e sub-processos foram organizados como ilustra o diagrama de caso de uso apresentado na figura 4.8. O banco de dados acessado foi modelado externamente por ser utilizado apenas via conexão padrão.

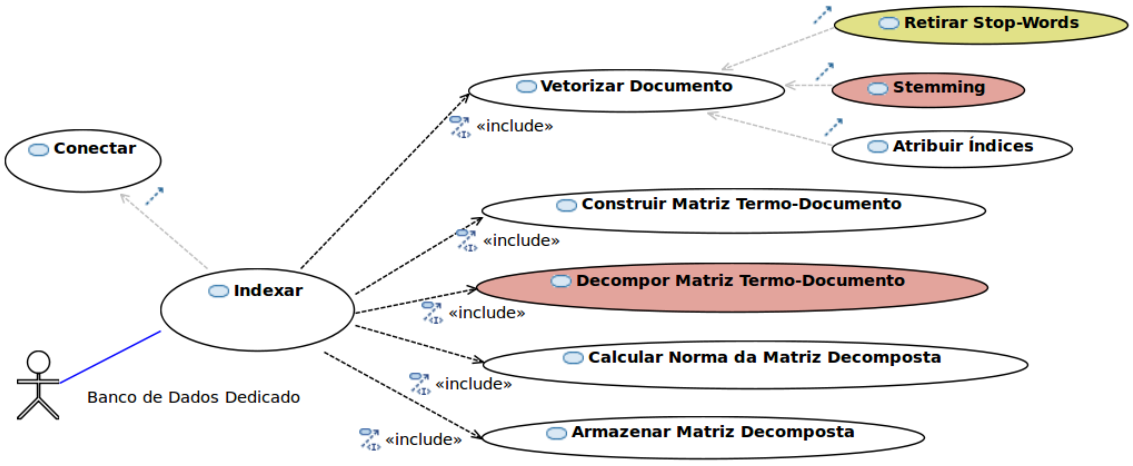


Figura 4.8: Caso de uso detalhado do indexador desenvolvido.

O desenvolvimento do indexador, por ser crucial, foi pautado nas experiências realizadas anteriormente, sendo escolhida a linguagem C (ANSI) para implementação. Como citado, funcionalidades internas foram divididas em funções específicas para chamada serializada. Na primeira versão estável não foram codificados mecanismos para acelerar processamento através de rotinas multi-tarefa (*multi-thread*) uma vez que estas exigem controle de concorrência mais apurado e aumentam consideravelmente a complexidade de programação.

Minimizar dependências externas é importante na manutenção de requisito estrito de desempenho. Operando sobre um banco de dados dedicado e reconhecidamente rápido para os padrões atuais, o código do indexador fez uso somente do driver para conexão ao SGBD e bibliotecas padrão do C (ANSI) leves e otimizadas no consumo de memória e processamento. Na compilação foi utilizado o compilador padrão chamado GCC[7] na versão 4.3, sem otimizações de plataforma.

Baseado no código disponível do SVDPACKC já citado, foi utilizada a estratégia de decomposição pelo método de Lanczos por vetor simples simétrico (las2). Avaliando os testes simples disponíveis no próprio código e as características genéricas da matriz termo-documento a ser decomposta este método apresentou desempenho e robustez satisfatórios sem restringir estruturalmente a entrada.

Como previsto, mesmo apresentando carga de processamento algébrico elevada, o acesso a memória segundária (disco rígido) ainda constituiu gargalo na execução. Buscando implementar uma configuração genérica e suficientemente reconfigurável para outras aplicações, foi tomada a decisão de escrever resultados parciais de cada etapa (construção da matriz, decomposição e cálculo das normas) em arquivos texto padronizados, previamente a sua inserção no banco de dados. Esta alternativa tornou a computação mais robusta contra interrupções, podendo ser reiniciada a partir do último resultado parcial, mas gerou intenso acesso ao disco elevando o tempo total do processo. No entanto, considerando a massa de dados de teste proposta, originada no HUCFF, e os benefícios acadêmicos da entrada/saída padronizada, os tempos de execução se mostraram razoáveis na prática, não extrapolando o intervalo de cinco horas em nenhum caso.

A configuração do serviço em batelada mereceu atenção atenção especial pela característica da carga de trabalho e demanda. Com o gatilho de início de operação fixo no tempo e independente do sistema operacional, é necessário garantir disponibilidade computacional e desacoplamento de outras tarefas neste intervalo. As prioridades de execução do serviço e do acesso a rede, para o SGBD em outro computador, devem ser garantidas minimamente para sucesso na indexação.

[7]GNU C Compiler

4.5 Máquina de Busca: WebApp

Implementado como uma barra de pesquisas, constitui interface popular na web para recuperar informações mineradas. Em um conceito prático e limpo visualmente, a barra de pesquisa traduz uma caixa de texto na qual é inserida a informação para consulta e um botão para pesquisar. Como resultado são expostos registros classificados referentes a coleção de documentos indexada. Em um segundo momento, informações mais detalhadas sobre determinado resultado são obtidas explorando o documento desejado através de referências fornecidas.

Como mencionado, avaliar uma consulta envolve os cálculos algébricos mostrados, em forma matricial aproximada pelo truncamento, na equação (4.2). Nesta, o vetor e_j é uma linha da matriz identidade que serve apenas para selecionar o respectivo cosseno. A matriz A_k é uma aproximação e pode ser substituída por sua decomposição, como mostrado, e o vetor q_n é resultado da vetorização da consulta seguindo as mesmas regras já desenvolvidas para os documentos. A partir deste prisma, a computação das relevâncias de cada documento para uma dada consulta será avaliada observando o denominador e numerador da referida equação:

$$\cos(\theta_j) \approx \frac{e_j A_k q}{||e_j A_k||_2 ||q||_2} = \frac{e_j U_{m,k} \Sigma_{k,k} V_{n,k}^T q_n}{||e_j U_{m,k} \Sigma_{k,k} V_{n,k}^T||_2 ||q_n||_2} \tag{4.2}$$

Denominador

A avaliação do denominador requer o cálculo de duas normas euclidianas, sendo uma por linhas (documentos) da matriz aproximada e outra do vetor-consulta. O primeiro caso, mostrado na equação (4.3) é uma operação custosa sendo necessário remontar a matriz gradualmente para quantificar a norma. Como vantagem, este processamento somente é executado após cada indexação e depende apenas da coleção de documentos. Sem vínculos com futuras consultas, fica previamente disponível e carregado em memória para acesso imediato. No segundo caso do vetor-consulta a norma é mais simples e leve computacionalmente, mas deve ser calculada sob demanda a cada consulta. Este último passo, por sua vez, pode ser otimizado de forma a considerar somente elementos não nulos, reduzindo consideravelmente o número de operações disponíveis.

$$\underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}}_{e_j} \underbrace{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \vdots & \vdots & \vdots \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}}_{U_{m,k}} \underbrace{\begin{bmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \end{bmatrix}}_{\Sigma_{k,k}} \underbrace{\begin{bmatrix} \diamond & \diamond & \dots & \diamond \\ \diamond & \diamond & \dots & \diamond \\ \diamond & \diamond & \dots & \diamond \end{bmatrix}}_{V_{n,k}^T} \tag{4.3}$$

Numerador

O cálculo do numerador depende estritamente do vetor-consulta, sendo por isso executado sob demanda a cada pesquisa. Buscando atenuar a carga neste processo e aumentar a eficiencia, foi desenvolvida uma metodologia otimizada por passos $X \to Y \to Z$ para este caso, mostrada na equação (4.4).

$$\underbrace{e_j \quad \underbrace{U_{m,k} \quad \underbrace{\Sigma_{k,k} \quad \underbrace{V^T_{n,k} \quad q_n}_{X_{k,1}}}_{Y_{k,1}}}_{Z_{m,1}}} \tag{4.4}$$

A estratégia de avaliação se traduz em iniciar calculando a partir da operação matriz-vetor $V^T q$ mostrada na equação (4.5). Como pode ser observado, o algebrismo trata da soma ponderada de colunas da matriz V^T por pesos no vetor-consulta (q). Computacionalmente, utilizando estruturas adequadas, como mapas, é possível otimizar esta operação poupando ciclos de trabalho preciosos.

$$V^T_{n,k} q_n = \begin{bmatrix} \diamond & \diamond & \dots & \dagger \\ \diamond & \diamond & \dots & \dagger \\ \diamond & \diamond & \dots & \dagger \end{bmatrix} \begin{bmatrix} \cdot \\ \alpha \\ \vdots \\ \beta \end{bmatrix} = \alpha \begin{bmatrix} \diamond \\ \diamond \\ \diamond \end{bmatrix} + \beta \begin{bmatrix} \dagger \\ \dagger \\ \dagger \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{4.5}$$

No próximo passo, utilizando da matriz diagonal Σ e o vetor X, previamente calculado, é possível melhorar a eficiência deste processo tranformando a operação convencional em uma multiplicação ponto a ponto a partir da diagonal não nula dos valores singulares, como mostrado na equação (4.6). O produto Y é alcançado em um numero de passos proporcional ao truncamento utilizado.

$$\begin{bmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \bullet \cdot x_1 \\ \bullet \cdot x_2 \\ \bullet \cdot x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{4.6}$$

A multiplicação da matriz U pelo vetor Y calculado ocorre sem adapatações, sendo a etapa mais custosa, como mostrado na equação (4.7).

$$\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \vdots & \vdots & \vdots \\ \star & \star & \star \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} \tag{4.7}$$

Uma vez calculado, o vetor Z é ordenado decrescentemente para classificar a relevância dos documentos. Este trabalho é realizado usando uma estrutura especial

que guarda os índices originais, sendo futuramente utilizados como referência aos documentos na coleção indexada. Neste projeto foi adotada a estratégia de retornar um número fixo de documentos considerados mais relevantes (topo da lista) ao invés de usar um gatilho segundo avaliação convencional. Acredita-se que esta alternativa permitirá, em um primeiro momento, melhores resultados e também uma ponto de partida para definição dos limiares de relevância.

4.5.1 Funcionamento

A interface de busca implementada, baseada na indexação anterior, é descrita simplificadamente no fluxograma mostrado da figura 4.9. Da mesma forma, a cor vermelha indica funcionalidade importada enquanto amarelo sinaliza recurso externo utilizado.

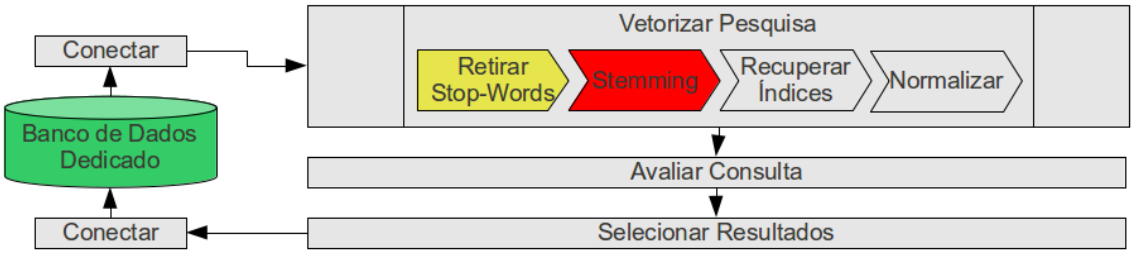


Figura 4.9: Fluxograma simplificado da recuperação de informação.

De forma semelhante ao indexador o desenvolvimento deste módulo foi segregado em funções chamadas sequencialmente. O método de *stop-words* foi integralmente re-implementado nesta nova plataforma utilizando, por questão de coerência, a mesma lista de palavras fornecida pelo projeto Snowball. A função do *stemming* também foi importada do projeto Snowball configurando um pacote independente nesta nova aplicação. Cada etapa foi exaustivamente testada antes da integração.

- **Vetorizar Consulta:** A vetorização da consulta ocorre em mecanismo semelhante à vetorização dos documentos, tendo único diferencial nas últimas etapas de recuperação dos índices e normalização. Por este motivo, é fundamental que este processo seja coerente ao primeiro, operando sobre as mesmas raízes e índices.

  - Retirar Stop-Words: Tomando a consulta fornecida a exclusão das palavras com baixa relevância semântica é o primeiro processo e favorece uma redução considerável de tamanho e carga de trabalho para as próximas etapas, realizadas também sob demanda.

- Stemming: Recebendo uma lista enxuta de palavras não necessariamente únicas esta etapa extrai os radicais individualmente segundo as regras configuradas para o idioma. As mesmas regras para idiomas no *stemming* no indexador foram configuradas neste caso.
- Atribuir Índices: Partindo da lista compacta de raízes extraídas, nesta etapa procura-se vincular cada termo fornecido à um índice minerado pelo indexador (termos não indexados são descartados), armazenando seu valor na respectiva célula do vetor pesquisa, montado progressivamente.
- Normalizar: Uma vez constituído, o vetor pesquisa, sua norma euclidiana é calculada e armazenada para avaliação segundo equação (2.3). Diferente da normalização por linhas da matriz aproximada realizada no indexador, este processo foi otimizado para considerar somente os termos utilizados, implicando em número de operações proporcionais ao termos válidos na consulta.

- **Avaliar Consulta:** Utilizando a equação (4.2), otimizada para os cálculos das normas previamente realizados, cada documento minerado na base tem sua relavância calculada em relação à consulta realizada, sendo os resultados armazenados para posterior classificação. Esta etapa oferece a maior carga no seu processamento além de ocupar quantidade significativa de espaço pelo manutenção das matrizes de decomposição e resultados parciais em mémoria.

- **Selecionar Resultados:** A seleção ocorre essencialmente de duas formas: valores gatilho ou ordenação total. No primeiro caso (mais rápido) a relevância é aferida até que um número configurado de reaultados superem um limite mínimo estabelecido (gatilho) de relevância, sendo então retornados. No segundo caso, todas as relevâncias são calculadas e ordenadas para retorno de um número configurado dos resultados mais próximos. Como citado anteriormente, este último método oferece melhores resultados e foi selecionado para implementação neste trabalho.

4.5.2 Implementação

A figura 4.10 ilustra caso de uso detalhado da aplicação web desenvolvida.

O desenvolvimento deste módulo de recuperação da informação foi realizado utilizando plataforma Java na plataforma J2EE[8], considerando a possibilidade de interface web. Esta decisão foi pautada em requisitos de robustez algébrica intrínseca ao processo e flexibilidade para exposição dos resultados em forma de páginas web. Corporativamente a plataforma escolhida oferece recursos para lidar com

[8] *Java 2 Enterprise Edition*

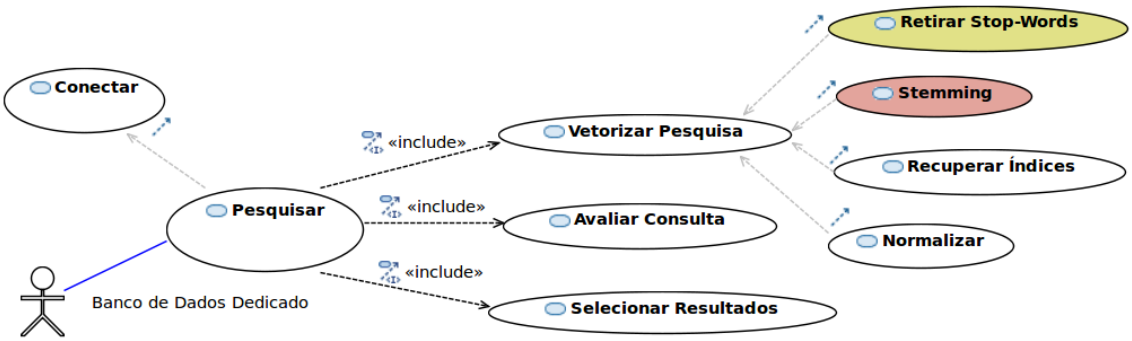


Figura 4.10: Caso de uso detalhado da recuperação de informação.

grande quatidade de memória alocada e agilidade no processamento de pesquisas simultâneas.

Apesar da simplicidade aparente, máquinas de busca na web requerem tratamento de memória diferenciado quanto ao volume e estratégia de alocação. Em relação a quantidade de memória ocupada, todo o resultado da decomposição (matrizes U, Σ e V truncadas) realizada pelo indexador além das normas individuais de cada documento minerado e armazenado no banco de dados necessita estar disponível na memória para cada consulta realizada.

O pré-carregamento durante a inicialização da aplicação é uma estratégia inteligente para resolver o gargalo de acesso à memória sob demanda, tornando assim o tempo de busca razoável e normalmente inferior a um segundo. Entretanto, esta solução apresenta problemas de volatilidade pela quantidade de memória alocada sem uso constante. Alternativas para gerenciamento eficiente de memória requerem configurações especiais e foram realizadas neste projeto para melhorar a usabilidade.

Buscando uma implementação genérica e configurável, neste módulo as funcionalidades visuais, dedicadas aos navegadores foram divididas das lógicas, abordadas no caso de uso referido na figura 4.10. Nesta estrutura, a máquina de busca desenvolvida neste trabalho foi empacotada em forma de biblioteca compilada e independente, viabilizando utilização futura em outros projetos que utilizem o indexador desenvolvido ou forneçam a entrada padronizada.

Finalmente, considerando o ambiente de rede altamente escalável, a possibilidade de segregação de servidores e distribuição de carga configura outra vantagem desta implementação modularizada e multi-plataforma. Considerando o SGBD como único vínculo entre indexador e máquina de busca, não existem restrições quanto ao número de buscadores em operação simultânea.

Capítulo 5

Resultados Utilizando Documentos Clínicos

A implementação dos módulos de **extração**, **indexação** e **busca** foi realizada com sucesso, verificado em testes unitários. Tomando **documentos clínicos** como textos de conteúdo médico genérico, a análise de resultados em coleções reais fornece métricas quantitativas úteis para avaliação deste projeto. Foram realizados testes em duas coleções distintas de documentos: descrições do CID-10 e laudos do HUCFF.

Cada teste encontra-se sub-dividido em etapas lógicas de interesse na verificação. No caso das descrições no CID-10, por se tratar de uma coleção menor e de apresentação viável nas páginas deste texto, os resultados númericos intermediários são apresentados em parte visando validação das idéias e do código compilado. Para a coleção proveniente dos laudos do HUCFF, são exploradas outras dimensões da computação, sendo expostas métricas mais gerais devido ao tamanho da apresentação incompatível da coleção e das abstrações matemáticas geradas.

5.1 Prova de Conceito: CID-10

Como primeiro teste foi tomada uma prova de conceito baseada no **Código Internacional de Doenças** em sua décima versão, ou simplesmente CID-10. Esta coleção se traduz numa lista de patologias conhecidas e identificadas por códigos alfa-numéricos, servindo ao intercâmbio de informações médicas de forma padronizada, compacta e reservada.

A lista do CID-10 encontra-se divida em capítulos, grupos e sub-grupos de doenças e contém 12.450 ítens no total. Na avaliação do código compilado basta que o tamanho atenda requisitos mínimos, sem interferir na lógica. Porém, buscando melhor apresentação didática, foi selecionado apenas o capítulo VII, que aborda "Doenças do olho e anexos" com 262 ítens e códigos entre H000 até H599.

5.1.1 Preparação dos documentos

A preparação dos documentos é a primeira etapa a ser executada. Mesmo numa coleção reduzida como o capítulo VII do CID-10, não é razoável listar nestas páginas todos os ítens presentes. Neste sentido, os primeiros documentos são listados para avaliação por amostra dos processos executados.

Stop-List

H000	→	Hordéolo e outras inflamações profundas das pálpebras
STOP	→	hordéolo outras inflamações profundas pálpebras
H001	→	Calázio
STOP	→	calázio
H010	→	Blefarite
STOP	→	blefarite
H011	→	Dermatoses não infecciosas da pálpebra
STOP	→	dermatoses não infecciosas pálpebra
H018	→	Outras inflamações especificadas da pálpebra
STOP	→	outras inflamações especificadas pálpebra

Tabela 5.1: Demonstração do procedimento de *stop-words* no CID-10.

Nesta etapa é importante observar que para efeito de comparação com as palavras presentes na *stop-list* todos os sinais de pontuação foram retirados e as palavras foram transformadas para sua forma minúscula.

Stemming

H000	→	hordéolo outras inflamações profundas pálpebras
STEM	→	hordéol outr inflam profund pálpebr
H001	→	calázio
STEM	→	calázi
H010	→	blefarite
STEM	→	blefarit
H011	→	dermatoses não infecciosas pálpebra
STEM	→	dermatos nã infecc pálpebr
H018	→	outras inflamações especificadas pálpebra
STEM	→	outr inflam especific pálpebr

Tabela 5.2: Demonstração do procedimento de *stemming* no CID-10.

Após o *stemming* é perceptível a atuação do algoritmo utilizado, baseado em partição das palavras. Na maior parte das vezes os sufixos e algumas vezes prefixos das palavras recebidas do processo de *stop-words* são retirados, sem alterar o núcleo.

5.1.2 Matriz Termo-Documento

A matriz termo-documento construída pelo indexador a partir da coleção do CID-10 é primeiro montada no formato Harwell-Boeing. Esta estrutura viabiliza a entrada da informação na função de decomposição e também permite outros testes em plataformas compatíveis. Por simplicidade os campos do ponteiro de colunas, índice de linhas e valores estão dispostos em uma linha cada. Na demonstração que segue o arquivo original foi truncado a direita para se conter na página.

```
Term-By-Document Matrix for linkSaude Project                    20110802
             7             1             1             1             0
RUA                      262           250           983             0
(8I3)           (17I3)          (5E15.8)            (5E15.8)
1 2 3 4 6 7 8 12 24 25 27 28 41 46 47 51 52 53 54 56 57 58 59 60 64...
142 7 34 233 234 129 99 152 153 154 162 169 185 245 248 249 250 252...
1.0000000000E+00 1.0000000000E+00 1.0000000000E+00 1.0000000000E+00...
```

Por inspeção do cabeçalho apresentado nota-se que a matriz construída possui 262 linhas e 250 colunas tendo um total de 983 elementos diferentes de zero armazenados com precisão de dez casas decimais. Utilizando código Matlab, disponível nos anexos em A.2, referente a esta prova de conceito, a matriz termo-documento A foi carregada em memória, permitindo impressão parcial dos seus elementos e verificação amostral da esparsidade.

```
A(1:10, 1:10) =

     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

No mesmo ambiente, é possível realizar uma demonstração gráfica qualitativa, fazendo ao mesmo tempo uma aferição da densidade desta matriz como mostrado na figura 5.1. Na mesma figura ainda é possível observar que a densidade calculada para esta matriz é de 1,472% sendo inferior aos 5% inicialmente estimados.

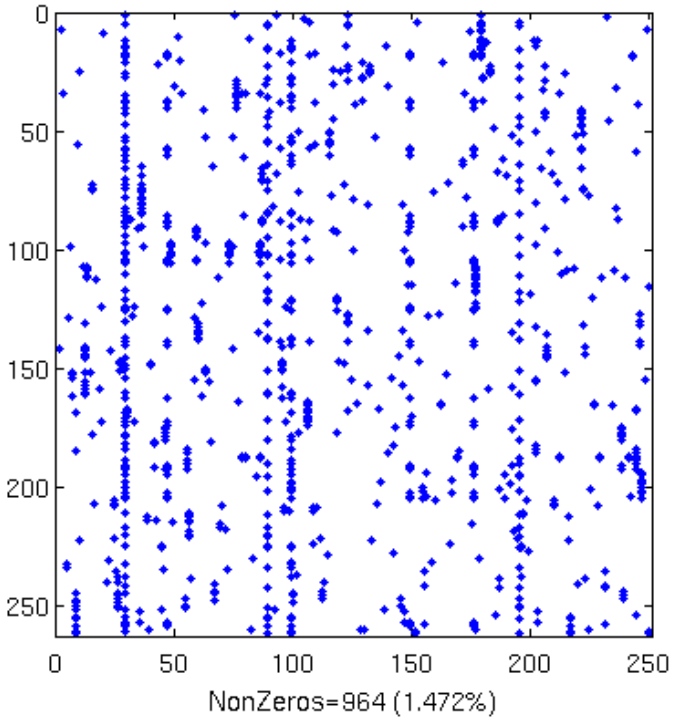


Figura 5.1: Avaliação visual da matriz termo documento do CID-10.

5.1.3 Decomposição em Valores Singulares

Construída a matriz termo-documento e admitindo ser uma abstração razoável da coleção, a decomposição em valores singulares é o próximo passo na indexação semântica latente. Buscando primeiramente validação numérica para o algoritmo implementado, os resultados calculados via indexador são comparados com valores obtidos no software Matlab. Com propósito didático, foi adotado truncamento em 20 valores singulares e as normas Frobenius e euclidiana tomadas como métricas.

Sabendo que os valores singulares calculados são únicos, e os vetores singulares podem ser diferentes (não únicos) os valores mostrados na equação (5.1) foram obtidos para as matrizes truncadas mostradas com subscritos do software utilizado:

$$\begin{cases} ||U_{matlab} - U_{indexer}||_F = 4,8990 & \rightarrow \text{ são diferentes} \\ ||\Sigma_{matlab} - \Sigma_{indexer}||_F = 1,8353 \times 10^{-10} & \rightarrow \text{ são muito parecidas} \\ ||V_{matlab} - V_{indexer}||_F = 4,8990 & \rightarrow \text{ são diferentes} \end{cases} \tag{5.1}$$

Analisando os resultados numéricos é possível concluir que as matrizes U e V calculadas no Matlab e no Indexer não são iguais. Este fato, no entanto não comprometeu a semelhança dos valores singulares que também pode ser notada individualmente em uma amostra parcial dos dez maiores valores calculados.

```
[  Sc        Sm     (Sc-Sm)]
15.3958   15.3958   -0.0000
 9.8712    9.8712   -0.0000
 6.3841    6.3841   -0.0000
 5.0935    5.0935   -0.0000
 5.0003    5.0003   -0.0000
 4.4215    4.4215   -0.0000
 4.3491    4.3491    0.0000
 4.2701    4.2701    0.0000
 4.1751    4.1751    0.0000
 4.1203    4.1203    0.0000
```

Ao recompor a matriz termo-documento aproximada pelo truncamento, é esperado que tanto o cálculo do Indexador quanto no Matlab sejam muito próximos. Este resultado pode ser confirmado pela avaliação de normas específicas em ambos os casos. A norma Frobenius da diferença entre a matriz original e a aproximação mostrada na equação (5.2) é coerente com o teorema (3.4) nos dois ambientes testados, comprovando consistência e precisão numérica da implementação.

$$||A - A_{indexer}||_F = ||A - A_{matlab}||_F = \sqrt{\sigma_{21} + \ldots + \sigma_{250}} = 19,3989 \tag{5.2}$$

No caso da norma euclidiana se verifica resultado semelhante para cálculo da diferença entre a matriz termo-documento original e a aproximação mostrada na equação (5.3) que também confere com o teorema (3.6) em ambos os softwares.

$$||A - A_{indexer}||_2 = ||A - A_{matlab}||_2 = \sigma_{21} = 3,0615 \tag{5.3}$$

Analisando os resultados de forma qualitativa mais ampla, um questionamento razoável sobre valores singulares seria até que ponto (ou quantos) valores necessitariam ser calculados para se atingir determinado desempenho desejado. Em [1] Berry propõe quantificar, através da norma Frobenius, a informação alterada relativa ao erro de aproximação e a matriz original, pela equação (5.4) neste caso.

$$\frac{||A - A_{20}||_F}{||A||_F} = 0.6071 \approx 60\% \text{ de informação alterada} \tag{5.4}$$

No entanto, para sistemas maiores frequentemente não é possível avaliar a norma Frobenius da matriz original devido a suas dimensões. Neste caso, a experiência deste trabalho sugere que uma análise qualitativa do decaimento dos valores singulares como mostrado na figura 5.2 pode fornecer bom ponto de partida para estimar a precisão da aproximação, auxiliado também pelas equações (3.4) e (3.6).

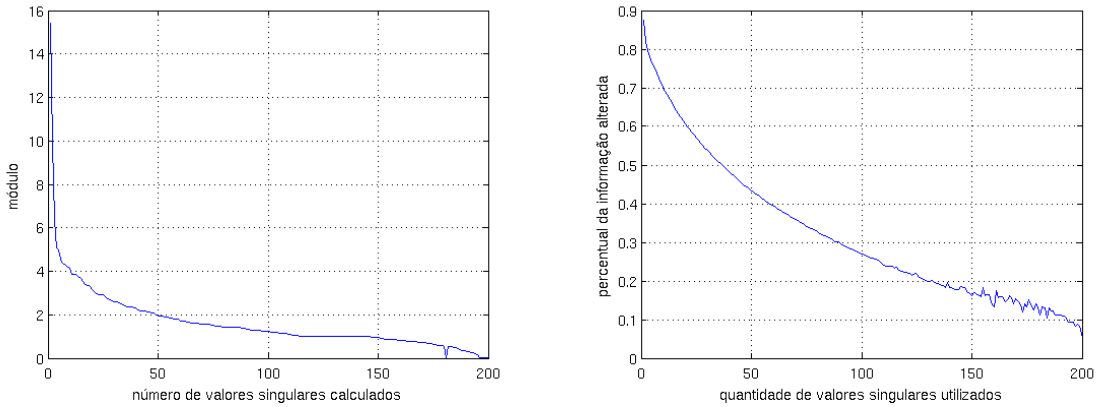


Figura 5.2: Decaimento dos valores singulares e perda da informação para CID-10.

5.1.4 Recuperação da Informação

A implementação real, mostrada na figura 5.3, teve a decomposição truncada em 200 valores singulares, sem apresentar alteração de informação considerável. Devido ao tamanho reduzido da coleção, os tempos de consulta foram inferiores a décimos de segundo e praticamente inalterados com quantidade de palavras fornecidas.

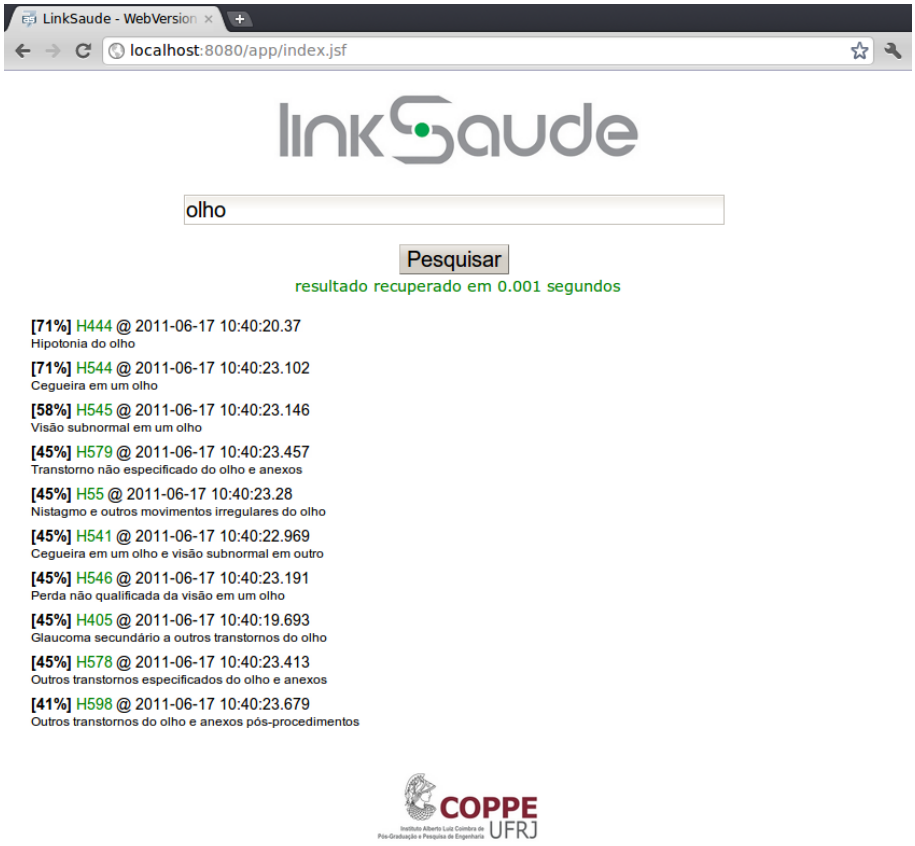


Figura 5.3: Interface web para recuperação da informação no CID-10.

5.2 Banco de Dados Hospitalar

O banco de dados hospitalar, utilizado nesta seção, serve ao propósito de estender a demonstração das funcionalidades gerais abordadas na prova de conceito para as dimensões corporativas de uma grande instituição de saúde. Esta tentativa busca sedimentar os avanços alcançados através de métricas mais qualitativas, quando as proporções do problema inviabilizam aferições exatas. Neste ambiente, a nova coleção de documentos, alvo da mineração, são laudos provenientes de vários setores do HUCFF que utilizam o software Medtrak, como já foi citado.

5.2.1 Preparação dos documentos

Os processos de *stop-list* e *stemming* seguem a mesma metodologia demonstrada para o CID-10, utilizando inclusive os mesmos binários e arquivos de configuração. Como resultado, as mesmas palavras presentes na *stop-list* são retiradas e a extração de raízes funciona recortando prefixos e sufixos para obter o núcleo. Agora, merecem considerações particulares as palavras geradas por erros gramaticais que potencialmente poluem a matriz termo-documento e a recuperação da informação.

Caracterização dos Documentos

Superficialmente documentos são caracterizados pelo tamanho e frequencia dos mesmos, na coleção testada. Ao todo existem 4638 documentos vazios (registros não preenchidos). Documentos com apenas um caractere, automaticamente desconsiderados no processo de *stemming* (valor semântico não significativo) somam 67. O maior documento armazenado tem 11.554 caracteres sendo o tamanho médio de 553. O número médio de termos por documento é estimado em 50, sendo a norma euclidiana do menor documento igual a 0, por estar vazio, e a maior igual a 113. A norma média dos documentos é 7,68.

Lista de Termos

Uma abordagem baseada em frequencia foi utilizada para salientar caracterísitcas particulares do conjunto de 52.222 termos extraídos da coleção testada. No conjunto de termos mais frequentes temos no topo o termo "normal" com 177.402 ocorrências na coleção, seguido por outros também comuns e listados em parte na lista que segue, com a frequencia na coleção entre parênteses.

```
normal(177402) norm(141511) direit(110172) esquerd(106702) nã(97095)
calibr(96965) alter(89893) aspect(89524) volum(85622) realiz(78212)
pared(70595) observ(67731) exam(65187) ausênc(59685) mucos(58533)
form(57249) contorn(54772) pulmon(53623) regul(52551) parênquim(50312)
```

No outro extremo, os termos de menor frequência ocorrem, obviamente, apenas uma vez em toda a coleção. Surpreendentemente estes termos somam elevados 29.025 ocorrências unitárias, sendo responsáveis por mais da metade do conjunto e das colunas da matriz termo-documento gerada posteriormente. Como características especiais deste conjunto pode ser notado que muitos deste, conforme a lista parcial que segue, são provenientes de erros gramaticais ou inserções de códigos e registros como o CRM nos documentos.

```
endoscoscóp(1) oponenet(1) nódulc(1) micrroangiopat(1) moent(1) pôs(1)
nódull(1) reexpansã(1) esxtra(1) 61213(1) oomelhor(1) 120mm(1) coccíp(1)
61202(1) coindid(1) 61096(1) mofitos(1) endoprotes(1) xyloain(1)
```

5.2.2 Matriz Termo-Documento

A matriz termo-documento, construída pelo indexador a partir da coleção proveniente do HUCFF, é primeiro montada no formato Harwell-Boeing. Como citado, esta estrutura viabiliza a entrada da informação na função de decomposição e também facilita o estabelecimento de marcos, no caso de interrupções abruptas e inesperadas no processo. Por simplicidade os campos do ponteiro de colunas, índice de linhas e valores estão dispostos em uma linha cada. Na demonstração que segue o arquivo original foi truncado a direita para se conter na página.

```
Term-By-Document Matrix for linkSaude Project                  20110802
             7             1             1             1             0
RUA                   212174         52222      11228520             0
(8I3)           (17I3)          (5E15.8)            (5E15.8)
1 2 3 4 5 56 57 59 60 111 112 113 139 142 143 144 145 146 147 148.....
159 165 166 167 168 202 211 215 216 221 292 293 294 621 622 623 657...
1.0000000000E+00 1.0000000000E+00 1.0000000000E+00 1.0000000000E+00...
```

Por inspeção do cabeçalho apresentado nota-se que a matriz construída possui 212.174 linhas e 52.222 colunas, tendo um total de 11.228.520 elementos não-nulos, gravados com precisão de dez casas decimais. O arquivo que armazena esta matriz em disco possui aproximadamente 253 megabytes em texto puro. Dada a impossibilidade de carregar uma matriz com estas dimensões na memória do Matlab, uma demonstração gráfica qualitativa da densidade desta matriz foi realizada utilizando esta ferramenta em dimensões menores como pode ser observado na figura 5.4. Na mesma figura, ainda é possível observar que a densidade calculada desta matriz de 0.1012% é bem inferior aos 5% considerados inicialmente. Como esperado, as dimensões da matriz aumentem com o tamanho da coleção e decrescem com a frequência mínima de termos considerados, como mostra a figura 5.5.

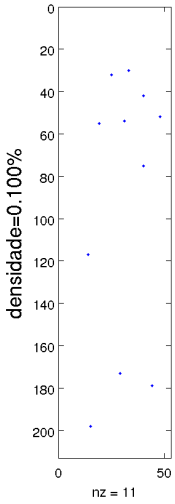


Figura 5.4: Visualização da esparsidade da matriz termo-documento do HUCFF.

Seguindo a metodologia de construção adotada, o número de linhas na matriz termo-documento reflete o tamanho da coleção enquanto as colunas representam os termos extraídos. Admitindo que a quantidade prática de termos técnicos tenda a ser limitada em cada idioma, era esperado uma atenuação no aumento dos termos para grandes conjuntos. No entanto, foi observado crescimento linear nas duas dimensões da matriz que pode ser interpretada como o não atingimento do referido ponto de saturação. Como causas deste fenômeno merecem consideração a recente e gradual digitalização do hospital, resultando numa inserção cronológica de novos termos provenientes de setores recém informatizados.

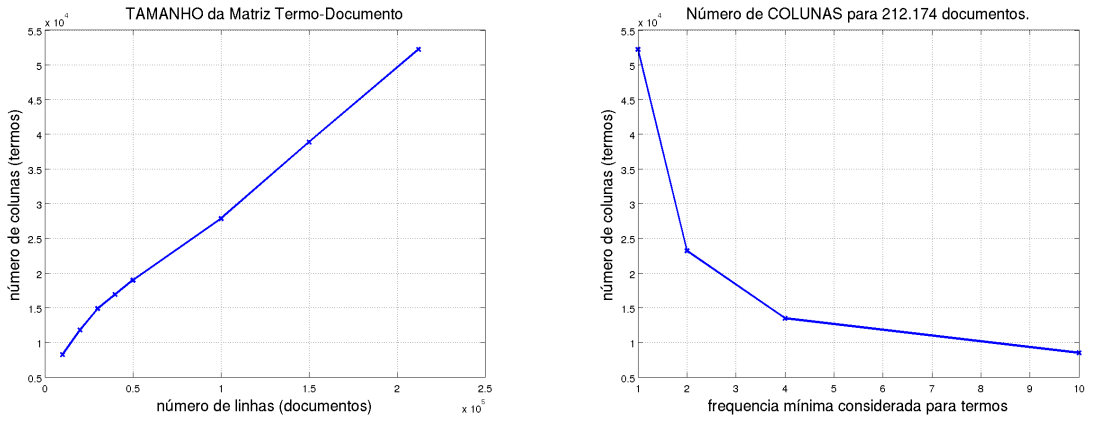


Figura 5.5: Dimensões para tamanho da coleção e frequência de termos.

A frequência mínima de ocorrência dos termos, mostrada na figura 5.5, demonstrou forte impacto no número de colunas da matriz, principalmente para frequência unitária. Com o entendimento que o esforço computacional na indexação e na re-

cuperação da informação é diretamente proporcional ao tamanho desta matriz, e observando a quantidade de termos de baixa ocorrência originados a partir de erros gramaticais, a consideração de frequências mínimas acima da unitária (pelo menos dupla) merece atenção em otimizações futuras.

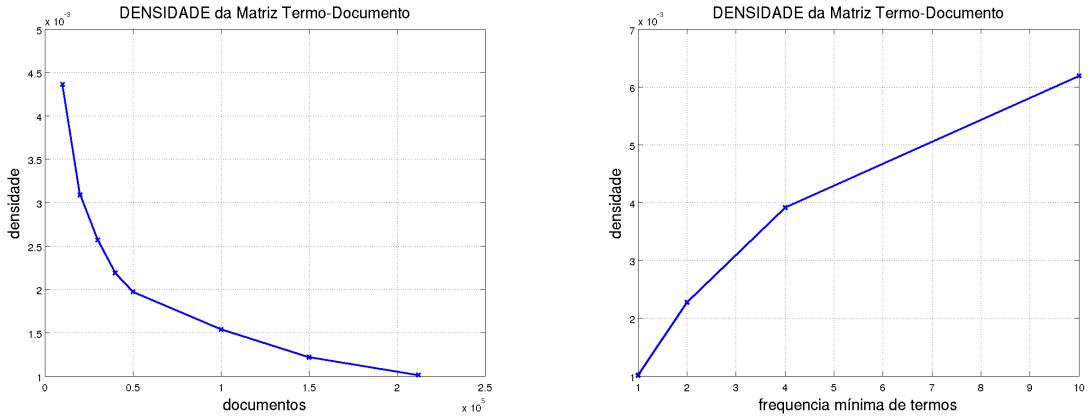


Figura 5.6: Densidade para tamanho da coleção e frequencia de termos.

Análise semelhante ao caso dimensional foi realizada para densidade, parametrizada por variações no tamanho da coleção e também frequência mínima dos termos. Com o aumento do número de documentos indexados nota-se inicialmente uma queda acentuada de densidade, tendendo a estabilizar próximo ao valor de 0,1% aferido. Numa suposição prática, este valor estabeleceria a relação de compromisso entre inserção de novos documentos e aumento dos termos (novos supostamente em menor parte) e erros gramaticais.

Analisando a densidade é perceptível que seu valor dobra para frequência mínima considerada igual a 2, mantendo este comportamento, aproximadamente linear, até 4. Desta observação é possível inferir que um percentual significativo das colunas (termos) da matriz termo-documento são fracamente populadas, ou seja, alteram conteúdo semântico em poucos documentos. A consideração destes, em termos de eficiência na recuperação da informação e carga computacional faz parte dos questionamentos levantados por Berry em [1] e objeto de discussão também neste texto.

5.2.3 Decomposição em Valores Singulares

Considerando a eficiência númerica deste algoritmo mostrada na seção 5.1.3 e buscando solução de compromisso computacionalmente realizável foi adotado, neste caso prático, o truncamento em 200 valores singulares como estimativa inicial. Visualmente, esta escolha implica em compressão significativa da informação, observada na figura 5.7, pela demonstração das proporções relativas da matriz original aproximada e do produto externo matricial resultante.

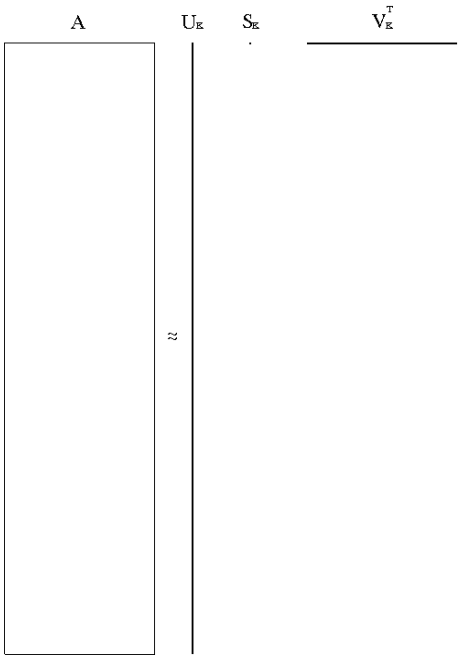


Figura 5.7: Visualização da compressão gerada pela SVD no HUCFF.

Buscando embasar melhor o truncamento escolhido e aumentar a percepção da informação retida após a decomposição, a figura 5.8 mostra os módulos dos primeiros 1000 valores singulares desta mesma matriz. Na prática, esta computação é muito custosa e não escala rotineiramente. Como abordagem inicial, no entanto, serve ao propósito de auxiliar em escolhas mais acertadas para a aproximação.

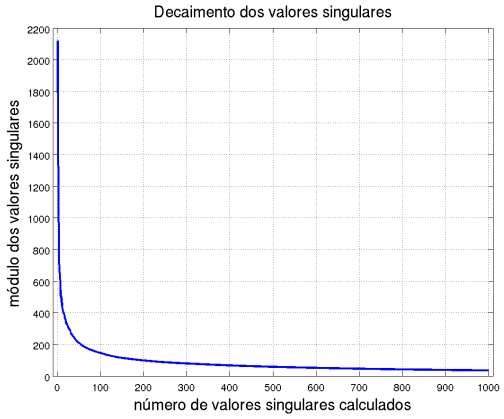


Figura 5.8: Módulos dos valores singulares calculados.

Segundo Berry em [1] a escolha do truncamento é um compromisso entre eficiência e potência computacional. Analisando a figura 5.8 é aceitável que os 200 valores singulares utilizados realizam um truncamento após estabilização da queda acentuada (joelho da curva) dos valores singulares. Este indicativo reforça a idéia de preservação significativa do conteúdo, justificando numericamente a escolha heurística dentro do intervalo apontado por Berry em [1].

O custo computacional pode ser traduzido no consumo de memória primária (RAM) pelo alocamento de matrizes e resultados parciais e também pela quantidade de iterações necessárias até a convergência do algoritmo. Na parte esquerda da figura 5.9 é mostrado o consumo de memória em relação ao tamanho da coleção indexada. Visualmente é detectado um pequeno deslocamento para cima na curva (espaço opcupado pelas variáveis auxiliares) e comportamento linear com baixa inclinação. Esta característica salientou a potencial escalabilidade deste algoritmo, consumindo menos de 1,5 gigabytes na decomposição da maior matriz.

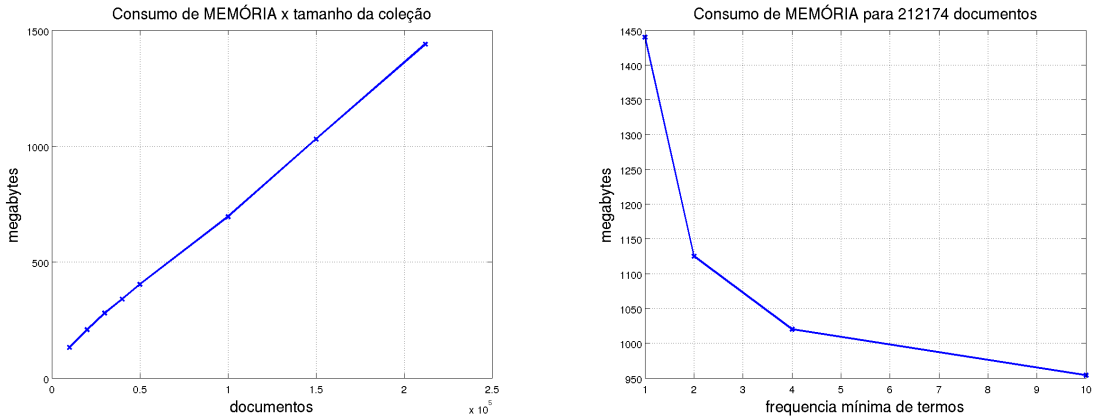


Figura 5.9: Memória consumida para tamanho da coleção e frequencia de termos.

Na parte direita da figura 5.9 o uso de memória é ponderado pela frequência mínima considerada. Como nos outros experimentos, pode ser notado forte decaimento quando a frequência unitária é desconsiderada.

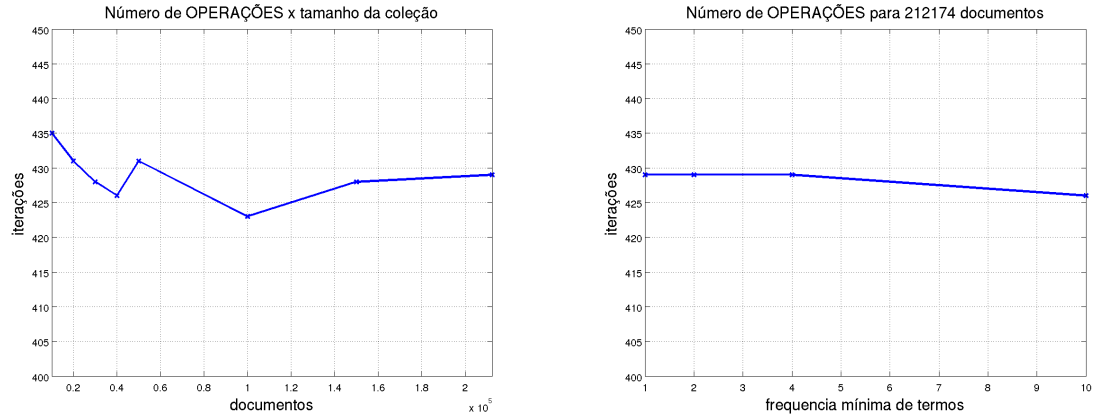


Figura 5.10: Iterações para tamanho da coleção e frequencia de termos.

A quantidade de iterações até a convergência é uma métrica de processamento útil para este algoritmo. Como mostrado na figura 5.10 ocorreram variações insignificantes, dados o tamanho da coleção e a frequência mínima considerada. A

carga final, no entanto, é ponderada pelo tamanho das matrizes utilizadas em cada iteração. A combinação destes fatores permite a dedução de capacidade desta implementação para minerar uma coleção significativamente maior, talvez com alguns milhões de documentos, sem custos proibitivos em servidores modernos.

5.2.4 Desempenho Temporal

O desempenho temporal é avaliado para os quatro processos de **montagem**, **decomposição**, **normalização** e **armazenamento**, realizados no indexador. Todo o código implementado é executado de forma serializada, com funções sendo invocadas sequencialmente. Como já citado, nesta abordagem não foram utilizados recursos de multi-tarefa ou paralelização, que certamente otimizariam consideravelmente os tempos apresentados. Nestas condições, valem as mesmas considerações feitas para o tamanho da coleção e frequência mínima adotada, mostradas na figura 5.5.

Nos experimentos realizados, o tempo de montagem aumentou a uma taxa exponencial com o tamanho da coleção. Esse aumento, provavelmente provocado pela carga das tabelas *hash* utilizadas no mapeamento, permaneceu dentro da faixa projetada para coleções do porte avaliado. O maior valor atingido foi considerado razoável, sendo inferior a 14 minutos para o conjunto completo, para uma operação intensiva em disco. Na possibilidade de mineração em coleções muito maiores seria interessante reconfigurar as tabelas *hash* de maneira que, ao custo de mais memória alocada este tempo fosse mantido no mesmo patamar.

A frequência mínima dos termos não afetou significativamente o tempo de montagem, como esperado e exposto na direita da figura 5.11, dado que este parâmetro somente é aferido depois de composta toda a matriz em suas frequências globais.

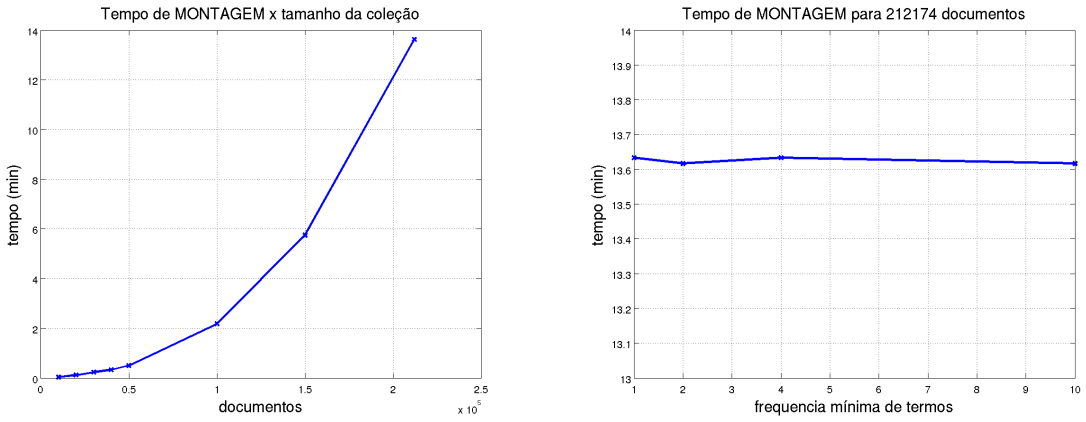


Figura 5.11: Tempos de montagem para tamanho da coleção e frequência de termos.

Montada a matriz termo-documento, sua decomposição é o próximo passo. Como esperado, e sendo o algoritmo de Arnoldi-Lanczos linear conforme a entrada

fornecida, os tempos consumidos na decomposição aumentaram proporcionalmente ao tamanho da coleção. Como mostrado na figura 5.10, o número de iterações praticamente se manteve estável para variações no tamanho do conjunto, sendo seu principal influenciador a quantidade de valores singulares a serem calculados. Neste contexto o aumento verificado na esquerda da figura 5.12 pode ser atribuído, em sua maior parte, ao tempo gasto na operação com matrizes maiores. Coerentemente, nota-se também redução expressiva do tempo de decomposição de matrizes menores, resultado da desconsideração de termos ocorrendo com baixas frequências.

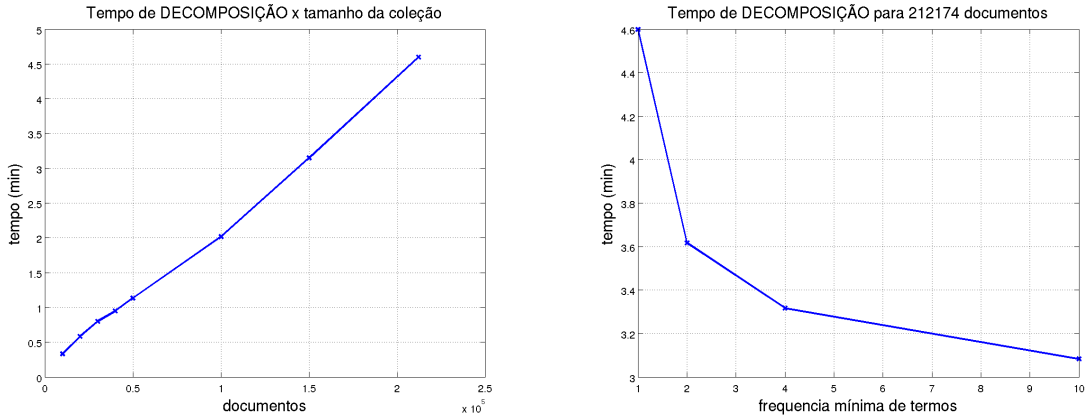


Figura 5.12: Tempos de decomposição para tamanho da coleção e frequência.

Realizada a decomposição e com o resultado gravado em disco, toma lugar a próxima etapa de cálculo das normas da matriz aproximada por linhas (documentos). Pelo fato de envolver grandes multiplicações matricias densas e operar completamente serializado, este procedimento apresentou os maiores tempos individuais. Mesmo neste caso os valores mantiveram patamares aceitáveis, sem extrapolar 5 horas no pior caso.

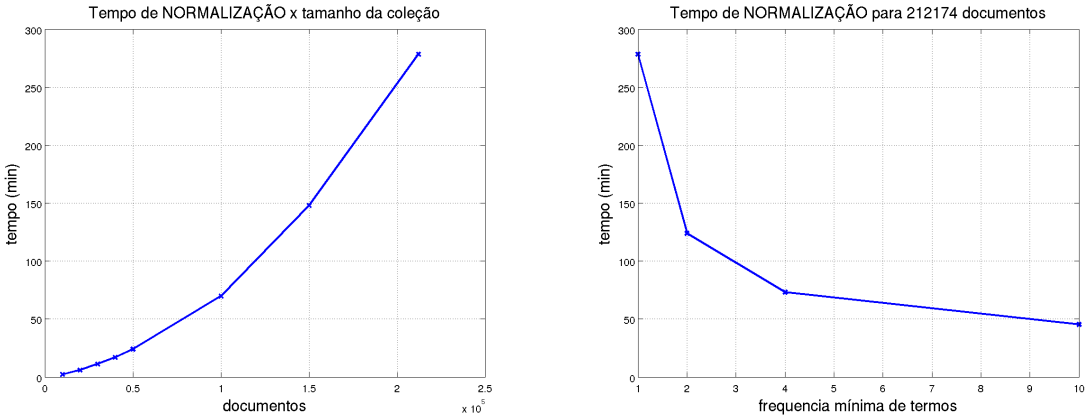


Figura 5.13: Tempos de normalização para tamanho da coleção e frequência.

Realizada indexação prática, o procedimento de armazenamento acontece para transferir ao banco de dados os índices relevantes gerados. Este processo apresentou comportamento linear no consumo de tempo, sem ultrapassar os 7 minutos no pior caso. Pela característica intensiva em disco é fortemente influenciado pela velocidade deste e também pelo volume de dados a ser gravado, vinaculado à frequência mínima adotada, como pode ser observado na figura 5.14.

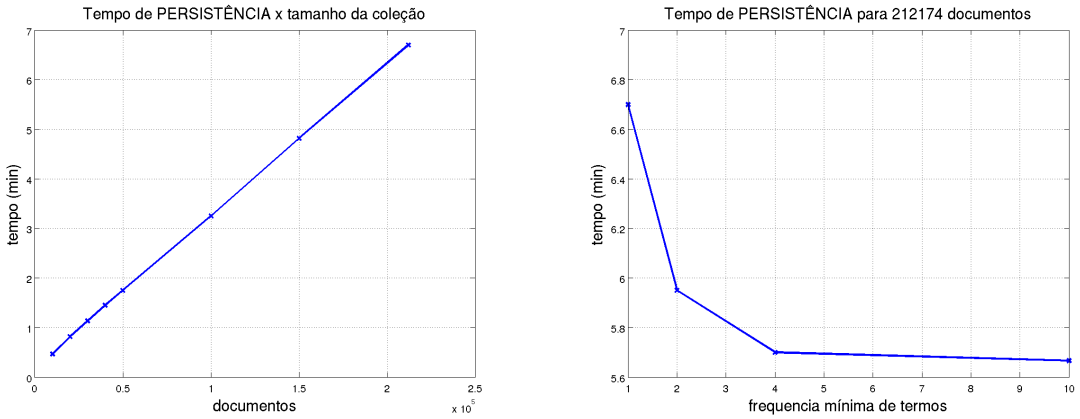


Figura 5.14: Tempos de amazenamento para tamanho da coleção e frequencia.

Finalmente, a figura 5.15 apresenta uma visão mais qualitativa e de alto nível do processo global. Tomando os mesmos parâmetros de tamanho da coleção e frequência mínima considerada pode ser notado a característica ascendente no primeiro caso e descendente no segundo. Comparando as quantidades de tempo consumidas em cada etapa é perceptível a forte predominância do tempo de normalização. Tomado como principal gargalo devido ao elevado tempo relativo, o processo de normalização pode ser fortemente auxiliado por implementações multi-tarefa ou migração deste cálculo para a extração, se tornando menos preciso porém mais diluído.

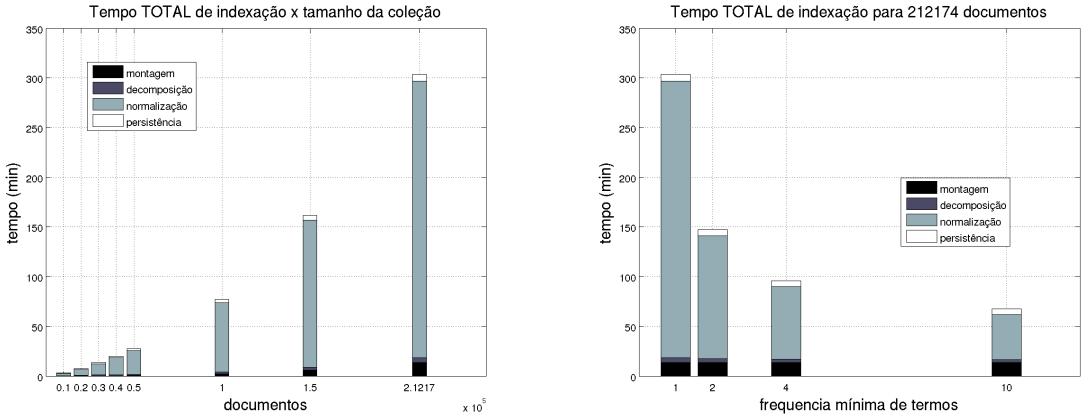


Figura 5.15: Tempos totais acumulados para todo o processo de indexação.

5.2.5 Recuperação da Informação

A implementação real teve a decomposição truncada também em 200 valores singulares, mantendo requisitos de qualidade de informação dentro de patamares aceitáveis. Em testes de conceito simples, realizados junto ao corpo clínico, os resultados se mostraram coerentes. Como pode ser observado na figura 5.16, mesmo para o tamanho considerável da coleção indexada os tempos de busca foram inferiores a um segundo. Mantiveram-se praticamente inalterados tanto para termos com frequência de ocorrência variada, quanto para a quantidade de palavras fornecidas na consulta. Entre os motivos para a inalteração significativa dos tempos pode ser apontada a baixa carga de processamento relativo na avaliação otimizada das consultas, em um computador moderno.

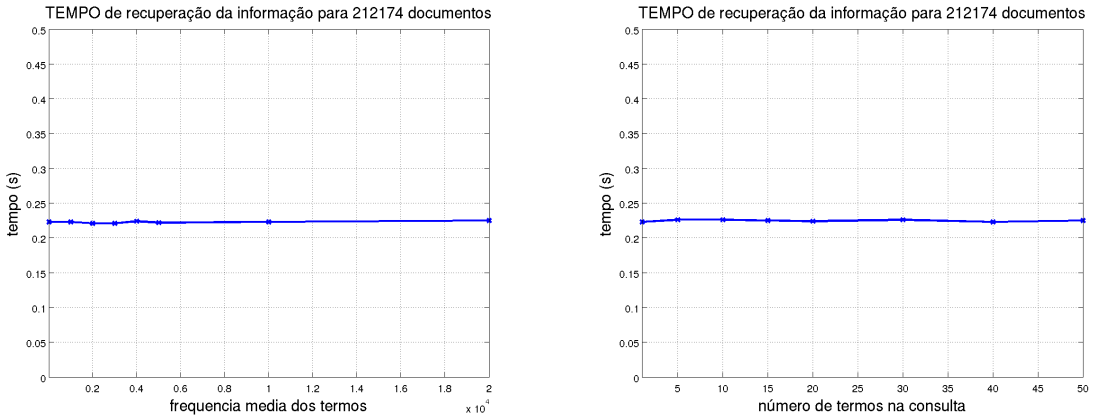


Figura 5.16: Tempos de recuperação da informação para coleção do HUCFF.

A figura 5.17 expõe a interface web apresentada para na prova de conceito utilizando a coleção de documentos indexada oriunda do HUCFF. Visualmente os resultados são diferenciáveis apenas pela quantidade de texto retornado. Internamente, a execução do ambiente web implicou em consumo de memória de trabalho (RAM) significativamente maior, ocupando aproximadamente 6 gigabytes de forma contínua.

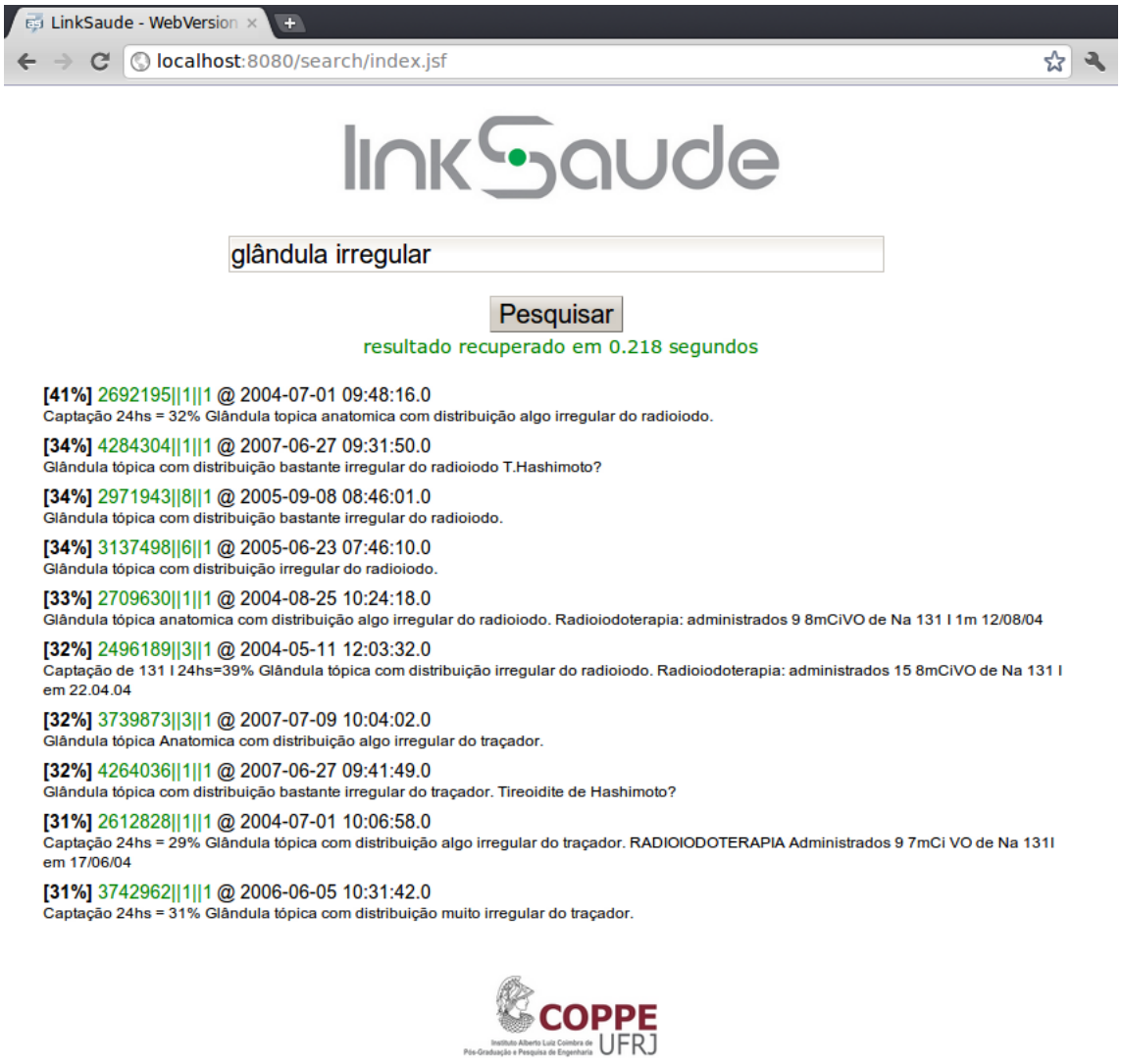


Figura 5.17: Interface web para recuperação da informação no HUCFF.

5.3 Experiência de Uso

Devido ao estágio de desenvolvimento recente da implementação, a experiência de uso, neste trabalho, teve como objetivo principal a validação simples da consistência dos resultados e também a coleta de algumas impressões iniciais do usuário final relacionadas a utilidade usabilidade da ferramenta desenvolvida. Critérios mais estatísticos e subjetivos como análise de relevância e precisão foram dimensionados em trabalhos futuros para garantir realizabilidade desta experiência.

Como metodologia, uma interface de pesquisa simplificada e baseada em modo texto foi oferecida ao médico usuário final. Após passar instruções simples sobre a coleção de documentos completa do HUCFF, a barra de pesquisa e o comando de consultas, algumas pesquisas foram estimuladas. Sem outra ressalva e considerando a experiência popular na web, pesquisas por termos isolados foram executadas.

Inicialmente foi observado impacto nos resultados retornados devido à abrangência do conjunto dos documentos. Pesquisas mais simples e com poucos termos genéricos retornaram tendenciosamente resultados ordenados provenientes de diversos setores do hospital, que causou estranheza ao usuário. Sem ser salientado, o tempo de espera pelo resultado foi considerado razoável e permaneceu abaixo de um segundo em todos os casos.

Ao adquirir maior aprendizado e familiaridade no funcionamento da ferramenta, foi observada a tendência de inserir termos através de refinamento pessoal baseado nas informações retornadas. Com termos mais específicos e em maior número em cada pesquisa, resultados mais focados e coerentes ao objetivo inicial foram alcançados satisfazendo necessidades iniciais pela busca de informações isoladas.

Como conlcusão, esta experiência demonstrou viabilidade e potencial promissor de uso. Conforme esperado, após uma curva de aprendizado rápida proporcionada pela funcionalidade simples e interface limpa resultados coerentes e úteis foram obtidos. Foi sugerido pelo usuário final a possibilidade de configuração para selecionar segmento (área) específica da coleção e também oferecer tratamento estatístico às informações retornadas.

Capítulo 6

Conclusões e Trabalhos Futuros

Inspirado no aprimoramento do conhecimento e desenvolvimento de uma ferramenta útil, este trabalho deixa os seguintes frutos e outras sementes promissoras.

6.1 Contribuições

Integração é a palavra chave neste trabalho de dissertação. As técnicas abordadas durante o estudo e a implementação prática, como extração, *stop-words*, *stemming*, montagem de matriz termo-documento, decomposição em valores singulares e recuperação da informação via análise dos cossenos, já estavam desenvolvidas e utilizadas separadamente em outras aplicações diversas. O trabalho de minerar automaticamente texto em bancos de dados médicos de grande porte utilizando decomposições matriciais contribuiu em desenvolver uma nova estrutura na qual essa engenharia pudesse ser utilizada, na prática, favorecendo outras ciências, como a da saúde.

A combinação do elevado número de ingredientes intrínsecamente complexos exigiu uma receita específica e calibrada, para que o resultado final fosse útil e realizável. Cada detalhe, discutido ao longo deste texto é fruto de algumas tentativas infrutíferas e outras acertadas, sendo escolhida a melhor opção, visando otimização global e flexibilidade de aproveitamento em outras áreas.

O aumento de inteligência integrada, através da recuperação de informação rápida e eficiente, criou a possibilidade de classificar e tratar estatisticamente uma grande massa de dados, antes inacessível devido ao formato de texto não estruturado. A conversão em vetores apropriados para manipulação através de algoritmos numéricos potencializou manipulação através de novas lógicas.

Novos *insights* foram obtidos para estruturação de banco de dados e camadas de negócio em aplicações locais ou distribuídas que favorecem técnicas de trabalho maciço da informação através de melhores normas, consideração de escala e atualização periódica menos onerosa.

Pela alta flexibilidade e potencial teórico comprovado, a ferramenta desenvolvida, fomenta outras utilizações em áreas diversas como reconhecimento de padrões e detecção/tratamento de erros de digitação durante a inserção, além de estímular adaptação desta metodologia para outros fins, em áreas diversas.

6.2 Viabilidade

Um sistema eficiente para recuperação da informação clínica de forma rápida, por si próprio, é útil em instituições de qualquer porte. Consultórios menores em busca de alguma raropatia ou instituições maiores buscando estatísticas mais elaboradas encontrariam uso nesta funcionalidade. Neste sentido, é evidente a viabilidade local da tecnologia desenvolvida, podendo ser personalizada para casos mais específicos.

Numa escala maior, a integração de várias coleções de documentos e outros dados provenientes de sistemas de bancos de dados, distribuídos ou não, forneceriam vasto conteúdo a ser minerado, buscando explorar padrões regionais ou segmentados de determinada população. A comparação entre resultados de diferentes áreas do conhecimento também forneceria fomento a estudos multidisciplinares mais abrangentes e potencialmente proveitosos.

Todas as experiências de mineração realizadas requisitaram recursos de forma modesta e plenamente alcançável pelos padrões atuais de computação. Com uma estrutura convencional de servidores seria possível atender confortavelmente instituições maiores como hospitais e centros de saúde.

Conforme citado por Meenan no trabalho [11], a ampla utilização de ferramentas livres proporcionou uma solução robusta, altamente testada, documentada e de baixo custo. Sem restrições administrativas de licensas de software é possível replicar a infra-estrutura desenvolvida em outros pontos difundindo sua utilização.

Do ponto de vista legal, mesmo operando em conteúdo sigiloso e restrito, o desenvolvimento, teste e utilização da ferramenta não infringiram nenhuma regra clínica estabelecida atualmente. Como esperado, foram preservadas a anonimidade e segurança dos pacientes.

6.3 Trabalhos Futuros

A decomposição em valores singulares fornece a melhor forma de aproximar uma matriz por outra de posto reduzido e mais tratável. A montagem da matriz termo-documento, no entanto, não é única, sendo influênciada por diversos parâmetros de projeto subjetivos e de aferição não trivial. No caso específico da mineração de documentos médicos, o aprimoramento da transformação entre a coleção e a referida matriz permanece como interessante questão a ser aprofundada. Como

citado no texto, várias ponderações relacionadas ao uso de dicionários e avaliação de frequências globais e pesos locais podem ser consideradas com maior efetividade, buscando aumento de desempenho.

Em relação ao código desenvolvido, seria interessante considerar uma implementação multi-tarefa, distribuída e/ou paralela. Apesar de ter apresentado desempenho satisfatório, otimizações deste tipo aumentariam a robustez e elasticidade desta solução testada e consagrada. Estas características permitiram expandir o uso potencialmente em coleções maiores que as tratadas além de viabilizar o uso em áreas mais computacionalmente intensivas em processamento, como o tratamento de imagens ou objetos tridimensionais.

Conceitualmente, a ferramenta desenvolvida nesta disertação possibilita o estudo de uma metodologia para tratamento estatístico científico de variáveis de interesse identificadas em textos minerados. De acordo com o interesse, este tratamento poderia ter foco clínico, como na detecção e prevenção de padrões de patologia ou pacientes, ou administrativos, verificando a quantidade de vezes aproximadamente determinada droga ou medicamento foi citado em documentos específicos.

Os agrupamentos ou *clustering* são técnicas matemáticas e computacionais que permitem agrupar objetos (vetores-documento) em conjuntos, considerando semelhanças explícitas ou implícitas. Uma vez que a base de dados a ser minerada está mapeada em uma matriz, realizar o agrupamento não é difícil, podendo utilizar algoritmos como, por exemplo, o de vizinhos mais próximos ou KNN[1]. Decomposições, como a de valores singulares utilizada nesta dissertação, oferecem ainda maior eficiência na formação de grupos, devido ao filtro de ruídos implícito na decomposição.

Finalmente, todo o arsenal disponível poderia ser estruturado de forma a potencializar a tomada de decisões clínicas, principalmente em momentos críticos, pela gravidade ou tempo. Desta forma, a vida e o bem estar social, objetivos nobres de todas as ciências seriam mantidos em patamares mais confortáveis.

[1] *K-Nearest Neighbor*

Apêndice A

Códigos Fonte em Matlab

A.1 Exemplo Simplificado

```
% Initial setup
clc; clear all; close all;
o = [0 0 0];

% Documents after stop-words and stemming
d1 = [1 1 0];
d2 = [0 1 1];
d3 = [1 0 1];
d4 = [1 1 1];
q  = [0 1 2];  % query

% Drawing documents
figure(1);
arrow(o,d1,'X:matem t','Y:control','Z:comput','blue'); hold on;
arrow(o,d2,'X:matem t','Y:control','Z:comput','blue'); hold on;
arrow(o,d3,'X:matem t','Y:control','Z:comput','blue'); hold on;
arrow(o,d4,'X:matem t','Y:control','Z:comput','blue'); hold on;

% Drawing query in vectorial space
figure(2);
arrow(o,d1,'X:matem t','Y:control','Z:comput','blue'); hold on;
arrow(o,d2,'X:matem t','Y:control','Z:comput','blue'); hold on;
arrow(o,d3,'X:matem t','Y:control','Z:comput','blue'); hold on;
arrow(o,d4,'X:matem t','Y:control','Z:comput','blue'); hold on;
arrow(o,q, 'X:matem t','Y:control','Z:comput','red'); hold on;

% Angles (relevance) between documents and query
cos1 = (d1*q') / (norm(d1)*norm(q));
cos2 = (d2*q') / (norm(d2)*norm(q));
cos3 = (d3*q') / (norm(d3)*norm(q));
cos4 = (d4*q') / (norm(d4)*norm(q));
```

```
32
33 % Term-by-document matrix
34 A = [d1' d2' d3' d4'];
35
36 % Truncated SVD: tridimensional -> bidimensional
37 [U, S, V] = svds(A, 2);
38
39 % Optimal plane according to SVD: n = [a b c]'
40 Ua = U(:,1);
41 Ub = U(:,2);
42 a = Ua(2)*Ub(3) - Ua(3)*Ub(2);
43 b = Ua(3)*Ub(1) - Ua(1)*Ub(3);
44 c = Ua(1)*Ub(2) - Ua(2)*Ub(1);
45
46 figure(3);
47 arrow(o,d1,'X:matem t','Y:control','Z:comput','blue'); hold on;
48 arrow(o,d2,'X:matem t','Y:control','Z:comput','blue'); hold on;
49 arrow(o,d3,'X:matem t','Y:control','Z:comput','blue'); hold on;
50 arrow(o,d4,'X:matem t','Y:control','Z:comput','blue'); hold on;
51 arrow(o,q, 'X:matem t','Y:control','Z:comput','red'); hold on;
52
53 % Drawing plane
54 [X,Y] = meshgrid(0:0.1:1);
55 Z = -(a*X + b*Y)/c;  % plane equation: ax + by + cz = d, but d = 0;
56 mesh(X,Y,Z);
57
58 USV = U*S*V';
59
60 cos1s = USV(:,1)'*q'/(norm(USV(:,1))*norm(q));
61 cos2s = USV(:,2)'*q'/(norm(USV(:,2))*norm(q));
62 cos3s = USV(:,3)'*q'/(norm(USV(:,3))*norm(q));
63 cos4s = USV(:,4)'*q'/(norm(USV(:,4))*norm(q));
64
65 % Frobenius norm
66 frobA   = norm(A, 'fro')
67 frobSVD = norm((U*S*V'), 'fro')
68 error = (frobA - frobSVD) / frobA
69
70 [cos1 cos1s (cos1-cos1s);
71  cos2 cos2s (cos2-cos2s);
72  cos3 cos3s (cos3-cos3s);
73  cos4 cos4s (cos4-cos4s)]
```

A.2 Prova de Conceito

```
1  % ---------- Initial setup ----------
2  clear all; close all; clc;
3  m = 262;
4  n = 250;
5  nzmax = 983;
6  nsig = 212;
7  trunc = 200;
8
9  % ---------- Load matrix A ----------
10 fa = fopen('dados/A');
11 fgets(fa); fgets(fa); fgets(fa); fgets(fa);  % Discard first four lines
12 colptr = fscanf(fa, '%d', [1 n+1 ]);
13 rowind = fscanf(fa, '%d', [1 nzmax]);
14 value  = fscanf(fa, '%f', [1 nzmax]);
15
16 for a = 1:n
17     ki = colptr(a);
18     kf = colptr(a+1) - 1;
19     for b = ki:kf
20         j(b) = a;
21     end
22 end
23
24 A = sparse(rowind, j, value, m, n, nzmax);
25 fclose(fa);
26
27 figure(1);
28 spy(A);
29 %title('Matriz termo-documento do CID-10 capitulo 7')
30 xlabel(sprintf('NonZeros=%d (%.3f%%)',nnz(A), 100*(nnz(A)/numel(A))) );
31
32
33 % ---------- Load matrix U ----------
34 fu = fopen('dados/U');
35 Uc = fscanf(fu, '%g', [nsig m]);
36 Uc = Uc';
37 Uc = Uc(1:m, 1:trunc);
38 fclose(fu);
39
40 % ---------- Load matrix S ----------
41 fs = fopen('dados/S');
42 Sc = diag( fscanf(fs, '%g', [1 nsig]) );
43 Sc = Sc(1:trunc, 1:trunc);
44 fclose(fs);
45
```

```
46 % ---------- Load matrix V ----------
47 fv = fopen('dados/V');
48 Vc = fscanf(fv, '%g', [nsig n]);
49 Vc = Vc';
50 Vc = Vc(1:n, 1:trunc);
51 fclose(fv);
52
53
54 % ---------- SVD ----------
55 [Um, Sm, Vm] = svds(A, trunc);
56 Akm = Um*Sm*Vm';  % Matlab calculations (from previous svds() command)
57 Akc = Uc*Sc*Vc';  % ANSI C calculations (from files loaded)
58
59 % ---------- Sigma decay ----------
60 figure(2);
61 d = ones(1,trunc)';
62 plot(Sc*d);  grid on;
63 %title('Decaimento dos valores singulares na cole  o CID-10 cap tulo 7');
64 xlabel('n mero de valores singulares calculados');
65 ylabel('m dulo');
66
67 % ---------- Frobenius Norm Comparation ----------
68 DeltaU = norm((Um-Uc),'fro') % Matrix U (can be different)
69 DeltaS = norm((Sm-Sc),'fro') % Matrix S (should be the same)
70 DeltaV = norm((Vm-Vc),'fro') % Matrix V (can be different)
71
72 ErrorAc = norm(A-Akc,'fro') / norm(A,'fro')  % Matrix A (ANSI C)
73 ErrorAm = norm(A-Akm,'fro') / norm(A,'fro')  % Matrix A (Matlab)
74
75 DeltaAc_frob = norm((A-Akc),'fro')  % Matrix A (ANSI C)
76 DeltaAm_frob = norm((A-Akm),'fro')  % Matrix A (Matlab)
77 DeltaAc_eucl = norm(A-Akc)  % Matrix A (ANSI C)
78 DeltaAm_eucl = norm(A-Akm)  % Matrix A (ANSI C)
79
80
81 % ---------- Sigma element-by-element Comparation ----------
82 deltaS = [(Sc*d) (Sm*d) (Sc*d-Sm*d)];
83 deltaS(1:10, 1:3)
84
85 % ---------- Evaluating losted data during truncate ----------
86 [Ut, St, Vt] = svds(A, n);     % Full rank sparse SVD, for test
87 Sr = St*ones(length(St), 1);  % Put all singular values in a vector
88 Sr = Sr(trunc+1:length(Sr));  % Discarded singular values vector
89 trash = norm(Sr,'fro')
90 error = DeltaAc_frob - trash  % Delta between A-Ak Frobenius norm and trash
91
92 % ---------- Information lost (HARD WAY) ----------
```

```
93  for i=1:trunc
94      [Ui, Si, Vi] = svds(A, i);
95      Ai = Ui*Si*Vi';
96      info(i) = ( norm(A-Ai,'fro') / norm(A,'fro') );
97  end
98  figure(3);
99  plot(info); grid on;
100 xlabel('quantidade de valores singulares utilizados');
101 ylabel('percentual da informa  o alterada');
```

Referências Bibliográficas

[1] BERRY, M., BROWNE, M., 2005, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. 2 ed. , SIAM.

[2] CLEVERDON, C., 1984, "Optimizing Convenient Online Access to Bibliographic Databases", *Information Services and Use*, v. 4 (April), pp. 1–2.

[3] DUMAIS, S. T., 1991, "Improving the Retrieval of Information from External Sources", *Behavior Research Methods, Instruments Computers*, v. 23, n. 2, pp. 229–236.

[4] ECKART, C., YOUNG, G., 1936, "The Approximation of One Matrix by Another of Lower Rank", *Psychometrika*, v. 1, n. 3 (September), pp. 211–218.

[5] ELDÉN, L., 2007, *Matrix Methods in Data Mining and Pattern Recognition*. 1 ed. , SIAM.

[6] ERINJERI, J., PICUS, D., PRIOR, F., etal., 2009, "Development of a Google-Based Search Engine for Data Mining Radiology Reports", *Journal of Digital Imaging*, v. 22, n. 4, pp. 348–356.

[7] GOLUB, G. H., LOAN, C. F. V., 1996, *Matrix Computations*. 3 ed. Baltimore, Maryland, The Johns Hopkins University Press.

[8] HURLEN, P., OSTBYE, T., BORTHNE, A., etal., 2009, "Do clinicians read our reports? Integrating the radiology information system with the eletronic patient record: experiences from the first 2 years", *Eur Radiol*, pp. 31–36.

[9] JÚNIOR., G. O. A., 2009, *Implementação em Paralelo do Método de Arnoldi/Lanczos com Reinício Implícito*. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

[10] KORFHAGE, R. R., 1997, *Information Storage and Retrieval*. 1 ed. New York, John Wiley Sons.

[11] MEENAN, C., KING, A., TOLAND, C., etal., 2010, "Use of a Wiki as a Radiology Departmental Knowledge Management System", *Journal of Digital Imaging*, v. 23, n. 2, pp. 142–151.

[12] PAGE, L., BRIN, S., MOTWANI, R., etal., 1999, "The PageRank Citation Ranking: Bringing Order to the Web", *Technical Report*.

[13] PRESSMAN, R. S., 2007, *Software Engineering: A Practitioner's Approach*. 6 ed. Boca Raton, Florida, R.S. Pressman Associates, Inc.

[14] PREVEDELLO, L. M., ANDRIOLE, K. P., HANSON, R., etal., 2010, "Business Intelligence Tools for Radiology: Creating a Prototype Model Using Open-Source Tools", *Journal of Digital Imaging*, v. 23, n. 2, pp. 133–141.

[15] RAMON, J., FIERENS, D., GUIZA, F., etal., 2007, "Mining Data From Intensive Care Patients", *Elsevier*.

[16] RUBIN, D., 2009, "Informatics Methods to Enable Patient-centered Radiology", *Acad Radiol*, v. 16, pp. 524–534.

[17] STRANG, G., 2006, *Linear Algebra and its Applications*. 4 ed. , Thomson Brooks/Cole.

[18] VOET, T., P.DEVOLDER, PYNOO, B., etal., 2007, "Design and Implementation of an Open Source Indexing Solution for a Large Set of Radiological Reports and Images", *Journal of Digital Imaging*, v. 20, n. 2, pp. 11–20.

[19] WURMAN, R. S., 1996, *Information Architects*. 1 ed. Zurich, Graphis Press Corporation.